Computers & Industrial Engineering 85 (2015) 102–109

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability ☆

Cheol Min Joo [a], Byung Soo Kim [b,*]

[a] Department of Industrial and Management Engineering, Dongseo University, San 69-1, Jurye-dong, Saanmg-gu, Busan 617-716, Republic of Korea
[b] Department of Industrial and Management Engineering, Incheon National University, 119, Academy-ro, Songdo-dong, Yeonsu-gu, Incheon 406-772, Republic of Korea

ARTICLE INFO

Article history:
Received 19 August 2014
Received in revised form 7 January 2015
Accepted 28 February 2015
Available online 7 March 2015

Keywords:
Genetic algorithm
Dispatching rule
Unrelated parallel machine scheduling
Sequence- and machine-dependent setup time
Production availability

ABSTRACT

This article considers the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup times and machine-dependent processing times. Furthermore, the machine has a production availability constraint to each job. The objective of this problem is to determine the allocation policy of jobs and the scheduling policy of machines to minimize the total completion time. To solve the problem, a mathematical model for the optimal solution is derived, and hybrid genetic algorithms with three dispatching rules are proposed for large-sized problems. To assess the performance of the algorithms, computational experiments are conducted and evaluated using several randomly generated examples.

1. Introduction

Companies that manufacture goods based on orders generally have several machine groups consisting of simple-purpose machines or general-purpose machines that are similar function but differ shape and performance. Therefore, each (ordering) job can or cannot be processed on a specific machine, and the processing time of the job depends upon the machine to which it is assigned. This physical limitation of machines is called *production availability constraint* of the machine. In the manufacturing companies, the consideration of setup times between jobs is dependent upon not only the sequence of jobs to be processed on a machine but also the machine to which the jobs are assigned. In this article, we consider unrelated parallel machine scheduling with sequence- and machine-dependent setup times, machine dependent processing times, and production availability constraints.

Several studies have investigated unrelated parallel machines without consideration of setup time. Rogendran and Subur (2004) proposed a tabu-search-based heuristic minimizing the total weighted tardiness in the unrelated parallel machine scheduling problem with job splitting and dynamic machine availability. Agarwal, Colak, Jacob, and Pirkul (2006) suggested 12 combinations of single-pass heuristics and a neural network approach for minimizing a makespan for non-identical parallel machines. Balin (2011) presented a genetic algorithm (GA) for non-identical parallel machine scheduling to minimize the makespan of the machines without having setup times, ready times, and due times. In this article, the authors propose a new crossover operator and a new optimality criterion in order to adapt GA to the non-identical parallel machine scheduling problem. Lin, Pfund, and Fowler (2011) also proposed a GA for unrelated parallel machine problem to minimize the total weighted tardiness. Lin, Lin, and Hsieh (2013) proposed ant colony optimization incorporating a number of new ideas such as heuristic initial solution, machine reselection step, and local search procedure to solve the problem of scheduling unrelated parallel machine to minimize total weighted tardiness. Rodriguez, Lozano, Blum, and Garcia-Martinez (2013) proposed an iterated greedy algorithm for the large-scale unrelated parallel machine scheduling problem without consideration of setup. They tested instances with up to 1000 jobs and 50 machines and showed relatively good performance compared to other meta-heuristics. Yang (2013), Yang and Yang (2013), Yang, Cheng, and Yang (2013), Yang, Hsu, and Yang (2014) consider unrelated parallel-machine scheduling with several maintenance activities and deterioration effects to minimize the total completion time without set-up consideration.

On the other hand, several studies have investigated unrelated parallel machine scheduling problems with consideration of setup time. Weng, Lu, and Ren (2001) presented seven constructive procedures for the unrelated parallel machine scheduling problem for minimizing total completion time and performed an experimental comparison between them. Vredeveld and Hurkens (2002)

☆ This manuscript was processed by Area Editor Mitsuo Gen.

* Corresponding author. Tel.: +82 32 835 8482; fax: +82 32 835 0777.
E-mail address: bskim@incheon.ac.kr (B.S. Kim).

http://dx.doi.org/10.1016/j.cie.2015.02.029
0360-8352/

proposed two meta-heuristics based on local search. The local search procedures make use of two types of different neighborhood functions. Kim, Kim, Jang, and Chen (2002) presented a scheduling problem for unrelated parallel machines with sequence- and machine-dependent setup times to minimize total tardiness using simulated annealing (SA). Hop and Nagarur (2004) considered non-identical parallel machines with sequence-dependent times to minimize makespan in a printed-circuit board (PCB) production line. They developed a mathematical model and propose a genetic algorithm. Li and Yang (2009) considered non-identical parallel machine scheduling to minimize total weighted completion time. Tavakkoli-Moghaddam, Taheri, Bazzazi, Izadi, and Sassani (2009) proposed a genetic algorithm for minimizing the number of tardy jobs and the total completion time constrained by minimizing the number of tardy jobs for an unrelated parallel machine scheduling problem with sequence-dependent and machine-dependent setup times, ready times, and due-times. Chen and Chen (2009) presented a hybrid metaheuristic combining tabu search and variable neighborhood descent to minimize the total weighted tardiness on unrelated parallel machine. Gharehgozli, Tavakkoli-Moghaddam, and Zaerpour (2009) presented a new mixed-integer goal programming model to minimize the total weighted flow time and the total weighted tardiness simultaneously for a non-identical parallel machine scheduling problem with sequence-dependent setup times, ready times, and due-times. Ko, Kim, Kim, and Baek (2010) proposed a dispatching rule that guarantees a predetermined minimum quality level for non-identical parallel machines with multiple product types. They only considered sequence-dependent setup times between product types. Vallada and Ruiz (2011) presented hybrid genetic algorithms to minimize makespan including a fast local search and a local search enhanced crossover operator based on a two-dimensional representation of a chromosome for the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup time. Ruiz and Andres-Romano (2011) investigated the unrelated parallel machine problem with resource-assignable sequence and machine dependent setup times. Their new characteristic is that the amount of setup time depends not only upon the machine and job sequence but also upon the number of resources assigned. The objective function of the study is a linear combination of the total completion time and the number of human resources assigned to the setup. They developed a mixed-integer programming (MIP) model and proposed some fast dispatching heuristics. Lin and Lin (2013) considered an unrelated parallel scheduling problem with sequence- and machine-dependent setup times and release date for minimizing makespan, total weighted completion time, and total weighted tardiness. They developed a mixed-integer programming model for an optimal solution and proposed several dispatching rules to find good solutions quickly.

To the best of our knowledge, the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup time and production availability of machines constraints has not been considered previously in the literature. The remainder of this article is organized as follows. After a brief problem description, a mathematical model for finding an optimal solution is derived in Section 2. As the solution approaches, hybrid GAs are proposed with three dispatching rules for assigning jobs to machines in Section 3. Section 4 evaluates the performances of the GAs on the basis of computational experiments. Finally, Section 5 summarizes the study and discusses for further research.

2. Mathematical model

In this section, we propose an MIP model for the unrelated parallel machine scheduling problem with sequence and machine dependent setup times, machine dependent processing times, and machine availability constraints. The objective is to minimize the total completion time of all jobs within a given working horizon. The following notations are used for the MIP model:

Indices

SJ	: set of jobs to be scheduled.
SM	: set of machines.
h, i, j	: job indices, where $h, i, j \in SJ$.
k	: machine index, where $k \in SM$.

Parameters

p_{ik}	: machine-dependent processing time of job i at machine k.
s_{ijk}	: sequence- and machine-dependent setup time of job j immediately following job i in the sequence at machine k.
u_{ik}	equal to 1, if job$i$ is available for production at machine k; 0, otherwise.
H	: working horizon.
M	: big positive number.

Continuous variables

x_i	: starting time of job i.
C_k	: completion time of machine k.

Binary variables

y_{ik}	: equal to 1, if job i is assigned to machine k; 0, otherwise.
z_{ijk}	: equal to 1, if job i precedes job j at machine k; 0, otherwise.

The notations s_{iik} and z_{iik} have a special use in the model. The parameter s_{iik} denotes the setup time of job i, if the job is the first job of machine k and the variable z_{iik} equals 1, if job i is the first job of machine k. Fig. 1 describes an example of an unrelated parallel machine schedule. The sum of sequence- and machine-dependent setup times and machine dependent processing times of the assigned jobs $h, i, j, \ldots$ to machine k becomes the completion time of the machine C_k. The total completion time for the schedule is the sum of the completion times of all the machines.

The MIP model for assigning jobs to each machine and sequencing jobs for the corresponding machine to minimize the total completion time is as follows:

$$Min\ z = \sum_{k \in SM} C_k \tag{1}$$

$s.t.$

$$x_i + \sum_{k \in SM} \left(\sum_{h \in SJ} s_{hik} \cdot z_{hik} + p_{ik} \cdot y_{ik} \right) \leqslant x_j + M \cdot \left(1 - \sum_{k \in SM} z_{ijk} \right), \quad for\ \forall i, j \in SJ; j \neq i \tag{2}$$

$$x_i + \sum_{h \in SJ} s_{hik} \cdot z_{hik} + p_{ik} \cdot y_{ik} \leqslant C_k + M \cdot (1 - y_{ik}) \quad for\ \forall i \in SJ; \forall k \in SM \tag{3}$$

$$\sum_{k \in SM} u_{ik} \cdot y_{ik} = 1, \quad for\ \forall i \in SJ \tag{4}$$

$$\sum_{k \in SM} y_{ik} = 1, \quad for\ \forall i \in SJ \tag{5}$$

$$\sum_{i \in SJ} z_{iik} \leqslant 1, \quad for\ \forall k \in SM \tag{6}$$

$$\sum_{j \in SJ} z_{jik} = y_{ik}, \quad for\ \forall i \in SJ; \forall k \in SM \tag{7}$$

$$\sum_{\substack{j \in SJ \\ j \neq i}} z_{ijk} \leqslant y_{ik}, \quad for\ \forall i \in SJ; \forall k \in SM \tag{8}$$

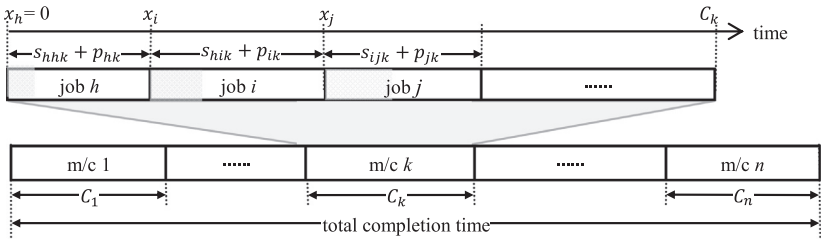


Fig. 1. Unrelated parallel machine schedule.

$$x_i \geqslant 0, \quad for\ \forall i \in SJ \tag{9}$$
$$C_k \leqslant H, \quad for\ \forall k \in SM \tag{10}$$
$$y_{ik} = 0 \text{ or } 1, \quad for\ \forall i \in SJ; \forall k \in SM \tag{11}$$
$$z_{ijk} = 0 \text{ or } 1, \quad for\ \forall i,j \in SJ; \forall k \in SJ \tag{12}$$

Constraint (2) ensures the precedence relation of jobs assigned to the same machine and calculates the starting time of each job. Constraint (3) calculates the completion time of each machine by calculating the sequence- and machine-dependent setup times and machine-dependent processing times of all jobs assigned to the machine. Constraint (4) ensures that each job will be processed by one of the available machines. Constraint (5) confirms that each job is processed in exactly one machine. Constraints (6)–(8) ensure that all jobs assigned to the same machine must appear once in 8their sequence. Constraint (10) ensures that the completion time of each machine is less than or equal to the working horizon.

3. Genetic algorithms

The MIP model presented in the previous section is not tractable for the unrelated parallel machine scheduling problem because it is a typical combinatorial optimization problem. Furthermore, the simplification of this model, the problem $P|S_j|\sum C$ is already NP-hard (Webster, 1997). Thus, we focus on developing effective meta-heuristic approaches instead, and propose GAs.

In a GA, the representation of a solution, a *chromosome*, has a great influence on the performance of the algorithm. The chromosome for the unrelated parallel machine scheduling problem considered in this article must simultaneously represent the job assignments to machines and the sequences of assigned jobs in each machine. There are two popular chromosomes for representing the assigning and sequencing, one that consists of a double-dimensional string array for a sequence array (Liu & Wu, 2003; Omara & Arafa, 2010) and assignment array and the other that consists of a

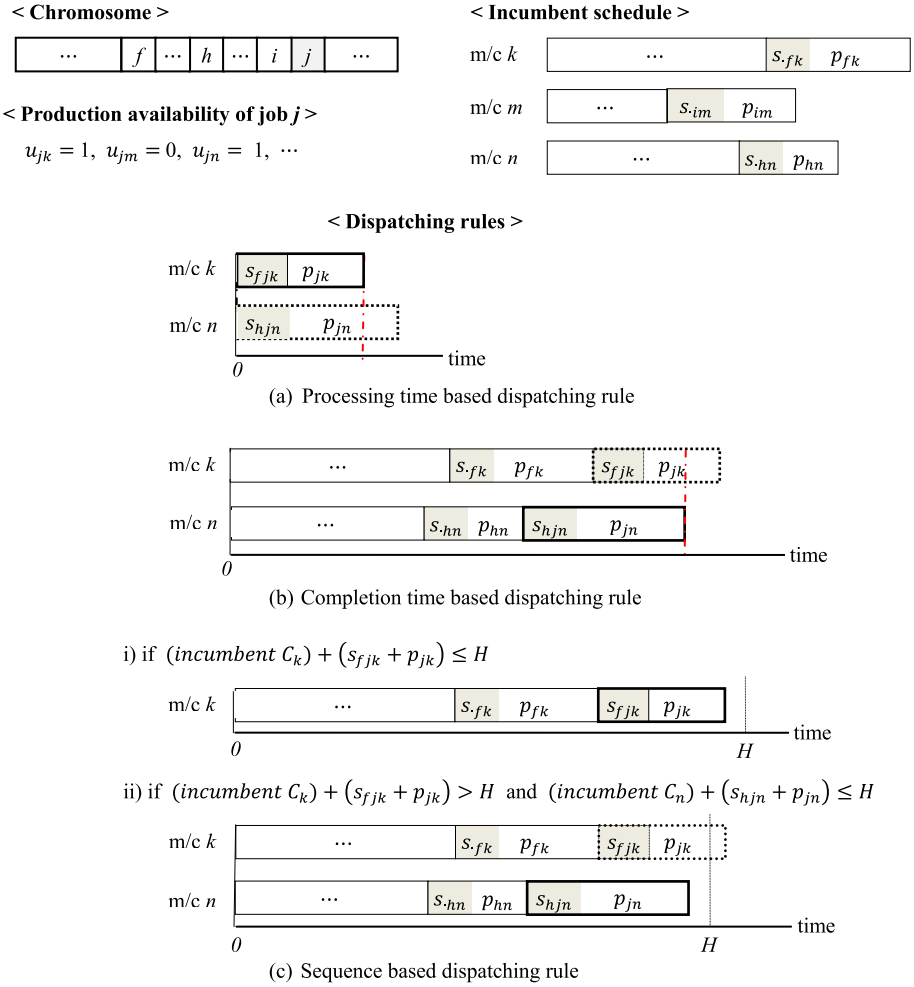


Fig. 2. The dispatching rules.

single-dimensional string array with a special character as a machine-separation indicator (Cheng & Gen, 1997; Tavakkoli-Moghaddam et al., 2009; Lin et al., 2011). However, the major drawback of these chromosomes is the difficulty in maintaining feasibility during genetic operations, because the solutions from the chromosomes must satisfy production availability constraints. To avoid this drawback, we propose hybrid GAs with dispatching rules (GA_DRs). GA_DR is introduced by Lee, Kim, and Joo (2012). The basic idea of GA_DR is that the chromosome consists only of a single-dimensional string array for representing a job order for applying a dispatching rule, and the dispatching rule determines the job assignment to a specific machine. Hence, the occurrence of infeasible solutions caused by the production availability constraints could be avoided during application of the dispatching rule. Lee et al. (2012) tested GA_DR with a completion-time-based dispatching rule for the parallel scheduling problem and the truck scheduling problem, and showed that GA_DR provides higher quality performance than the GAs with traditional two popular chromosome representations, in both effectiveness and efficiency. But the performance of GA_DR is highly dependent upon the method applying for the dispatching. So, in order to good performance of GA_DR, we have to propose a good dispatching rule for a specific problem.

In this article, we propose three different dispatching rules of GA_DR for the unrelated parallel machine scheduling problem, which are the processing-time-based dispatching rule, the completion-time-based dispatching rule, and the sequence-based dispatching rule. In Fig. 2, the three dispatching rules with a chromosome, a corresponding incumbent schedule, and production availability constraints are illustrated. The incumbent schedule in the figure implies the schedule of jobs before job j in the chromosome with a given dispatching rule. Fig. 2(a) illustrates the processing-time-based dispatching rule. To assign job j in the chromosome to one of its available machines k or n, the processing times $s_{fjk}+p_{jk}$ and $s_{hjn}+p_{jn}$ are compared and assigned to machine k with the smallest processing time. The general pseudo code of the processing-time-based dispatching rule is as follows:

Procedure: processing-time-based dispatching rule
Inputs: a chromosome with ordering jobs $j_1, j_2, \cdots, j_n$ and p_{ik}, s_{ijk}, u_{ik}

Output: a corresponding schedule
Begin
Let current final job of machine $k, F_k^0 = \emptyset$.
Let order index θ = 1.
While ($\theta \leqslant n$)
Let machine index k = 1.
While ($k \leqslant$ *number of machines*)
If ($u_{(j_\theta)(k)}$ = 1)
Calculates temporary processing time $s_{(F_k^{\theta-1})(j_\theta)(k)} + p_{(j_\theta)(k)}$.
End If
$k = k + 1$.
End While
Select the machine δ with the smallest temporary processing time.
$F_\delta^\theta = j_\theta$.
$\theta = \theta + 1$.
End While
End

Fig. 2(b) illustrates the completion-time-based dispatching rule. To assign job j in the chromosome to one of its available machines k or n, candidate schedules of the machines are compared. The candidate schedule of a machine is the schedule in which the job j is assigned to the end of sequence in the corresponding machine. The temporary completion times of all the candidate schedules are calculated, and then the job j is assigned to the machine n with the smallest temporary completion time. The general pseudo code of the completion-time-based dispatching rule is as follows:

Procedure: completion -time-based dispatching rule
Inputs: a chromosome with ordering jobs $j_1, j_2, \cdots, j_n$ and p_{ik}, s_{ijk}, u_{ik}

Output: a corresponding schedule
Begin
Let current completion time of machine $k, C_k^0 = 0$.
Let current final job of machine $k, F_k^0 = \emptyset$.
Let order index θ = 1.
While ($\theta \leqslant n$)
Let machine index k = 1.
While ($k \leqslant$ *number of machines*)
If ($u_{(j_\theta)(k)}$ = 1)
Calculates temporary completion time $C_k^{\theta-1} + s_{(F_k^{\theta-1})(j_\theta)(k)} + p_{(j_\theta)(k)}$.
End If
$k = k + 1$.
End While
Select the machine δ with the smallest temporary completion time.
$C_\delta^\theta = C_\delta^{\theta-1} + s_{(F_\delta^{\theta-1})(j_\theta)(\delta)} + p_{(j_\theta)(\delta)}$
$F_\delta^\theta = j_\theta$.
$\theta = \theta + 1$.
End While
End

Fig. 2(c) illustrates the sequence-based dispatching rule. To assign job j in the chromosome to one of its available machines k and n, the temporary completion time of the candidate schedules of the machines are compared sequentially with working horizon H. If the temporary completion time of the candidate schedule of machine k is less than or equal to H, the job j is assigned to the machine k. Otherwise, the comparison is completed with machine n. The general pseudo code of the sequence-based dispatching rule is as follows:

Procedure: sequence-based dispatching rule
Inputs: a chromosome with ordering jobs $j_1, j_2, \cdots, j_n$ and $p_{ik}, s_{ijk}, u_{ik}, H$

Output: a corresponding schedule
Begin
Let current completion time of machine $k, C_k^0 = 0$.
Let current final job of machine $k, F_k^0 = \emptyset$.
Let order index θ = 1.
While ($\theta \leqslant n$)
Let machine index k = 1.
While ($k \leqslant$ *number of machines*)
if ($u_{(j_\theta)(k)}$ = 1)
Calculates temporary completion time $CT = C_k^{\theta-1} + s_{(F_k^{\theta-1})(j_\theta)(k)} + p_{(j_\theta)(k)}$.
If ($CT \leqslant H$)
Selected machine $\delta = k$.
Break

(*continued on next page*)

End If
$k = k + 1$.
End While
$C_{\delta}^{\theta} = C_{\delta}^{\theta-1} + s_{(F_{\delta}^{\theta-1})(j_{\theta})(\delta)} + p_{(j_{\theta})(\delta)}$
$F_{\delta}^{\theta} = j_{\theta}$.
$\theta = \theta + 1$.
End While
End

The hybrid GAs with three dispatching rules are called, GA using chromosomes with processing-time-based dispatching rule (GA_DR_P), GA using chromosomes with completion-time-based dispatching rule (GA_DR_C), and GA using chromosomes with sequence-based dispatching rule (GA_DR_S). Each generation of GAs has a set of chromosomes, called the population of the generation. We randomly generate an initial population for the first generation, and the chromosomes in the population are evaluated using a fitness value. The total completion time for the corresponding schedule with a given dispatching rule from a chromosome is used as a measure of fitness. The population of the next generation is primarily composed of the best chromosomes migrating from the current generation and new chromosomes (i.e., children) reproduced by crossover and mutation operators (with randomly selected parents from the population of the current generation). The *one-point crossover* and *swap mutation* is used for all of the GAs in this article. For one-point crossover, one point is randomly selected for dividing one parent. The set of genes on the left side is passed on from the parent to the child, and the other genes are placed in order of their appearance in the other parent. For swap mutation, two genes of a randomly selected parent are interchanged. The population of next generation is composed of the chromosomes selected by a roulette-wheel from the children. The next generation is evaluated, and this process is repeated until a stopping criterion (maximum number of generations) is met. The general pseudo code of the hybrid GAs are as follows:

Procedure: the hybrid GA (HGA)
Inputs: crossover rate P_C, mutation rate P_M, population size S_{pop}, and maximum generation G_{max}.

Output: the near optimal solution
Begin
Let generation index θ = 1.
Randomly generate an initial population for the first generation.
While ($\theta \leqslant G_{max}$)
Let population index δ = 1.
While ($\delta \leqslant S_{pop}$)
// make children
Randomly select two chromosomes from current population.
If (random number $\leqslant P_C$)
Do one-point crossover operations.
End If
If (random number $\leqslant P_M$)
Do Swap mutation operations.
End If
// Evaluation
Apply one of the ***dispatching rules*** to make corresponding schedule of each child chromosome.
Calculate the fitness value of each schedule.
$\delta = \delta + 1$.
End While
Construct the next generation by a roulette-wheel from the children.
$\theta = \theta + 1$.
End While
End

Table 1
Test results of small sized problems.

Jobs/MC	*MCs*	CPLEX		GA_DR_P			GA_DR_C			GA_DR_S		
		Opt.	Time (s)	*RPD* (%)	*MAD* (%)	Time (s)	*RPD* (%)	*MAD* (%)	Time (s)	*RPD* (%)	*MAD* (%)	Time (s)
2	2	817	1.332	0.00	0.00	0.024	0.00	0.00	0.023	0.00	0.00	0.020
		830	1.099	0.00	0.00	0.020	0.00	0.00	0.021	0.12	0.02	0.019
		717	1.109	0.00	0.00	0.018	0.00	0.00	0.020	0.00	0.00	0.018
	3	1115	2.005	0.00	0.00	0.042	0.22	0.03	0.044	0.00	0.00	0.036
		1168	2.784	0.33	0.02	0.048	0.09	0.02	0.051	0.34	0.03	0.043
		1121	1.665	0.00	0.00	0.044	0.00	0.00	0.045	0.11	0.02	0.040
	4	1427	3.252	0.00	0.00	0.085	0.07	0.01	0.092	0.19	0.03	0.072
		1433	2.218	0.00	0.00	0.083	0.00	0.00	0.087	0.08	0.02	0.070
		1400	3.335	0.00	0.00	0.083	0.00	0.00	0.085	0.00	0.00	0.069
3	2	926	2.198	0.00	0.00	0.038	0.00	0.00	0.038	0.09	0.01	0.035
		758	1.656	0.03	0.00	0.039	0.29	0.00	0.041	0.05	0.01	0.035
		872	1.645	0.00	0.00	0.037	0.00	0.00	0.038	0.00	0.00	0.035
	3	945	4.403	0.00	0.00	0.093	0.00	0.00	0.098	0.17	0.01	0.083
		1087	4.397	0.01	0.00	0.098	0.02	0.00	0.103	4.07	0.04	0.085
		1098	2.771	0.00	0.00	0.093	0.36	0.00	0.097	0.02	0.00	0.083
	4	1456	195.042	0.00	0.00	0.193	0.15	0.00	0.204	2.26	0.02	0.149
		1418	35.683	0.01	0.00	0.185	0.25	0.02	0.195	0.83	0.01	0.156
		1537	19.066	0.77	0.02	0.158	0.61	0.06	0.189	28.17	3.85	0.143
4	2	844	1.235	0.17	0.01	0.065	0.05	0.01	0.066	0.14	0.01	0.060
		726	2.201	0.19	0.03	0.067	1.20	0.02	0.069	0.10	0.02	0.062
		758	3.290	0.00	0.00	0.066	0.03	0.00	0.069	0.16	0.01	0.062
	3	1115	486.669	1.31	0.15	0.169	2.86	0.22	0.177	4.58	0.01	0.147
		892	83.403	0.09	0.01	0.166	0.18	0.02	0.175	17.01	0.00	0.144
		1086	506.013	0.14	0.01	0.169	0.64	0.01	0.179	0.39	0.01	0.145
	4	NA	3600.000	0.06	0.00	0.342	0.51	0.03	0.361	2.81	0.04	0.282
		NA	3600.000	0.16	0.01	0.326	0.34	0.02	0.336	91.83	5.59	0.196
		NA	3600.000	0.05	0.01	0.328	0.41	0.02	0.342	1.39	0.05	0.273
Avg.				0.12	0.01	0.114	0.31	0.02	0.120	5.74	0.36	0.095

4. Computational results

We conducted computational experiments using randomly generated test problems to evaluate the performances of the GAs proposed in this article. Since the complexity of a problem is highly dependent on the number of jobs, two problem groups are randomly generated according to the number of jobs. A small sized problem group, involving problem instances with the number of machines being 2, 3, and 4 and the average number of jobs per machine being 2, 3, and 4, is for comparing solutions obtained by GAs with the optimal solution. A large sized problem group, which involves problem instances with the number of machines as 4, 6, 8, and 10 and the average number of jobs per machine as 4, 6, 8, and 10, is for comparing the relative performance of each GA.

To generate test problems for unrelated parallel machine scheduling, we first generate the maximum base time p_{max} as $\frac{H}{|SJ|/|SM|}$. The base processing time of job i, P_i, is randomly generated in the range of $[0.6 \times p_{max}, p_{max}]$, the base unloading time of job i, U_i, is randomly generated in the range of $[0.05 \times P_i, 0.1 \times P_i]$, the base loading time of job i, L_i, is randomly generated in the range of $[0.05 \times P_i, 0.15 \times P_i]$, and the base change time between jobs i and j, C_{ij}, is randomly generated in the range of $[0, 0.1 \times p_{max}]$. To generate machine-dependent processing times and sequence- and machine-dependent setup times with the base times, the machine adjusting factor of job i on machine k, δ_{ik}, is randomly generated in the range of $[0.5, 1.5]$ if the production availability u_{ik} equals 1. Hence, the machine-dependent processing times of job i on machine k, p_{ik}, is calculated as $P_i \times \delta_{ik}$, and the sequence and machine dependent setup time between jobs i and j on machine k, s_{ijk}, is calculated as $U_i \times \delta_{ik} + L_j \times \delta_{jk} + C_{ij}$. Note that since the setup time of the initial job on a machineinvolves only the loading time of the job, the initial setup time of job i on machine k, s_{iik}, is calculated as $L_i \times \delta_{ik}$.

ILOG CPLEX 10.2 is used to obtain the optimal solution through mathematical programming as presented in Section 2. A 3600s time limit is imposed and a particular run is simply terminated if

Table 2
Test results of large sized problems.

Jobs/MC	*MCs*	GA_DR_P				GA_DR_C			GA_DR_S		
		Best	*RPD* (%)	*MAD* (%)	Time (s)	*RPD* (%)	*MAD* (%)	Time (s)	*RPD* (%)	*MAD* (%)	Time (s)
4	4	1526	0.36	0.02	0.321	1.16	0.02	0.325	2.52	0.01	0.269
		1632	0.09	0.01	0.315	0.17	0.01	0.323	1.77	0.03	0.268
		1602	0.16	0.01	0.331	0.59	0.02	0.345	0.61	0.03	0.270
	6	2385	0.22	0.01	0.831	2.03	0.07	0.869	184.66	3.82	0.326
		2408	0.40	0.02	0.826	0.98	0.04	0.860	207.55	2.67	0.340
		1988	0.30	0.02	0.918	1.45	0.06	0.987	3.76	0.08	0.679
	8	2337	0.15	0.01	1.923	2.64	0.05	2.131	15.40	0.10	1.280
		2483	0.13	0.01	1.831	2.19	0.06	1.978	8.13	0.05	1.249
		2474	0.09	0.01	1.903	1.65	0.03	2.074	10.53	0.06	1.271
	10	3227	0.20	0.01	3.326	4.79	0.06	3.593	12.66	0.12	2.241
		2977	0.10	0.01	3.390	1.89	0.03	3.736	9.85	0.13	2.182
		3348	0.22	0.01	3.365	2.59	0.04	3.640	7.97	0.18	2.141
6	4	1290	0.27	0.01	0.739	1.40	0.05	0.780	8.07	0.01	0.599
		1419	0.16	0.01	0.746	0.98	0.04	0.791	5.50	0.04	0.611
		1512	0.15	0.01	0.744	0.84	0.02	0.780	29.11	3.90	0.564
	6	2188	0.16	0.01	2.015	1.80	0.04	2.135	37.02	3.76	1.277
		2078	0.39	0.02	2.032	4.62	0.07	2.178	4.37	0.09	1.424
		2061	0.23	0.01	2.056	2.35	0.04	2.195	9.86	0.06	1.535
	8	2528	0.33	0.01	4.321	4.32	0.10	4.651	80.87	6.08	2.543
		2503	0.19	0.01	4.365	2.80	0.05	4.759	13.04	0.16	3.090
		2685	0.43	0.03	4.332	4.13	0.05	4.731	74.34	5.84	2.744
	10	3187	0.21	0.02	7.638	4.84	0.08	8.347	10.50	0.09	5.232
		2939	0.09	0.00	8.147	3.72	0.07	8.993	16.68	0.07	5.412
		2970	0.23	0.01	7.715	3.83	0.05	8.470	17.39	0.17	5.201
8	4	1492	0.54	0.02	1.299	2.54	0.04	1.374	117.93	6.17	0.742
		1516	0.34	0.02	1.289	1.38	0.03	1.342	4.34	0.04	1.058
		1642	0.68	0.03	1.303	1.67	0.02	1.362	3.84	0.08	1.004
	6	1834	0.35	0.01	3.906	3.01	0.03	4.232	12.72	0.08	2.883
		2115	0.67	0.03	3.604	4.71	0.09	3.898	218.70	3.94	1.687
		2136	0.24	0.02	3.685	3.69	0.08	3.955	304.49	0.00	1.245
	8	2669	0.42	0.01	7.979	4.18	0.05	8.694	12.90	0.20	5.623
		2555	0.16	0.01	8.249	3.62	0.06	8.978	16.82	0.16	5.314
		2476	0.36	0.02	8.210	6.18	0.08	8.900	20.27	0.10	5.649
	10	2929	0.18	0.01	14.760	5.88	0.09	16.184	23.53	0.11	10.097
		2861	0.16	0.01	14.524	5.25	0.05	15.926	22.03	0.15	10.019
		3045	0.13	0.01	14.624	5.37	0.06	16.133	20.00	0.12	10.053
10	4	1395	0.27	0.01	2.329	2.04	0.07	2.442	5.79	0.07	1.803
		1539	0.36	0.02	2.214	2.52	0.03	2.341	59.10	5.41	1.531
		1572	0.35	0.02	2.185	2.73	0.07	2.295	240.18	1.39	0.867
	6	2190	0.35	0.02	5.931	4.81	0.06	6.302	209.33	3.86	2.774
		2038	0.22	0.01	6.086	2.35	0.05	6.522	8.62	0.06	4.582
		1954	0.28	0.02	6.071	3.59	0.03	6.571	12.86	0.10	4.606
	8	2472	0.33	0.02	12.830	5.74	0.06	13.763	15.85	0.20	9.135
		2579	0.22	0.01	13.116	4.66	0.04	14.227	17.15	0.17	9.197
		2618	0.37	0.02	12.712	5.88	0.09	13.731	50.14	3.86	8.446
	10	2940	0.28	0.02	24.047	6.49	0.09	26.205	22.14	0.08	16.417
		2915	0.32	0.01	24.015	6.92	0.07	26.013	23.33	0.09	16.646
		2979	0.24	0.01	23.636	7.11	0.06	25.638	23.65	0.15	16.541
Avg.			0.27	0.01	5.890	3.33	0.05	6.390	46.62	1.13	3.972

the optimal solution is not found and verified in the given amount of time. All the GAs are run with the population size of $2 \times$ (number of jobs) and the maximum generation size of 1000. The crossover P_C and mutation rates P_M for the GAs are 0.8 and 0.2, which are predetermined by extensive preliminary experimentations. All experiments utilize CPLEX and GAs are executed on a PC with 1.86 GHz Intel Core 2 processor and 2 GB RAM. Three replications in each instance are tested in both groups. To compare the performances of GAs, the relative percent deviation (*RPD*) is calculated using expression (13).

$$RPD(\%) = \frac{Best - GA_{Sol}}{Best} \times 100, \tag{13}$$

where GA_{Sol} is a solution obtained by GAs and *Best* is the best solution of all experiments for each test problem. *Best* can be the optimal solution if CPLEX obtains the optimal solution.

The test results for the problems in the small sized problem group are summarized in Table 1. The optimal solution by CPLEX and the average *RPD* and mean absolute deviation (*MAD*) of ten replications by GA_DR_P, GA_DR_C, and GA_DR_S for all test

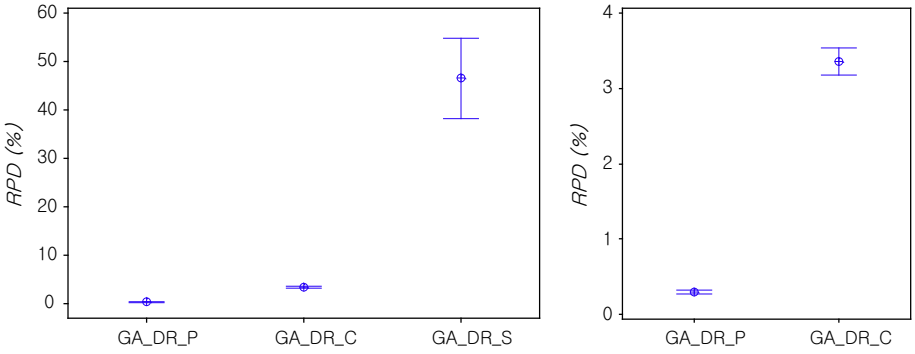
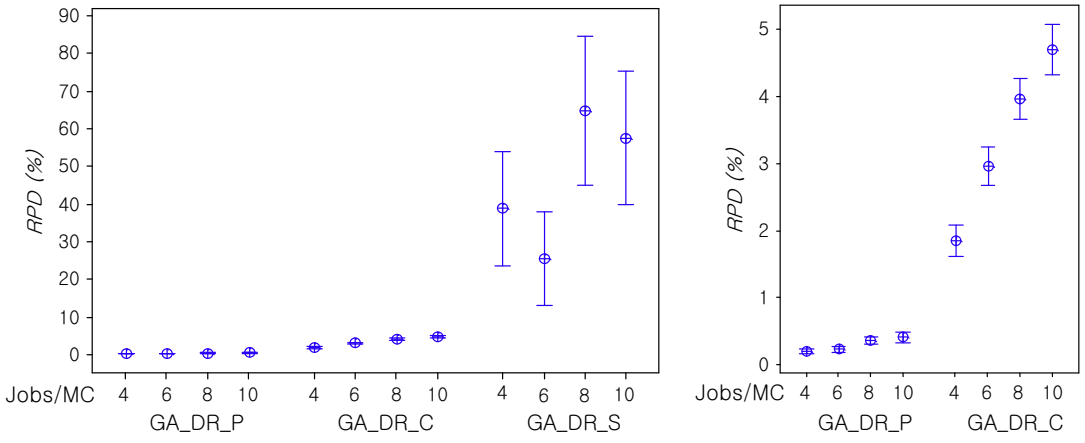
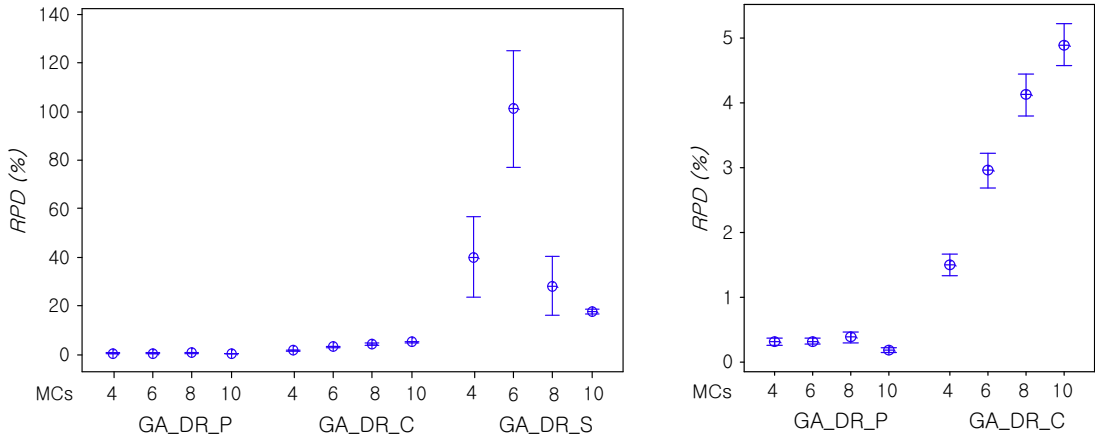


Fig. 3. Mean plots and Tukey HSD intervals at the 95% confidence level of GA_DR_P, GA_DR_C, and GA_DR_S.



(a) Groups of job size (*Jobs/MC*)



(b) Groups of machine size (*MCs*)

Fig. 4. Mean plots and Tukey HSD intervals at the 95% confidence level for GA_DR_P, GA_DR_C, and GA_DR_S of different job size (*Jobs/MC*) and machine size (*MCs*).

problems are compared. In the table, *Jobs/MC* means the average number of jobs per machine and *MCs* means the number of machines. The computational time of CPLEX for the small-sized problems in the small-sized problem group significantly increases as the number of jobs increases. In the test, CPLEX was not able to obtain the optimal solution for problems over an average four jobs on four machines within a 3600s time limit. However, GAs are able to provide near-optimal solutions within a second.

The average *RPD* and *MAD* of ten replications by GA_DR_P, GA_DR_C, and GA_DR_S of problems for the problems in the large-sized problem group are summarized in Table 2. In the test, GA_DR_P provided the best performance with the low values of *RPD* and *MAD*. In order to validate the results, we executed statistical significance tests of the observed differences in the *RPD* values of each implemented algorithm. Fig. 3 shows the mean plots and Tukey HSD intervals at the 95% confidence level of all problems in Table 2. Fig. 3 indicates that there are statistically significant differences between the *RPD* values of all the implemented algorithms. The observed differences between dispatching rules are more statistically significant as the number of jobs per machine and the number of machines increase, as shown in graph (a) and (b) in Fig. 4. Because GA_DR_P and GA_DR_C offer much lower RPD with small variance than GA_DR_S, it is difficult to detect the difference between GA_DR_P and GA_DR_C. Hence, we eliminate GA_DR_S from all the plots and redraw the plots with only GA_DR_P and GA_DR_C for detecting the difference between them. From the redrawn plots, we find that the *RPD* in GA_DR_P is significantly different from GA_DR_C, as *Jobs/MC* size and *MCs* size increase. These results are due to the fact that the processing-time-based dispatching rule of GA_DR operates as a good local search method for excluding bad solutions.

Computational times of GA_DR_P, GA_DR_C, and GA_DR_S for problems in the large-sized problem group are summarized in Table 2. The computation times of the proposed algorithms are small enough to obtain solutions within a reasonable time frame.

5. Conclusions

In this article, the unrelated parallel machine scheduling problem with sequence- and machine-dependent setup times, machine-dependent processing times, and production availability constraints was considered. The objective of our article is to find a schedule that minimizes the total completion time while simultaneously determining the assignment of jobs to available machines and the sequencing of assigned jobs on each machine. A MIP model is derived to search for the optimal solution. While CPLEX can be used for solving models of small-sized problems, it is inefficient and impractical for solving large-sized problems owing to the increased computation time requirement. In this artlcle, we propose hybrid GAs with three dispatching rules, GA using chromosomes with processing-time-based dispatching rule (GA_DR_P), GA using chromosomes with completion-time-based dispatching rule (GA_DR_C), and GA using chromosomes with sequence-based dispatching rule (GA_DR_S). The test results conclude that among all the implemented algorithms for the unrelated parallel machine scheduling problems, GA_DR_P provides the highest quality performance, in both effectiveness and efficiency.

Acknowledgment

This work was supported by the Incheon National University Research Grant in 2014 (Grant No.: 20141205).

References

Agarwal, A., Colak, C., Jacob, V., & Pirkul, H. (2006). Heuristics and augmented neural networks for task scheduling with non-identical machines. *European Journal of Operational Research, 175*(1), 296–317.
Balin, S. (2011). Non-identical parallel machine scheduling using genetic algorithm. *Expert System with Applications, 38*, 6814–6821.
Chen, C. L., & Chen, C. L. (2009). Hybrid metaheuristics for unrelated parallel machine scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 43*(1), 161–169.
Cheng, R., & Gen, M. (1997). Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering, 33*(3–4), 761–764.
Gharehgozli, A. H., Tavakkoli-Moghaddam, R., & Zaerpour, N. (2009). A fuzzy-mixed-integer goal programming mode for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. *Robotics and Computer-Integrated Manufacturing, 25*, 853–859.
Hop, N. V., & Nagarur, N. N. (2004). The scheduling problem of PCBs for multiple non-identical parallel machines. *European Journal of Operational Research, 158*, 577–594.
Kim, D-W., Kim, K-H., Jang, W., & Chen, F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing, 18*, 223–231.
Ko, H., Kim, J., Kim, S., & Baek, J. (2010). Dispatching rule for non-identical parallel machines with sequence-dependent setups and quality restrictions. *Computers & Industrial Engineering, 59*, 448–457.
Lee, K., Kim, B. S., & Joo, C. M. (2012). Genetic algorithms for door-assignment and sequencing of trucks at distribution centers for the improvement of operational performance. *Expert System with Applications, 39*, 12975–12983.
Li, K., & Yang, S. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied Mathematical Modelling, 33*, 2145–2158.
Lin, C. W., Lin, Y. K., & Hsieh, H. T. (2013). Ant colony optimization for unrelated parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology, 67*, 35–45.
Lin, Y. K., & Lin, C. W. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *The International Journal of Advanced Manufacturing Technology, 67*, 269–279.
Lin, Y. K., Pfund, M. E., & Fowler, J. W. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problem. *Computers & Operations Research, 38*(6), 901–916.
Liu, M., & Wu, C. (2003). Scheduling algorithm based on evolutionary computing in identical parallel machine production line. *Robotics and Computer-Integrated Manufacturing, 19*, 401–407.
Omara, F. A., & Arafa, M. M. (2010). Genetic algorithms for task scheduling problem. *Journal of Parallel Distributed Computing, 70*, 13–22.
Rodriguez, F. J., Lozano, M., Blum, C., & Garcia-Martinez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machine scheduling problem. *Computers & Operations Research, 40*, 1829–1841.
Rogendran, R., & Subur, F. (2004). Unrelated parallel machine scheduling with job splitting. *IIE Transaction, 36*, 356–372.
Ruiz, R., & Andres-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology, 57*, 777–794.
Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., & Sassani, F. (2009). Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. *Computers & Operations Research, 36*, 3224–3230.
Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research, 211*, 612–622.
Vredeveld, T., & Hurkens, C. (2002). Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *Informs Journal on Computing, 14*(2), 175–189.
Webster, S. T. (1997). The complexity of scheduling job families about a common due date. *Operations Research Letters, 20*(2), 65–74.
Weng, M., Lu, J., & Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International Journal of Production Economics, 70*(3), 215–226.
Yang, D.-L., & Yang, S-J. (2013). Unrelated parallel-machine scheduling with multiple rate-modifying activities. *Information Sciences, 235*, 280–286.
Yang, D.-L., Cheng, T. C. E., & Yang, S.-J. (2013). Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimize total cost involving total completion time and job compressions. *International Journal of Production Research, 52*(4), 1133–1141.
Yang, S-J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling, 37*, 2995–3005.
Yang, S.-J., Hsu, C-J., & Yang, D-L. (2014). Note on Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time. *Information Sciences, 260*, 215–217.