PERGAMON

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 30 (2003) 1777–1789

computers & operations research

www.elsevier.com/locate/dsw

Scheduling unrelated parallel machines to minimize total weighted tardiness

Ching-Fang Liaw*, Yang-Kuei Lin, Chun-Yuan Cheng, Mingchin Chen

Department of Industrial Engineering and Management, Chaoyang University of Technology, Taichung, Taiwan

Received 1 April 2001; received in revised form 1 September 2001

Abstract

This article considers the problem of scheduling a given set of independent jobs on unrelated parallel machines to minimize the total weighted tardiness. The problem is known to be NP-hard in the strong sense. Efficient lower and upper bounds are developed. The lower bound is based on the solution of an assignment problem, while the upper bound is obtained by a two-phase heuristic. A branch-and-bound algorithm that incorporates various dominance rules is presented. Computational experiments are conducted to demonstrate the performance of the proposed algorithm.

Scope and purpose

Parallel machine scheduling models are important from both the theoretical and practical points of view. From the theoretical point of view, they generalize the single machine scheduling models. From the practical point of view, they are important because the occurrence of a bank of machines in parallel is common in industries. In this article, the unrelated parallel machine total weighted tardiness scheduling problem is examined. The tardiness criterion has many applications in real world. This problem is difficult to solve. A branch-and-bound algorithm that incorporates various dominance rules along with efficient lower and upper bounds is proposed to find an optimal solution.

Keywords: Scheduling; Parallel machines; Branch-and-bound; Tardiness

1. Introduction

We consider the problem of scheduling n independent jobs on m unrelated parallel machines to minimize the total weighted tardiness. We assume that each machine is continuously available and

* Corresponding author. Tel.: +886-4-332-3000; fax: +886-4-374-2327.
E-mail address: cfliaw@mail.cyut.edu.tw (C.-F. Liaw).

0305-0548/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
PII: S0305-0548(02)00105-3

can process at most one job at a time and each job can be processed by only one machine. No job preemptions are allowed. Each job i becomes available at time zero, has a processing time p_{ij} on machine j, a weight w_i, and a due date d_i. The objective is to determine a schedule so that the total weighted tardiness $\sum_{i=1}^{n} w_i T_i = \sum_{i=1}^{n} w_i \max(C_i - d_i, 0)$ is minimized, where C_i is the completion time of job i. Following the three-field notation of Graham et al. [1], we refer to this problem as $R_m \| \sum_{i=1}^{n} w_i T_i$. The problem is NP-hard in the strong sense since its special case with $m = 1$ is already NP-hard in the strong sense [2].

Bruno et al. [3], and Horn [4] have examined the $R_m \| \sum_{i=1}^{n} C_i$ problem, while Azizoglu and Kirca [5] have studied the $R_m \| \sum_{i=1}^{n} w_i C_i$ problem. Alidaee and Rosa [6], and Koulamas [7] have proposed heuristic algorithms that can be used to solve the $R_m \| \sum_{i=1}^{n} w_i T_i$ problem. To the best of our knowledge no exact algorithms have previously been published for the $R_m \| \sum_{i=1}^{n} w_i T_i$ problem. In this article, we develop a branch-and-bound algorithm that incorporates a lower bound scheme via assignment solutions and various dominance rules. The rest of the article is organized as follows. In Section 2, the properties of an optimal schedule are introduced. A lower bound strategy is presented in Section 3. In Section 4 we describe some dominance rules. All of these elements are incorporated into a branch-and-bound algorithm given in Section 5. Computational results are reported in Section 6 followed by our conclusions in Section 7.

2. Properties of an optimal schedule

In this section, we present some properties of an optimal schedule. The first property is due to Azizoglu and Kirca [5], and the others generalize the results given by Azizoglu and Kirca [8].

Property 1. There exists an optimal schedule in which the sum of the processing times of the jobs processed on machine k, $k = 1, 2, \ldots, m$, does not exceed

$$\frac{1}{m}\left\{\sum_{i=1}^{n} \max_{1 \leqslant j \leqslant m} p_{ij} + \sum_{\substack{j=1 \\ j \neq k}}^{m} \max_{1 \leqslant i \leqslant n} p_{ij}\right\}.$$

Property 2. There exists an optimal schedule in which job i is processed last on any one of the machines if

$$d_i \geqslant \frac{1}{m}\left\{\sum_{i=1}^{n} \max_{1 \leqslant j \leqslant m} p_{ij} + \max_{1 \leqslant k \leqslant m} \sum_{\substack{j=1 \\ j \neq k}}^{m} \max_{1 \leqslant i \leqslant n} p_{ij}\right\}.$$

Proof. By Property 1, job i is never tardy in any position on any machine. Hence, it can be processed last on any machine. □

Property 3. There exists an optimal schedule in which job i completes no latter than job k if $p_{ij} = p_{kj}$ for all $j = 1, 2, \ldots, m$, $w_i \geqslant w_k$, and $d_i \leqslant d_k$.

Proof. Suppose that we have an optimal schedule in which job i completes latter than job k and the conditions above hold. Let C_i and C_k be the completion time of job i and k, respectively. Then, $C_i > C_k$. If we interchange jobs i and k, the decrease in the total weighted tardiness is

$$\Delta = w_i \max(C_i - d_i, 0) + w_k \max(C_k - d_k, 0) - w_i \max(C_k - d_i, 0) - w_k \max(C_i - d_k, 0).$$

Two cases need to be considered.

Case 1: $C_k - d_i > 0$

$$\begin{aligned}
\Delta &= w_i(C_i - d_i) + w_k \max(C_k - d_k, 0) - w_i(C_k - d_i) - w_k \max(C_i - d_k, 0) \\
&= w_i(C_i - C_k) + w_k[\max(C_k - d_k, 0) - \max(C_i - d_k, 0)] \\
&\geqslant w_k[C_i - \max(C_i - d_k, 0) + \max(C_k - d_k, 0) - C_k] \\
&= w_k[C_i + \min(d_k - C_i, 0) - \min(d_k - C_k, 0) - C_k] \\
&= w_k[\min(d_k, C_i) - \min(d_k, C_k)] \\
&\geqslant 0 \text{ since } C_i \geqslant C_k.
\end{aligned}$$

Case 2: $C_k - d_i \leqslant 0$

$$\begin{aligned}
\Delta &= w_i \max(C_i - d_i, 0) - w_k \max(C_i - d_k, 0) \\
&\geqslant 0 \text{ since } w_i \geqslant w_k \text{ and } d_i \leqslant d_k.
\end{aligned}$$

In both cases, $\Delta \geqslant 0$. This implies that the interchange of jobs i and k cannot increase the total weighted tardiness so there exists another schedule which is also optimal and satisfies the conditions of the property. □

Property 4. If $d_i \leqslant \min_{1 \leqslant j \leqslant m} p_{ij}$ for all job i, then there exists an optimal schedule in which each machine processes jobs in the weighted shortest processing time (WSPT) sequence.

Proof. Since $d_i \leqslant \min_{1 \leqslant j \leqslant m} p_{ij}$ implies that $d_i \leqslant C_i$ for all job i, we have

$$\sum_{i=1}^{n} w_i T_i = \sum_{i=1}^{n} w_i \max(C_i - d_i, 0) = \sum_{i=1}^{n} w_i C_i - \sum_{i=1}^{n} w_i d_i.$$

Thus, jobs on each machine must be processed in WSPT sequence in order to minimize total weighted completion time. □

3. Lower bound

In this section, we present a lower bound based on solving an assignment problem. This lower bounding scheme was originally proposed in [5] for the $R_m \| \sum_{i=1}^{n} w_i C_i$ problem. Here, we generalize

with slight modification this scheme to the $R_m \| \sum_{i=1}^{n} w_i T_i$ problem. The $R_m \| \sum_{i=1}^{n} w_i T_i$ problem can be formulated as the following integer programming problem (P):

$$Z_p = \min \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{t=1}^{n} w_i T_{ik}^t x_{ik}^t \tag{1}$$

$$\text{s.t.} \sum_{k=1}^{m} \sum_{t=1}^{n} x_{ik}^t = 1, \quad i = 1,2,\ldots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ik}^t \leqslant 1, \quad k = 1,2,\ldots,m;\ t = 1,2,\ldots,n, \tag{3}$$

$$C_{ik}^t = \sum_{l=1}^{n} \sum_{s=1}^{t-1} p_{lk} x_{lk}^s, \quad i,t = 1,2,\ldots,n;\ k = 1,2,\ldots,m, \tag{4}$$

$$T_{ik}^t = \max(0, C_{ik}^t - d_i), \quad i = 1,2,\ldots,n, \tag{5}$$

$$x_{ik}^t = 0 \text{ or } 1, \quad i,t = 1,2,\ldots,n;\ k = 1,2,\ldots,m, \tag{6}$$

where C_{ik}^t denotes the completion time of job i when it is scheduled in the tth position on machine k, and

$$x_{ik}^t = \begin{cases} 1, & \text{if job } i \text{ is scheduled in the } t\text{th position on machine } k, \\ 0, & \text{otherwise.} \end{cases}$$

Constraint (2) states that each job is assigned to exactly one position on one machine, whereas constraint (3) guarantees the assignment of at most one job to each position on each machine. Constraint (4) gives the completion time of job i when it is scheduled in the tth position on machine k, and constraint (5) is the definition of tardiness.

The basic idea of this lower bounding scheme is to replace C_{ik}^t with some estimate $\hat{C}_{ik}^t$. Define $\hat{T}_{ik}^t = \max(\hat{C}_{ik}^t - d_i, 0)$. Then problem (P) can be approximated as an assignment problem (AP) as follows:

$$Z_{\text{Ap}} = \min \sum_{i=1}^{n} \sum_{k=1}^{m} \sum_{t=1}^{n} w_i \hat{T}_{ik}^t x_{ik}^t, \tag{7}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{t=1}^{n} x_{ik}^t = 1, \quad i = 1,2,\ldots,n, \tag{8}$$

$$\sum_{i=1}^{n} x_{ik}^t \leqslant 1, \quad k = 1,2,\ldots,m;\ t = 1,2,\ldots,n, \tag{9}$$

$$x_{ik}^t = 0 \text{ or } 1, \quad i,t = 1,2,\ldots,n;\ k = 1,2,\ldots,m. \tag{10}$$

It can be seen easily that $Z_{\mathrm{Ap}} \leqslant Z_p$ if $\hat{C}^t_{ik} \leqslant C^t_{ik}$ for all i, k and t. The following definition of $\hat{C}^t_{ik}$ satisfies the requirement and hence the resulted Z_{Ap} provides a lower bound for Z_p.

$$\hat{C}^t_{ik} = \begin{cases} \sum_{l=1}^{t} p_{[l]k} & \text{if } S_{ik} \leqslant t, \\ \sum_{l=1}^{t-1} p_{[l]k} + p_{ik} & \text{if } S_{ik} > t. \end{cases}$$

where $p_{[l]k}$ is the processing time of the lth job when all of the n jobs are processed on machine k in the shortest processing time (SPT) order, and S_{ik} is the position of job i in this ordering. In this research the Hungarian method [9] is used to solve the assignment problem (AP).

Instead of solving an assignment problem at every node, we solve an assignment problem only at the root node and then use the corresponding dual variable values to find lower bounds at other nodes. Similar approaches have been used in [5,10]. For completeness, we now describe the details of this approach. The dual problem of AP is given by

$$Z_{\mathrm{Ap}} = \max \sum_{i=1}^{n} u_i + \sum_{t=1}^{n} \sum_{k=1}^{m} v_{tk}$$

$$\text{s.t.} \quad u_i + v_{tk} \leqslant w_i \hat{T}^t_{ik} \quad i,t = 1,2,\ldots,n; \; k = 1,2,\ldots,m,$$

where $\{u_i\}$ and $\{v_{tk}\}$ are the sets of dual variables corresponding to constraint set (8) and (9), respectively. Let $\{u^*_i\}$ and $\{v^*_{tk}\}$ be the optimal dual solution.

Define AP(π) as the assignment problem corresponding to partial schedule π whose optimal solution $Z_{\mathrm{Ap}(\pi)}$, is a lower bound on the total weighted tardiness of unscheduled jobs. Let $\hat{T}^t_{ik}(\pi)$ be the estimated tardiness of unscheduled job i if it is scheduled in the tth position on machine k, corresponding to partial schedule π. According to the definition of $\hat{T}^t_{ik}(\pi)$, we have $\hat{T}^t_{ik} \leqslant \hat{T}^t_{ik}(\pi)$. Hence, $u^*_i + v^*_{tk} \leqslant w_i \hat{T}^t_{ik} \leqslant w_i \hat{T}^t_{ik}(\pi)$. This implies that the optimal dual solution at the root node, i.e. $\{u^*_i\}$ and $\{v^*_{tk}\}$, is feasible to the dual problem corresponding to AP(π). Since the dual problem is a maximization problem, a feasible dual solution provides a lower bound for $Z_{\mathrm{Ap}(\pi)}$. Thus, a lower bound on the weighted tardiness of unscheduled jobs is given by

$$\sum_{i \in E(\pi)} u^*_i + \sum_{(t,k) \in F(\pi)} v^*_{tk} = Z_{\mathrm{Ap}} - \sum_{i \notin E(\pi)} u^*_i - \sum_{(t,k) \notin F(\pi)} v^*_{tk},$$

where $E(\pi)$ is the set of unscheduled jobs in π and $F(\pi)$ the set of unfilled positions in π.

Given a partial schedule π, if an unscheduled job l is added to the first available position on machine j, say position q, a simple lower bound on the weighted tardiness of all possible complete schedules will be

$$\mathrm{LB0} = \sum_{i \notin E(\pi)} w_i T_i(\pi) + w_l \max[PT_j(\pi) + p_{lj}, 0] + Z_{\mathrm{Ap}} - \sum_{i \notin E(\pi)} u^*_i - \sum_{(t,k) \notin F(\pi)} v^*_{tk} - u^*_l - v^*_{qj},$$

where $T_i(\pi)$ is the tardiness of job i in π and $PT_j(\pi)$ the sum of the processing times of the jobs assigned to machine j in π.

The lower bound LB0 can be further improved by consider the following. Given a partial schedule π, for each unscheduled job $i \in E(\pi)$, its earliest possible completion time is $\mathrm{EPC}_i = \min_{1 \leqslant j \leqslant m}$

$\{PT_j(\pi) + p_{ij}\}$. Hence, a lower bound on the weighted tardiness of all unscheduled jobs is $\sum_{i \in E(\pi)} w_i \max(0, \mathrm{EPC}_i - d_i)$. Combining this result with the lower bound LB0, we obtain our improved lower bound as

$$\mathrm{LB1} = \sum_{i \notin E(\pi)} w_i T_i(\pi) + w_l \max[PT_j(\pi) + p_{lj}, 0] + \max\left\{ Z_{\mathrm{Ap}} - \sum_{i \notin E(\pi)} u_i^* - \sum_{(t,k) \notin F(\pi)} v_{tk}^* - u_l^* - v_{qj}^*, \quad \sum_{i \in E(\pi)} w_i \max(0, \mathrm{EPC}_i - d_i) \right\}.$$

Our computational tests show that LB1 significantly outperforms the lower bound scheme LB0. The average improvement is 72% in terms of number of nodes generated, and 66% in terms of CPU time.

4. Dominance rules

In this section, we describe some dominance rules which can be used in the branch-and-bound algorithm given in Section 5 to eliminate unpromising partial schedules during the search process. Based on the properties of an optimal schedule given in Section 2, we can develop the following dominance rules:

Rule 1: Let π be a partial schedule in which job l is processed on machine k and completed at time CT_l. If $CT_l > 1/m\{\sum_{i=1}^{n} \max_{1 \leqslant j \leqslant m} p_{ij} + \sum_{\substack{j=1 \\ j \neq k}}^{m} \max_{1 \leqslant i \leqslant n} p_{ij}\}$, then partial schedule π cannot lead to an optimal schedule.

Rule 2: Let π be a partial schedule in which job l is not processed last on any one of the machines. If $d_l \geqslant 1/m\{\sum_{i=1}^{n} \max_{1 \leqslant j \leqslant m} p_{ij} + \max_{1 \leqslant k \leqslant m} \sum_{\substack{j=1 \\ j \neq k}}^{m} \max_{1 \leqslant i \leqslant n} p_{ij}\}$, then partial schedule π cannot lead to an optimal schedule.

Rule 3: Let π be a partial schedule in which job i completes latter than job k. If $p_{ij} = p_{kj}$ for all $j = 1, 2, \ldots, m$, $w_i \geqslant w_k$, and $d_i \leqslant d_k$, then partial schedule π cannot lead to an optimal schedule.

Rule 4: If $d_i \leqslant \min_{1 \leqslant j \leqslant m} p_{ij}$ for all job i, then an optimal schedule must have jobs processed on each machine in the WSPT order.

Emmons [11], Rachamadugu [12], and Akturk and Yildirim [13] have developed various dominance rules for the $1||\sum_{i=1}^{n} w_i T_i$ problem. All of these dominance rules can be used in our branch-and-bound algorithm since for the $R_m||\sum_{i=1}^{n} w_i T_i$ problem each machine itself is a $1||\sum_{i=1}^{n} w_i T_i$ problem. The single machine dominance rules consider the precedence relations between jobs on the same machine. We present in the following a dominance rule that concerns with the jobs on different machines.

Rule 5: Let π be a partial schedule in which job i is processed on machine k and completed at time CT_i, and job l is processed on machine $j, j \neq k$ and completed at time CT_l. If $p_{ik} \geqslant p_{lk}$, $p_{lj} \geqslant p_{ij}$, and $w_i \max(CT_i + d_i, 0) + w_l \max(CT_l - d_l, 0) > w_i \max(CT_l - p_{lj} + p_{ij} - d_i, 0) + w_l \max(CT_i - p_{ik} + p_{lk} - d_l, 0)$, then partial schedule π cannot lead to an optimal schedule.

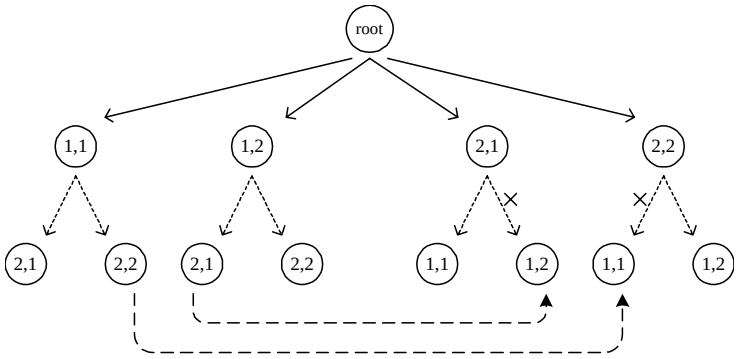

Fig. 1. Illustration of duplicated schedule elimination.

Proof. Let π' be the partial schedule obtained by interchanging jobs i and j in π. Define $CT_q(\pi)$ as the completion time of job q in partial schedule π. Since $p_{ik} \geqslant p_{lk}$ and $p_{lj} \geqslant p_{ij}$, we have $CT_q(\pi') \leqslant CT_q(\pi)$ for all $q \neq i, l$. This implies that the sum of total weighted tardiness of all jobs except jobs i and l in π' is less than or equal to that in π. Furthermore, the inequality $w_i \max(CT_i - d_i, 0) + w_l \max(CT_l - d_l, 0) > w_i \max(CT_l - p_{lj} + p_{ij} - d_i, 0) + w_l \max(CT_i - p_{ik} + p_{lk} - d_l, 0)$ implies that the sum of total weighted tardiness of jobs i and l in π' is less than that in π. Hence, the sum of total weighted tardiness of all jobs in π' is less than that in π, and partial schedule π cannot lead to an optimal schedule.

5. Branch-and-bound algorithm

In this section, we describe the implementation of our branch-and-bound algorithm.

Each node in the search tree corresponds to a partial schedule. A node in level r represents a partial schedule PS_r with r jobs scheduled. Given PS_r, we generate m partial schedules for each unscheduled job i, each consisting of PS_r plus the assignment of job i to the first available position on one machine. The number of complete schedules generated by this branching scheme is $m^n n!$. It can be seen easily that the branching strategy generates all possible schedules for the $R_m \| \sum_{i=1}^{n} w_i T_i$ problem. In fact, it can be further improved by eliminating the duplicated schedules generated during the branching process.

The duplicated schedule elimination scheme is based on the following idea. Let (i, j) denote the assignment of job i to the first available position on machine j. Suppose that there are two possible assignments (i, j) and (k, l) such that $i \neq k$ and $j \neq l$. We call such a pair of assignments *unrelated* since the completion times of jobs i and k remain unchanged if we reverse the order at which they are scheduled. If such a situation can be detected during the branching process, then it suffices to generate only one of the schedules since the second one is redundant. Consider the following example of $n = m = 2$ given in Fig. 1. Suppose that we are branching the root node. Then we have two pairs of unrelated assignments $\{(1,1),(2,2)\}$ and $\{(1,2),(2,1)\}$. For each unrelated pair of assignments, only one sequence of the assignments needs to be considered. As shown in Fig. 1, the marked branches can be discarded later during the branching process. The number of

complete schedules generated with this duplicated schedule elimination scheme is $C_{m-1}^{n+m-1}n!$, which is a significant reduction compared to $m^n n!$.

Initially, we apply the following heuristic (HEU) to obtain an upper bound (UB) for our branch-and-bound algorithm. The heuristic HEU is a parallel-machine extension of the apparent urgency (AU) rule proposed in [14]. The upper bound is updated whenever a complete schedule that improves the upper bound is generated during the branching process.

Step 1: Let U be the set of unscheduled jobs, and PT_j be the sum of processing times of the jobs that have already been scheduled on machine j, $j = 1, 2, \ldots, m$.

Initially, set $U = \{1, 2, \ldots, n\}$ and $PT_j = 0$, for $j = 1, 2, \ldots, m$.

Step 2: Determine the first available machine j^*, i.e., $PT_{j^*} = \min_{1 \leqslant j \leqslant m} PT_j$.

Step 3: Determine the unscheduled job i^* such that $I_{i^*j^*} = \max_{i \in U} I_{ij^*}$, where

$$I_{ij^*} = (w_i / p_{ij^*}) \exp[-(d_i - PT_{j^*} - p_{ij^*})^+ / K \bar{p}_{j^*}].$$

Step 4: Schedule job i^* in the first available position on machine j^* and update

$PT_{j^*} = PT_{j^*} + p_{i^*j^*}$. Set $U = \{U\} \setminus \{i^*\}$.

Step 5: Repeat steps 2–4 until all jobs are scheduled.

Step 6: For each machine j, $j = 1, 2, \ldots, m$, apply the adjacent pair-wise interchange procedure to improve the given schedule.

In step 3, K is a look-ahead parameter to be determined empirically, $\bar{p}_{j^*} = \sum_{i=1}^{n} p_{ij^*}/n$ is the average job processing time of machine j^*, and $(x)^+ = \max(0, x)$. HEU is a two-phase heuristic. In the first phase it constructs a complete schedule using a priority index (step 3), while in the second phase it improves the schedule obtained in the first phase by a local improvement procedure (step 6). More specifically, in step 6 we apply for each machine j, $j = 1, 2, \ldots, m$, the forward adjacent pair-wise interchange procedure $N_j - 1$ times, beginning with the jobs in the first two positions on machine j, where N_j is the number of jobs assigned to machine j. Two adjacent jobs are interchanged if it improves the total weighted tardiness of these two jobs. It can be seen easily that the complexity of heuristic HEU is $O(n^2)$.

We use the depth-first search strategy and break ties by selecting the node with the smallest lower bound. All of the dominance rules given in Section 4 are applied to eliminate unpromising nodes in the search tree. We calculate a lower bound for each node that cannot be eliminated. If the lower bound is greater than or equal to the current upper bound, the node is discarded.

6. Computational experiments

The algorithm is coded in C and implemented on a Pentium III-600 computer. We generate random test problems for various sizes. Processing times are randomly chosen from $U[1, 100]$. Due dates are generated from $U[\bar{P}(1 - T - R/2), \bar{P}(1 - T + R/2)]$ where $\bar{P} = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}/m^2$, T is the tardiness factor, set at 0.2, 0.6 and 1.0, and R is the relative range of due dates, set at 0.2, 0.6 and 1.0. The value of w_i for each i is chosen randomly from $U[1, 10]$. For each combination of n, m, T and R, 5 instances are generated, yielding 45 instances for each problem size.

Table 1
Performance of the heuristic HEU

		Solution quality		CPU time (s)	
T	R	MDD/HEU	KPM/HEU	MDD/HEU	KPM/HEU
0.2	0.2	*	1.00	1.92	0.96
	0.6	*	1.00	2.03	0.94
	1.0	*	1.00	2.11	0.99
0.6	0.2	*	*	2.18	0.91
	0.6	*	1.00	2.53	1.10
	1.0	2.47	1.30	2.96	1.10
1.0	0.2	1.42	1.08	5.54	1.63
	0.6	1.54	1.34	4.21	1.31
	1.0	1.30	1.30	4.04	1.28
Average		1.68	1.13	3.06	1.14

Table 2
Effectiveness of the initial lower bound (LB1) and upper bound (UB)[a]

	(UB/OPT) − 1			1 − (LB1/OPT)		
	m			m		
n	2	3	4	2	3	4
14	0.52	0.59	0.28	0.48	0.43	0.31
16	0.47	0.36	0.42	0.52	0.42	0.37
18	1.38	0.53	1.28	0.56	0.46	0.27
Average		0.65			0.42	

[a]The computation of the average ratio excludes the instances where the optimal solution value is zero.

To evaluate the performance of the proposed heuristic HEU, we compare HEU against two well known heuristics, i.e., the MDD rule by Alidaee and Rosa [6], and the KPM rule by Koulamus [7]. The problem size is set at $n = 500$ and $m = 10$. After solving a series of test problems, the parameter K is set to be 1.0. The results are given in Table 1. The symbol '*' denotes the cases for which the average ratio MDD/HEU or KPM/HEU takes the form $a/0$ where $a > 0$. Table 1 shows that HEU significantly outperforms both MDD and KPM in terms of solution quality and CPU time.

The average deviation from the optimal solution value of the initial lower and upper bounds are measured and reported in Table 2. The initial lower bound deviates from the optimal solution value by 42% on average, and the initial upper bound deviates from the optimal solution value by 65% on average.

To evaluate the effectiveness of the dominance rules presented in Section 4, we compare the performance of the branch-and-bound algorithm with and without the dominance rules. Table 3 summarizes the results for $n = 14$ and $m = 3$ in terms of number of nodes generated and CPU time. The symbol '—' denotes the cases for which the average ratio with D.R./without D.R. takes the

Table 3
Effectiveness of dominance rules

		No of nodes	CPU time (s)
T	R	$1-\frac{\text{with D.R.}}{\text{without D.R.}}$	$1-\frac{\text{with D.R.}}{\text{without D.R.}}$
0.2	0.2	—	—
	0.6	—	—
	1	0.95	0.85
0.6	0.2	0.97	0.94
	0.6	0.97	0.93
	1	0.97	0.94
1	0.2	0.9	0.79
	0.6	0.93	0.84
	1	0.91	0.8
Average		0.94	0.87

Table 4
Performance of the branch-and-bound algorithm

	No. of nodes			CPU time (s)		
	m			m		
n	2	3	4	2	3	4
14	1 912 580	2 959 970	3 675 089	9.27	30.81	30.82
16	6 928 761	28 250 214	47 481 557	58.48	305.22	460.31
18	6 428 1986	416 405 261	200 927 678	492.30	4057.09	2041.46

form 0/0. As shown in Table 3, when the dominance rules are used, the number of nodes generated and CPU time is reduced, respectively, by 94% and 87%. Hence, the proposed dominance rules are very effective in reducing the solution space.

Table 4 presents the computational results for the branch-and-bound algorithm. The results include the average number of nodes generated, and the average CPU time in seconds. Table 4 shows that both the average number of nodes generated, and the average CPU time increase rapidly as the problem size increases. The average CPU time for $n=14$ and $m=2$ is 9.27 s, while for $n=18$ and $m=4$ it rises to 2041.46 s.

Finally, the influence of the tardiness factor (T) and the due date range (R) on problem hardness is analyzed. The values of T and R are set at 0.2, 0.4, 0.6, 0.8 and 1.0. The problem size is set to be $n=16$ and $m=4$. Results for other problem sizes are similar. Table 5 shows the influence of T and R on the effectiveness of the initial lower bound LB1 and upper bound UB in terms of average deviation from the optimal solution value. The symbol '—' denotes the cases for which average ratio UB/OPT or LB1/OPT takes the form 0/0, i.e., both upper bound and optimal solution value equal to zero. We exclude the instances for which $\text{OPT}=0$ and $\text{UB}>0$ since in that case the ratio UB/OPT is meaningless. As shown in Table 5, when T is small ($\leqslant 0.4$), that is, due dates are loose,

Table 5
Influence of the tardiness factor (T) and the due date range (R) on LB1 and UB

	UB/Opt − 1					1 − LB1/Opt				
	R					R				
T	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
0.2	—	—	—	—	—	—	—	—	—	—
0.4	—	—	—	—	0.02	—	—	—	—	0.09
0.6	1.82	2.88	0.53	0.32	0.11	1.00	0.94	0.68	0.41	0.15
0.8	0.21	0.16	0.15	0.15	0.15	0.64	0.46	0.36	0.36	0.22
1	0.02	0.02	0.05	0.06	0.07	0.19	0.21	0.27	0.31	0.26

Table 6
Influence of the tardiness factor (T) and the due date range (R) on the performance of the branch-and-bound algorithm

	No. of nodes					CPU time (s)				
	R					R				
T	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1
0.2	1	1	1	1	1	0	0	0	0	0
0.4	1	1	1	1	15 892 600	0	0	0	0	169
0.6	96 823 465	103 222 094	67 254 882	51 274 909	20 420 633	1566	1006	648	500	197
0.8	118 663 134	94 387 291	83 223 653	56 180 186	32 479 610	1123	696	795	543	312
1.0	71 898 585	72 800 813	110 615 552	89 556 688	45 069 048	668	676	1054	841	422

optimal solution values are almost equal to zero. For the others values of T, the effectiveness of both LB1 and UB improves as T increases. Also, when $T \leqslant 0.8$, the effectiveness of both LB1 and UB improves as R increases, while when $T = 1.0$, the effectiveness of both LB1 and UB improves as R decreases.

Table 6 gives the influence of T and R on the performance of the branch-and-bound algorithm in terms of average number of nodes generated and average CPU time. It can be seen from Table 6 that when R is small ($\leqslant 0.4$) both the number of nodes generated and CPU time decrease as T increases, while when R is large both the number of nodes generated and CPU time increase as T increases. That is, when R is small ($\leqslant 0.4$) the larger the value of T the easier the problem becomes, while when R is large the larger the value of T the harder the problem becomes. However, it seems that the value of R does not have a significant influence on the problem hardness.

7. Conclusions

In this article, the problem of scheduling a given set of independent jobs on a set of unrelated parallel machines to minimize the total weighted tardiness is examined. A branch-and-bound

algorithm that incorporated efficient lower and upper bounds is proposed. Efficient dominance rules are also developed to help eliminating unpromising nodes in the search tree. Computational results show that the branch-and-bound algorithm performs well on problems with up to18 jobs and 4 machines.

The performance of the proposed branch-and-bound algorithm might be improved by using certain tighter but computationally slower lower and upper bounds. Also, additional research to develop stronger dominance rules to further cut down the size of the search tree may permit much larger sized problems to be solved.

References

[1] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 1979;5:287–326.
[2] Lenstra JK, Rinnooy Kan AHG, Brucker P. Complexity of machine scheduling problems. Annals of Discrete Mathematics 1977;1:343–62.
[3] Bruno LG, Coffman EG, Sethi R. Scheduling independent tasks to reduce mean finishing time. Communications of the ACM 1974;17:382–7.
[4] Horn WA. Minimizing average flow time with parallel machines. Operations Research 1973;21:846–7.
[5] Azizoglu M, Kirca O. Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. IIE Transactions 1999;31:153–9.
[6] Alidaee B, Rosa D. Scheduling parallel machines to minimize total weighted and unweighted tardiness. Computers and Operations Research 1997;24:775–88.
[7] Koulamas C. Decomposition and hybrid simulated annealing heuristics for the parallel-machine total tardiness problem. Naval Research Logistics 1997;44:109–25.
[8] Azizoglu M, Kirca O. Tardiness minimization on parallel machines. International Journal of Production Economics 1998;55:163–8.
[9] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 1955;1:83–97.
[10] Gelders L, Kleindorfer PR. Coordinating aggregate and detailed scheduling decisions in the one-machine job shop. Operations Research 1974;22:46–60.
[11] Emmons H. One-machine sequencing to minimize certain functions of job tardiness. Operations Research 1969; 17:701–15.
[12] Rachamadugu RMV. A note on weighted tardiness problem. Operations Research 1987;35:450–2.
[13] Akturk MS, Yildirim MB. A new dominance rule for the total weighted tardiness problem. Production Planning and Control 1999;10:138–49.
[14] Morton TE, Rachamadugu RM, Vepsalainen A. Accurate myopic heuristics for tardiness scheduling, GSIA Working Paper No. 36-83-84, Carnegie-Mellon University, PA, 1984.

Ching-Fang Liaw received B.B.A. and M.S. degrees in Industrial Management from National Cheng Kung University, Taiwan, and a Ph.D. degree in Industrial and Operation Engineering from the University of Michigan, MI. He is currently a Professor at the Chaoyang University of Technology, Taiwan. His research interests include combinatorial optimation and heuristic search methods with applications to production scheduling, and vehicle routing and scheduling.

Yang-Kuei Lin received B.S. and M.S. degrees in Industrial Engineering and Management from Chaoyang University of Technology, Taiwan. Her research interests include production planning and control, and system simulation.

Chun-Yuan Cheng received B.S. degree in Industrial Engineering from Chung-Yuan University, Taiwan, M.S. and Ph.D. degrees in Industrial Engineering from Auburn University, AI. She is currently an Associate Professor at Chaoyang University of Technology, Taiwan. Her research interests include input analysis in simulation, maintenance and reliability, and production scheduling.

Mingchih Chen received a B.S degree in Industrial Engineering from Chung Yuan Christian University, Taiwan, and M.S. and Ph.D. degrees in industrial Engineering from the Texas A&M University, Texas. She is currently an associate professor at the Chaoyang University of Technology, Taiwan. Her research interests include reliability, maintainability and production scheduling.