Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 195 (2009) 89–97
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Single machine scheduling to minimize total weighted earliness subject to minimal number of tardy jobs

Guohua Wan [a,b,*], Benjamin P.-C. Yen [c]

[a] Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200052, China
[b] College of Management, Shenzhen University, Shenzhen 518060, China
[c] School of Business, The University of Hong Kong, Pokfulam Road, Hong Kong

Received 16 May 2006; accepted 22 January 2008
Available online 1 February 2008

Abstract

Motivated by just-in-time manufacturing, we consider a single machine scheduling problem with dual criteria, i.e., the minimization of the total weighted earliness subject to minimum number of tardy jobs. We discuss several dominance properties of optimal solutions. We then develop a heuristic algorithm with time complexity $O(n^3)$ and a branch and bound algorithm to solve the problem. The computational experiments show that the heuristic algorithm is effective in terms of solution quality in many instances while the branch and bound algorithm is efficient for medium-size problems.

Keywords: Scheduling; Single machine; Dual criteria; Earliness; Number of tardy jobs

1. Introduction

Scheduling models with both earliness and tardiness costs are consistent with the philosophy of just-in-time (JIT) manufacturing, which emphasizes producing goods only when they are needed. In these scheduling models, jobs are scheduled to complete as close as possible to their due dates. A large body of literature on scheduling models with earliness and tardiness has appeared in the last two decades; see Baker and Scudder (1990) for an extensive review in 1990. However, most of the research focuses on JIT scheduling models with objective of minimizing total (weighted) costs of early and tardy jobs. For example, Sridharan and Zhou (1996) considered a single machine total weighted earliness and tardiness scheduling problem and developed a solution procedure based on decision theory; Cai and Zhou (1999) studied a parallel machine stochastic scheduling problem to minimize expected total cost for early and tardy jobs; Mazzini and Armentano (2001) studied a general single machine scheduling problem with early and tardy costs and developed a heuristic; Wan and Yen (2002) studied a general single machine scheduling problem with distinct due windows and weighted earliness and tardiness costs, and developed a tabu search procedure; Hino et al. (2005) developed a tabu search heuristic and a genetic algorithm to solve a single machine scheduling problem with a common due date to minimize total weighted earliness and tardiness costs.

On the other hand, only a few studies are concerned with JIT scheduling models with dual criteria. For single machine dual criteria scheduling models, Lee and Vairaktarakis (1993) provided a framework with classification and complexity results. Chen and Bulfin (1993) studied complexity of various single machine multi-criteria scheduling problems. Nagar et al. (1995) provided a detailed literature survey of multiple and bicriteria scheduling problems. Yen and Wan (2003) presented an extensive review of single machine bicriteria scheduling models, especially models with costs of early and tardy jobs. Vairaktarakis and Lee

* Corresponding author. Address: Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200052, China. Tel.: +86 21 52301045; fax: +86 21 62932577.
E-mail address: ghwan@sjtu.edu.cn (G. Wan).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.01.029

(1995) and Duffuaa et al. (1997) studied a single machine scheduling problem to minimize total tardiness subject to minimal number of tardy jobs, independently. They derived several dominance properties of optimal solutions, and developed heuristics as well as branch and bound algorithms that take advantage of the properties to solve the problem. For bicriteria scheduling models related to early and tardy costs, Chen et al. (1997) considered a single machine scheduling problem of minimizing total weighted earliness subject to maximum tardiness. They developed a heuristic and branch and bound algorithms based on the properties they derived to solve the problem. Chand and Schneeberger (1988) studied a single machine problem to minimize total weighted earliness subject to no tardy jobs. They transformed the problem into a scheduling problem of minimizing total weighted completion time, and developed heuristics and a dynamic programming procedure for solving the problem. However, their dynamic programming procedure is not efficient for problems of reasonable size as noted by the authors. Guner et al. (1998) considered one machine scheduling to minimize the maximum earliness with minimum number of tardy jobs. They proposed a procedure to minimize the maximum earliness when the set of tardy jobs is specified, and a branch and bound algorithm to minimize the maximum earliness subject to minimum number of tardy jobs. Karasakal and Koksalan (2000) developed a simulated annealing approach to two single machine bicriteria scheduling problems: one to minimize total flowtime and maximum earliness while the other to minimize total flowtime and number of tardy jobs. Later, Koksalan and Keha (2003) also developed genetic algorithms for the two problems.

Recently, Azizoglu et al. (2003a) considered a single machine scheduling problem with maximum earliness and number of tardy and no inserted idle time. They developed procedures to solve the problem optimally. Azizoglu et al. (2003b) further studied the same problem, but with allowed idle time. They gave polynomial time algorithms for problems of minimizing the number of tardy jobs subject to minimum maximum earliness, and of minimizing the maximum earliness subject to no tardy jobs and a given set of tardy jobs, respectively. Pathumnakul and Egbelu (2005) studied the same problem as in Chand and Schneeberger (1988) and developed a heuristic algorithm based on local optimality conditions. They showed numerically that their heuristic algorithm outperforms the methods by Chand and Schneeberger (1988).

In this paper, we consider a single machine bicriteria scheduling problem of minimizing the total weighted earliness subject to minimum number of tardy jobs, which is an extension of the problem studied in Chand and Schneeberger (1988) and Pathumnakul and Egbelu (2005). There is a couple of reasons for considering such dual criteria. In many practical situations, it is required to guarantee that as many jobs as possible to meet their due dates (i.e., to minimize the number of tardy jobs) since in such cases, having a job missing its due date is very costly. Thus minimization of the number of tardy jobs should be the primary concern. On the other hand, it is desirable to minimize the job earliness so as to minimize the inventory cost. A typical example is fashion manufacturing. In fashion manufacturing, if a job is overdue (on the basis of fashion season), then it misses the sale opportunity regardless of how overdue it is, which is very costly. Hence the minimization of the number of tardy jobs is the primary concern. Given this constraint, it is desirable to minimize the total weighted earliness so as to minimize the inventory cost.

The remainder of this paper is organized as follows. In Section 2, we give some notations and discuss several basic properties of optimal solutions of the problem. Section 3 is devoted to the development of algorithms for solving the problem, including a heuristic algorithm and a branch and bound algorithm. These algorithms are based on the properties developed in Section 2. Section 4 presents the computational results for the heuristic algorithm as well as the branch and bound algorithm. Section 5 concludes the paper.

2. Notations and dominance properties

Consider a scheduling problem with n jobs to be processed on one machine with the following assumptions:

(1) all the jobs are available at time zero;
(2) the machine can process at most one job at a time;
(3) no preemption is allowed; and,
(4) associated with each job j ($j = 1,2,\ldots,n$) there are a processing time p_j and a due date d_j.

The objective of the problem is to find a schedule that minimizes the total weighted earliness subject to the minimization of the number of tardy jobs. We denote this dual criteria as $\sum_j \alpha_j E_j | \sum_j U_j = U_{\min}$, where α_j is the earliness weight of job j ($j = 1,2,\ldots,n$), and $U_{\min}$ is the minimal number of tardy jobs.

Given a schedule π of the problem, we define the following notations.

$C_j(\pi)$ completion time of job j in schedule π;
$s_j(\pi)$ starting time of job j in schedule π;
$U_j(\pi)$ $\begin{cases} 1, & \text{if job } j \text{ is tardy,} \\ 0, & \text{otherwise.} \end{cases}$
$E_j(\pi)$ $\max\{0, d_j - C_j(\pi)\}$: earliness of job j in schedule π.

In cases where the schedule π is clearly known, we may not explicitly express the schedule π in the notations.

Based on the above notations, we have the following mathematical formulation for the scheduling problem:

$$\text{Min} \quad \sum_j \alpha_j E_j(\pi),$$

$$\text{Subject to:} \quad \sum_j U_j(\pi) = U_{\min} \quad \text{and} \quad \pi \in \Pi,$$

where Π denotes the set of all permutations of $\{1,2,\ldots,n\}$. We denote this problem as (**P**). From Chand and Schnee-

berger (1988), we know that problem (**P**) is strongly NP-hard.

To facilitate presentation, we give the algorithm for finding the minimum number of tardy jobs below (see Sidney, 1973).

Moore–Hodgson algorithm

Input: a job set J.
Output: the minimum number of tardy jobs for job set J.
Algorithm:

Step 0. Reindex the jobs in non-decreasing order of their due dates.
Step 1. $\sigma \leftarrow \phi, \sigma' \leftarrow \phi$,
Step 2. If $d_{j^*} = \min_{j\in J}\{d_j\}, \sigma \leftarrow \sigma \cup \{j^*\}, J \leftarrow J - \{j^*\}$.
Step 3. If $\sum_{j\in\sigma} p_j > d_{j^*}$, let k^* denote the job satisfying $p_{k^*} = \max_{j\in\sigma}\{p_j\}$ (break ties with large due date), then $\sigma' \leftarrow \sigma' \cup \{k^*\}$, $\sigma \leftarrow \sigma - \{k^*\}$.
Step 4. If $J = \phi$, stop; otherwise go to Step 2.

Remark 1. In this algorithm, σ and σ' denote the non-tardy job set and tardy job set, respectively. In the sequel, we assume the jobs in $\sigma = \{1,\ldots,|\sigma|\}$ are in non-decreasing order of their due dates and denoted as $\sigma = (1,\ldots,|\sigma|)$, where $|\sigma|$ denotes the cardinality of set σ, i.e., the number of non-tardy jobs.

Now we develop several properties of optimal solutions to facilitate solving problem (**P**). By Moore–Hodgson algorithm, the minimum number of tardy jobs can be easily identified. However, the number of choices of tardy jobs from the whole job set increases exponentially with respect to the total number of jobs, even if the number of tardy jobs is known in advance. Thus problem (**P**) is hard to solve. We first consider conditions for interchanges between non-tardy jobs. We then develop a condition for interchanges between a tardy job and a non-tardy job. Based on these conditions, we present a heuristic algorithm for solving problem (**P**).

Proposition 1. *In an optimal schedule, the completion time of the kth non-tardy job in the sequence* $\sigma = (1,\ldots,|\sigma|)$ *is equal to* $\min\{d_k, C_{k+1} - p_{k+1}\}$, $k = 1,\ldots,|\sigma| - 1$. *Especially, the completion time of the last non-tardy job is its due date.*

Proof. Note that only the non-tardy jobs are considered. It is easy to see that a non-tardy job should be completed at its latest possible time point, i.e., $\min\{d_k, C_{k+1} - p_{k+1}\}$. □

Remark 2. By this proposition, non-tardy jobs may be scheduled one by one backwards. This proposition forms the basis of the heuristic as well as the branch and bound algorithm.

Definition 1. For the *non-tardy* jobs in a given schedule, a series of consecutively processed jobs is called a *block*, i.e., jobs within a block are non-tardy and processed without inserted idle time between any two jobs.

Proposition 2. *In an optimal schedule, the last job in a block should be completed at its due date.*

Proof. It is easy to see that if the last job is completed after its due date, then it is overdue. On the other hand, if it is completed before its due date, then this job can be shifted to the right: (1) to reach its due date, or (2) to reach the starting time of the next job and join another block as the first job. Obviously, the earliness cost for this job decreases. Hence, the total cost decreases as well. □

Proposition 3. *If two adjacent jobs i and j within the same block satisfy the following condition*:

$$s \leqslant \min\{d_i, d_j\} \quad \text{and} \quad p_i/\alpha_i > p_j/\alpha_j, \tag{1}$$

where s is the starting time of the job immediately following these two jobs, then there exists an optimal schedule in which job i precedes job j.

Proof. It is a generalization of the weighted largest processing time rule (WLPT) since here the objective is to minimize the weighted earliness. Note that both jobs are non-tardy, by the interchanging argument, it is easy to see that the proposition holds. □

Proposition 4. *Assume the last two adjacent jobs i and j within a block B satisfy the following two inequalities*:

$$d_i < d'_j \tag{2}$$

and

$$\alpha_i * (p_j - R) < \alpha_j * (p_i + d'_j - d_i) + \sum_{l\in B-\{i,j\}} \alpha_l * \min\{(\min\{d_{l-1}, s'_l\} - s_l), R\}, \tag{3}$$

where $d'_j = \min\{d_j, s\}$ *and s is the starting time of the first job in the block immediately following block B*; $R = \min\{p_j, (d'_j - d_i)\}$ *(denoting the maximum possibly reduced time for jobs in block B due to putting job i before job j); and* s'_l, s_l *are the new and the original starting times of the job immediately following job* $l-1$ *in block B, respectively.*

Then there exists an optimal schedule in which job i precedes job j.

Proof (*see Fig. 1*). Suppose originally job j precedes job i. Now interchange job i and job j. By inequality (2), it is easy to see that the jobs in block B (except job j) may gain an opportunity to shift to the right in time, thus it may potentially reduce the earliness cost.

Note that $\alpha_i * (p_j - R)$ is the cost of such an interchange, and $\alpha_j * (p_i + d'_j - d_i) + \sum_{l\in B-\{i,j\}} \alpha_l * \min\{(\min\{d_{l-1}, s'_l\} - s_l), R\}$ is the benefit due to such an interchange. By inequality (3), the total cost is less than the benefit. Therefore, job i should precede job j in an optimal schedule. □

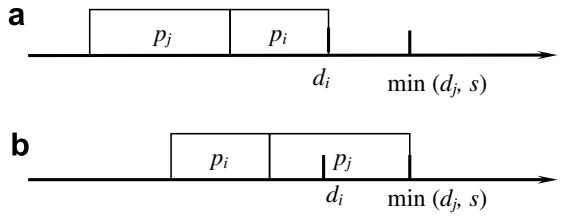

Fig. 1. The last two adjacent jobs in a block: (a) job j precedes job i, (b) job i precedes job j.

Remark 3. Propositions 3 and 4 specify two sufficient conditions on how to arrange two adjacent jobs within a block, for the non-last two jobs and the last two jobs, respectively, in an optimal schedule. They are the generalization of the WLPT rule.

Now we consider a condition for interchanging a tardy job and a non-tardy job.

Proposition 5. *Let σ and σ' be the non-tardy and the tardy job sets, respectively (identified by Moore–Hodgson algorithm), in a schedule. Assume job $i \in \sigma$ is a non-tardy job in a block B, and $j \in \sigma'$. If conditions in any of the following cases holds, then there exists an optimal schedule in which job j is scheduled in the original position of job i in block B, and job i is scheduled as a tardy job.*

Case 1:

$$d_i > \min\{d_j, s\} \quad \textit{and} \quad p_i > p_j \tag{4}$$

and

$$\alpha_i * (d_i - \min\{d_j, s\}) + \sum_{l \in B'} \alpha_l * (\min\{d_{l-1}, s'_l\} - s_l) > \alpha_j * (d_j - \min\{d_j, s\}) \tag{5}$$

or

Case 2:

$$d_i \leqslant \min\{d_j, s\} \quad \textit{and} \quad p_i > p_j - (\min\{d_j, s\} - d_i) \tag{4'}$$

and

$$\sum_{l \in B'} \alpha_l * (\min\{d_{l-1}, s'_l\} - s_l) > \alpha_j * (d_j - \min\{d_j, s\}), \tag{5'}$$

where s is the starting time of the job immediately following job i; s'_l, s_l denote, respectively, the new and the original starting times of job l in block B; and B' is a new block containing jobs scheduled before job i in the original block B.

Proof (*See Fig. 2*).

Case 1: By inequality (4), interchanging job i and job j does not delay any job after job i in the original schedule while it also does not push the job(s) before job i in the original schedule processed earlier.Now that $\alpha_i * (d_i - \min\{d_j, s\}) + \sum_{l \in B'} \alpha_l * (\min\{d_{l-1}, s'_l\} - s_l)$ is the cost before job interchange, and $\alpha_j{}^* (d_j - \min\{d_j, s\})$ is the cost after job interchange, thus by inequality (5), interchanging the two jobs may decrease the total cost because the savings are greater than the cost. Therefore, in an optimal schedule, job j should be scheduled in the original position of job i in block B, and job i is scheduled as a tardy job.

Case 2: Similar to the proof in Case 1.

By Cases 1 and 2, the proposition holds. □

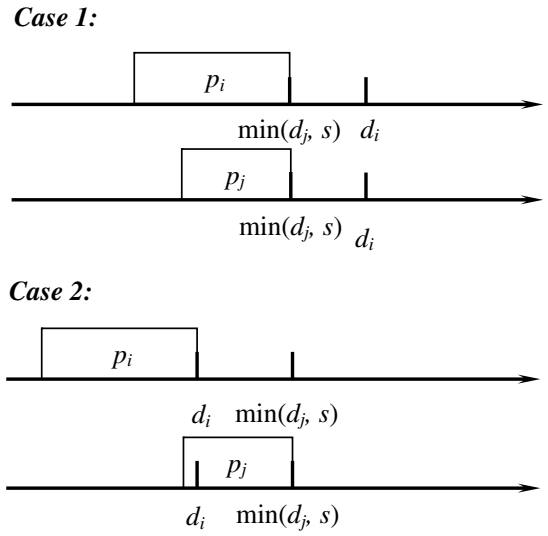

Fig. 2. Interchange between a non-tardy job i and a tardy job j. Case 1. $d_i > \min(d_j, s)$; Case 2. $d_i \leqslant \min(d_j, s)$.

Remark 4. This proposition indicates two sufficient conditions for interchanging a non-tardy job (in a block) and a tardy job. It effectively reduced the earliness cost as observed in the computational experiments.

3. The algorithms

In this section, we describe two algorithms for solving problem (**P**).

3.1. A heuristic algorithm and the polynomially solvable cases

Based on the propositions in last section, we give a heuristic algorithm below.

Heuristic algorithm (HA).

Input: a job set J.

Output: a sequence and a total weighted earliness with minimum number of tardy jobs for job set J.

Algorithm:

Step 0. Invoke Moore–Hodgson algorithm to get σ (non-tardy job sequence, in order) and σ' (tardy job set, arbitrary order), respectively. Assume the number of the non-tardy jobs is m.

Step 1. Let the sequence $\sigma = (1, 2, \ldots, m)$, $k \leftarrow m$, $s_k = d_k - p_k$;

Step 2. Find job j such that: $p_j/\alpha_j = \min_{i \in \sigma - \{k\}}\{p_i/\alpha_i : i \in \sigma \text{ and } d_i \geqslant s_k\}$ (break ties with large due date);

Step 3. If there exists such a job j, then $s_j = s_k - p_j$; $k \leftarrow j$, $\sigma \leftarrow \sigma - \{j\}$; else $j = \max\{j : j \in \sigma\}$, $s_j = d_j - p_j$; $k \leftarrow j$, $\sigma \leftarrow \sigma - \{j\}$;

Step 4. If $\sigma = \emptyset$, go to Step 5; otherwise, go to Step 2.

Step 5. For $i_1 = 1$ to $m - 1$:
For $i_2 = i_1$ to $m - 1$:
if the inequality (1), or both (2) and (3) for adjacent jobs i_2 and $i_2 + 1$ in σ hold, then interchange these two jobs.

Step 6. For $i_1 = 1$ to m:
For $i_2 = 1$ to $n - m$:
Determine if there is a job $i_2 \in \sigma'$ that can be interchanged with a job $i_1 \in \sigma$ by checking inequalities (4) and (5), or (4′) and (5′).

Step 7. The tardy jobs are sequenced after the last non-tardy job in an arbitrary order.

Remark 5. In this heuristic algorithm, the jobs are scheduled one by one backwards. Steps 5 and 6 improve the schedule by profitable interchanges of jobs. Note that the first part of the algorithm (Steps 0–4) is similar to the algorithm of Chand and Schneeberger (1988) while the second part (Steps 5–7) is developed specifically for problem (**P**). The experiments indicate that these steps can effectively reduce earliness cost.

Remark 6. In this heuristic algorithm, the computation time for step 0 is $O(n\log n)$, for step 2 to step 5 is $O(n)$, for step 6 is $O(n^2)$, and for step 7 is $O(n^3)$, thus the complexity of the heuristic algorithm is $O(n^3)$, which is polynomial.

Polynomially solvable cases: It is easy to show that the following cases are polynomially solvable using heuristic **HA**.

(1) Identical processing times and earliness weights: $p_j = p$ and $\alpha_j = \alpha$, for $j = 1, 2, \ldots, n$.
In this case, **HA** becomes the EDD rule with adjusting non-tardy jobs as close as possible to their due dates, and with tardy jobs identified by the Moore–Hodgson algorithm scheduled after the last non-tardy job. Thus **HA** finds the optimal solutions.

(2) A common due date: $d_j = d$, for $j = 1, 2, \ldots, n$.
First, by Moore–Hodgson algorithm, we know that in case of common due date, the processing time of any non-tardy job is less than or equal to that of any tardy job. Therefore, if the non-tardy jobs are scheduled by the WLPT rule and scheduled against d (Step 5 in heuristic **HA**), while the tardy jobs are scheduled arbitrarily after d, then the schedule is optimal. This is exactly what **HA** does, hence **HA** finds the optimal solution.

(3) Large due dates: $d_j \geqslant \sum_{j=1}^{n} p_j$, for $j = 1, 2, \ldots, n$.
In this case, it is easy to see that there is no tardy job. Thus the schedule by **HA** is optimal by Lemma 3 of Chand and Schneeberger (1988).

(4) $p_i/\alpha_i \geqslant p_j/\alpha_j$ implies $p_i \geqslant p_j$ and $d_j \geqslant d_i$, for $j = 1, 2, \ldots, n$.

In this case, the non-tardy jobs in the same block are scheduled by WLPT, and tardy jobs cannot be interchanged with non-tardy jobs (see Proposition 5). Thus by Theorem 2 in Chand and Schneeberger (1988), **HA** finds the optimal solutions.

Hence in all these cases, we could obtain the optimal solutions for the problems by using the heuristic **HA**.

3.2. A branch and bound algorithm

This subsection presents a branch and bound algorithm based on the properties developed in the last section to solve problem (**P**) optimally. The basic elements of the branch and bound algorithm are described as follows:

Search strategy: In the branch and bound algorithm, jobs are scheduled backwards by depth first search, and an incumbent value is obtained by heuristic **HA**.

Node representation: Let σ be the sequence of scheduled jobs (in increasing order of due dates, obtained backwards) and σ' be the set of unscheduled jobs, respectively. Let $s(\sigma)$ be the starting time of the first job in σ. Initially (when $\sigma = \emptyset$), $s(\sigma) = d_\ell - p_\ell$, where $\ell = \arg\max_{i \in \sigma'}\{d_i\}$ (break ties with smallest processing time). Now in the branch and bound tree, let σ also represent a node and $f(\sigma)$ be the lower bound of this node, which can be obtained by estimation (described below). Assume the job to be scheduled next is job k. Then job k is scheduled to start at time $\min\{s(\sigma), d_k\} - p_k$, and σ and σ' are updated as $\sigma \cup \{k\}$ and $\sigma' - \{k\}$, respectively. Continue this procedure until there are m scheduled jobs (where m is the number of non-tardy jobs).

Lower bound estimation: In the branch and bound algorithm, the following method is used to estimate the lower bound.

Define $f(\sigma \cup \{k\}) = f(\sigma) + g_0$ as a lower bound, where $f(\sigma)$ is the lower bound of node σ, and g_0 is obtained by calculating the earliness cost(s) of the unscheduled jobs whose due dates are greater than or equal to $s(\sigma)$. Note that $f(\sigma) = 0$ if $\sigma = \emptyset$.

Dominance conditions: In order to solve problems with a reasonable number of jobs, the following conditions for pruning the branches or nodes in the search tree are used.

Condition 1. Schedule the unscheduled job set σ' using the Moore–Hodgson algorithm and assume the non-tardy job set is E. Let job k be the job to be scheduled next. If $s(\sigma \cup \{k\}) < \sum_{j \in E} p_j$, then scheduling job k next (in the first position of the partial schedule) cannot lead to an optimal schedule.

Proof. Since the tardy job set obtained by the Moore–Hodgson algorithm majorizes any other tardy job set with the same number of tardy jobs (see Vairaktarakis and Lee, 1995 for the definition of the majorization of a tardy job set), it is impossible to further reduce the total processing time of the non-tardy jobs while keeping the minimum number of tardy jobs. Now that the total processing time of the non-tardy jobs is greater than the starting time of the first job in the scheduled job set, it is impossible to get a schedule with the minimum number of tardy jobs. □

The following two conditions (Conditions 2 and 3) follow from Propositions 3 and 4.

Condition 2. Suppose job j is the first job in σ. If $d_j \leqslant d_k$, and $p_j/\alpha_j \leqslant p_k/\alpha_k$, then scheduling job k next (in the first position of the partial schedule) cannot lead to an optimal schedule.

Condition 3. Suppose job j is the first job in σ. If $\min\{d_k, s\} \leqslant d_j$, and $(p_j/\alpha_j - p_k/\alpha_k) \leqslant (1/\alpha_j + 1/\alpha_k)(\min\{d_k, s\} - d_j)$, where s is the starting time of the second job in σ, then scheduling job k next (in the first position of the partial schedule) cannot lead to an optimal schedule.

The following condition originates from Chand and Schneeberger (1988) and Chen et al. (1997).

Condition 4. Let $s_\sigma = \min\{s(\sigma), \max_{i\in\sigma'}\{d_i\}\}, \delta = \min\{p_i | i \in \sigma' \text{ and } d_i \geqslant s(\sigma)\}$. Then scheduling an unscheduled job k with due date $d_k \leqslant s_\sigma - \delta$ next (in the first position of the partial schedule) cannot lead to an optimal schedule.

Remark 7. Condition 4 gives the maximum machine idle time immediately before the scheduled jobs. In the case where $s_\sigma = s(\sigma)$, the machine idle time is less than δ in an optimal schedule; in the case $s_\sigma < s(\sigma)$, the machine idle time is less than or equal to $s(\sigma) - \max_{i\in\sigma'}\{d_i\} + \delta$.

In the following, we use two examples to illustrate the heuristic **HA** as well as the branch and bound algorithm **BBA**.

Example 1. Suppose the problem has the job data shown in Table 1.

By the branch and bound algorithm **BBA**, the optimal schedule is $\{1(3), 4(6), 5(15)\}\{2, 3\}$, with $\sum_j \alpha_j E_j = 0$ and $\sigma' = \{2, 3\}$, where $j(s)$ denotes that job j starts its process at time s. Fig. 3 demonstrates the search tree for solving the problem, where **x** denotes an eliminated node. In Fig. 3, first the lower bound of node 0 is obtained as 1 by

Table 1
Job data in Example 1

j	1	2	3	4	5
p_j	3	7	5	6	10
d_j	6	8	10	12	25
α_j	1	1	1	1	1

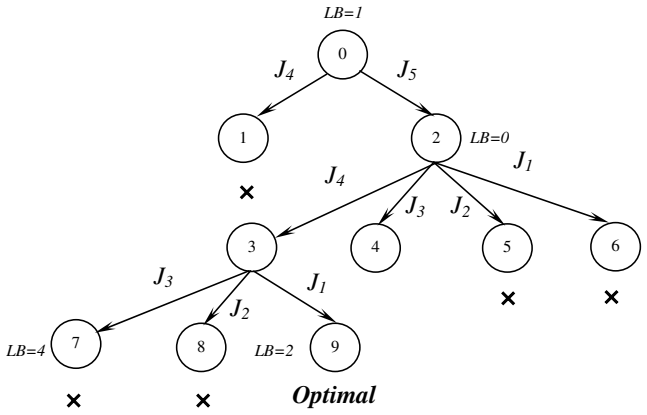

Fig. 3. A branch and bound search tree.

Table 2
Job data in Example 2

j	1	2	3	4
p_j	4	3	10	5
d_j	4	5	10	20
α_j	1	2	1	2

HA. Then consider job 4. However, job 4 cannot be scheduled at this position since the sum of processing times of any two other jobs is greater than $12 - 6 = 6$ (by Condition 1). Now consider job 5, then we reach node 2, whose total cost is not greater than the lower bound. Next, consider job 4, then we reach node 3, whose total cost is less than the lower bound 1. We continue this procedure until reach node 9, which gives an optimal solution to the problem.

On the other hand, by the Moore–Hodgson algorithm, we know $\sigma' = \{2, 3\}$ and $|\sigma'| = 2$. Now by heuristic **HA**, we can obtain a schedule $\{1(2), 4(6), 5(15)\}\{2, 3\}$ with $\sum_j \alpha_j E_j = 0$, which is an optimal schedule in this example.

Example 2. Suppose the problem has the job data shown in Table 2.

By the branch and bound algorithm **BBA**, the optimal schedule is $\{1(0), 3(4), 4(15)\}\{2\}$, with $\sum_j \alpha_j E_j = 0$ and $\sigma' = \{2\}$. By the Moore–Hodgson algorithm, we know $\sigma' = \{1\}$ and $|\sigma'| = 1$. Now by heuristic **HA**, we can obtain a schedule $\{2(1), 3(4), 4(15)\}\{1\}$ with $\sum_j \alpha_j E_j = 2$, which is, unfortunately, not optimal.

4. Computational experiments

To test the effectiveness of heuristic **HA** and efficiency of the branch and bound algorithm **BBA**, we implemented the algorithms with Borland C++ on a Pentium IV (1.5 GHz) personal computer.

The problem instances were generated in a manner similar to that used in Potts and van Wassenhove (1985) and Kim and Yano (1994), which is now a standard method to generate single machine scheduling problems with due dates. The processing times generated are uniformly dis-

Table 3
The performance of the heuristic algorithm ($n = 20$)

(EF, RDD)	Average (%)	Maximum (%)	# of optimal solutions
(0.1, 0.8)	6.2	20.4	2
(0.1, 1.0)	9.3	27	3
(0.1, 1.2)	5.5	15.6	2
(0.1, 1.4)	4.7	16.3	2
(0.1, 1.6)	5.2	18.9	3
(0.1, 1.8)	2.8	9.8	4
(0.2, 0.8)	9.2	15.7	2
(0.2, 1.0)	2.3	12.3	6
(0.2, 1.2)	4.5	16.4	5
(0.2, 1.4)	3.2	15.5	4
(0.2, 1.6)	0.6	3.4	5
(0.3,0.8)	4.2	22.6	7
(0.3,1.0)	4.1	18.6	5
(0.3,1.2)	3.9	32.8	9
(0.3,1.4)	5.8	25.1	4
(0.4,0.8)	9.4	30.3	5
(0.4,1.0)	3.9	16.2	6
(0.4,1.2)	4.9	17.4	4
(0.5,0.8)	17.7	37.9	3
(0.5,1.0)	8.3	21.6	4
Overall	5.785	19.69	(42.5%)

Table 4
The performance of the branch and bound algorithm ($n = 20$)

Problem instances (EF, RDD)	B&B with incumbent value (Average: seconds)	B&B with dominance conditions (Average: seconds)	B&B with both	
			Average (seconds)	Maximum (seconds)
(0.1, 0.8)	1083.6	581.8	438.4	2423.9
(0.1, 1.0)	437.9	208.9	147.4	673.8
(0.1, 1.2)	77.7	39.5	27.7	179.8
(0.1, 1.4)	15.6	7.8	4.3	9.2
(0.1, 1.6)	9.5	5.4	2.4	10.4
(0.1, 1.8)	8.9	3.97	2.2	5
(0.2, 0.8)	1254.9	477.9	353.7	2113.6
(0.2, 1.0)	86.2	33	24.5	194.4
(0.2, 1.2)	49.7	8.8	7.3	60.4
(0.2, 1.4)	5.8	1.9	0.7	5.3
(0.2, 1.6)	4.5	2.8	1.3	2.2
(0.3, 0.8)	3.9	2.1	1.6	2.5
(0.3, 1.0)	14.7	6.7	4.7	18.6
(0.3, 1.2)	13.9	6.2	4.8	21.7
(0.3, 1.4)	6.4	4.1	2.7	4.2
(0.4,0.8)	20.6	11.3	6.9	43.9
(0.4,1.0)	1.4	0.8	0.4	1.8
(0.4,1.2)	12.9	8.6	5.9	32.3
(0.5,0.8)	14.1	5.5	6.3	47.4
(0.5,1.0)	6.9	4.7	2.8	5.2
Overall	156.5	71.1	52.3	292.78

tributed in $U[1, 10]$ and the weights are randomly from the set $\{1, 2, 4\}$. Problem "hardness" depends on two parameters: EF (since we focus on the earliness penalties here, we call it earliness factor, similar to the tardiness factor as in Potts and van Wassenhove, 1985) and RDD (range of due date). EF takes value from set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ while RDD takes value from set $\{0.8, 1.0, 1.2, 1.4, 1.6, 1.8\}$. For a particular pair (EF, RDD), the due dates are integers uniformly distributed in $[(1 - \mathrm{EF} - \mathrm{RDD}/2) * \mathrm{MS}, (1 - \mathrm{EF} + \mathrm{RDD}/2) * \mathrm{MS}]$, where MS is the total processing time of all the jobs. Combinations of EF and RDD values are selected such that the due dates are non-negative (similar to those in Kim and Yano, 1994).

Experiment 1. To test heuristic **HA** in terms of the quality of the solutions with respect to (EF, RDD).

The average and the maximal deviations of objective values over the optimal solutions (which can be obtained by the branch and bound algorithm) are used as the performance measure. For problem instance k, define:

$$\mathrm{Performance}_k = \frac{\mathrm{HA}_k - \mathrm{Optimum}_k}{\mathrm{Optimum}_k} \times 100\%,$$

where HA_k and $\mathrm{Optimum}_k$ denote the objective value of solution by **HA** and the optimal objective value, respectively. Assume there are totally L instances with respect to a pair of (EF, RDD), then the numbers in the table are defined as

$$\mathrm{Average} = \frac{\sum_{k=1}^{L} \mathrm{Performance}_k}{L}$$

and

$$\mathrm{Maximum} = \max_{k=1,\ldots,L} \{\mathrm{Performance}_k\}.$$

In the experiment, the number of jobs was set as 20. For each pair of (EF, RDD), 10 problem instances ($L = 10$) were generated for testing, yielding a total of 200 problem instances. The results for the pairs of (EF, RDD) with Average and Maximum are summarized in the second column and third column in Table 3. The fourth column gives the number of optimal solutions obtained by the heuristic **HA**.

The computational results indicate that the heuristic algorithm performed preferably, particularly when RDD is relatively large, which means that the range of due dates are relatively large, as shown in Table 3. Overall, it obtained more than 40% optimal solutions in the experiment, the average objective value is only about 5.8% above the optimal objective value, and in the worst cases, the average objective value is about 20% above the optimal objective value. The computational results show that the heuristic algorithm is rather effective.

Experiment 2. To evaluate the branch and bound algorithm **BBA** in terms of computation time with respect to (EF, RDD).

This experiment was carried out with the same problem instances as in Experiment 1 but to evaluate computation time with respect to (EF, RDD). The running times of the branch and bound algorithm are summarized in Table 4, in

Table 5
The computation time of the branch and bound algorithm

N (number of jobs)	Average (seconds)	Minimum (seconds)	Maximal (seconds)
15	17.8	0.4	443.8
20	84.3	1.3	1798.7
25	189.4	3.7	4074.2
26	234.7	3.2	5991.3
27	321.9	5.4	8215.2
28	413.5	6.9	10173.9
29	757.3	6.3	13872.1
30	997.8	9.7	17410.3

which columns 2, 3, 4 show, respectively, the average running time when the incumbent value is used, the average running time when the dominance conditions are used, and the average and the maximum running time in the experiment with both the incumbent value and the dominance conditions (all in second).

The computational results indicate that the branch and bound algorithm runs quickly when RDD is relatively large, which means that the distribution of the due dates are not too "tight". The reason is, when the due dates are too "tight", it is hard for the algorithm to distinguish the jobs for the right positions. This is true particularly when EF is also small since in this situation, the due dates are not only "tight" but also relatively small, thus presenting difficulty to the branch and bound algorithm. Overall, the average running time is about 1 minute and the average maximum running time is about 5 minutes (the average minimum running time is not significant). Furthermore, the computational results also show the effectiveness of the dominance conditions. In fact, the average running time of the branch and bound algorithm with only the dominance conditions is around 36% more than that of the branch and bound algorithm, while the average running time of the branch and bound algorithm with only the incumbent value is almost three times of that of the branch and bound algorithm.

Experiment 3. To evaluate the branch and bound algorithm **BBA** in terms of computation time with respect to the number of jobs.

The numbers of jobs were set to be 15, 20, 25, 26, 27, 28, 29, and 30. The number of randomly generated problem instances was 30 for each case. All the parameters, including processing times, due dates and weights, were generated randomly. The results are summarized in Table 5, which shows the average, the minimum and the maximum running time in the experiment.

The computational results indicate the branch and bound algorithm can be used preferably for problems with less than 30 jobs (note that it almost doubled the job size of the algorithm in Chand and Schneeberger, 1988), while the maximal computational time may be relatively long and the algorithm may not be practical when the number of jobs exceeds 30.

5. Conclusions

We have studied a single machine scheduling problem with dual criteria, i.e., to minimize total weighted earliness subject to minimum number of tardy jobs. We first discussed several dominance properties of optimal solutions. Then we developed a heuristic algorithm and a branch and bound algorithm for solving the problem. We illustrated the branch and bound algorithm by two examples and carried out computational experiments to test both algorithms. The computational results show that the heuristic performs preferably in most cases and the branch and bound algorithm is suitable for medium-size problems. In the future, it would be interesting to consider the same problem on parallel machines.

Acknowledgements

The authors thank two anonymous referees for their constructive comments, which help to improve presentation of the paper substantially. This research is supported in part by NSF of China (70372058) and Guangdong NSF (031808).

References

Azizoglu, M., Kondakci, S., Koksalan, M., 2003a. Single machine scheduling with maximum earliness and number tardy. Computers and Industrial Engineering 45, 257–268.

Azizoglu, M., Koksalan, M., Koksalan, S.K., 2003b. Scheduling to minimize maximum earliness and number of tardy jobs where machine idle time is allowed. Journal of the Operational Research Society 54, 661–664.

Baker, K.R., Scudder, G.D., 1990. Sequencing with earliness and tardiness penalties: A review. Operations Research 38, 22–36.

Cai, X., Zhou, S., 1999. Stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy jobs. Operations Research 47, 422–437.

Chand, S., Schneeberger, H., 1988. Single machine scheduling to minimize weighted earliness subject to no tardy jobs. European Journal of Operations Research 34, 221–230.

Chen, C.L., Bulfin, R.L., 1993. Complexity of single machine multicriteria scheduling problems. European Journal of Operations Research 70, 115–125.

Chen, T., Qi, X., Tu, F., 1997. Single machine scheduling to minimize weighted earliness subject to maximum tardiness. Computers and Operations Research 24, 147–152.

Duffuaa, S.O., Raouf, A., Ben-Daya, M., Makki, M., 1997. One machine scheduling to minimize mean tardiness with minimum number tardy. Production Planning and Control 8, 226–230.

Guner, E., Erol, S., Tani, K., 1998. One machine scheduling to minimize the maximum earliness with minimum number of tardy jobs. International Journal of Production Economics 55, 213–219.

Hino, C.M., Ronconi, D.P., Mendes, A.B., 2005. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. European Journal of Operational Research 160, 190–201.

Karasakal, E.K., Koksalan, M., 2000. A simulated annealing approach to bicriteria scheduling problems on a single machine. Journal of Heuristics 6, 311–327.

Kim, Y.D., Yano, C.A., 1994. Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. Naval Research Logistics 41, 913–933.

Koksalan, M., Keha, A.B., 2003. Using genetic algorithms for single-machine bicriteria scheduling problems. European Journal of Operational Research 145, 543–556.

Lee, C.Y., Vairaktarakis, G.L., 1993. Complexity of single machine hierarchical scheduling. In: Pardalos, P.M. (Ed.), Complexity in Numerical Optimization. World Scientific, Singapore, pp. 269–298.

Mazzini, R., Armentano, V.A., 2001. A heuristic for single machine scheduling with early and tardy costs. European Journal of Operational Research 128, 129–146.

Nagar, A., Haddock, J., Heragu, S., 1995. Multiple and bicriteria scheduling: A literature survey. European Journal of Operational Research 81, 88–104.

Pathumnakul, S., Egbelu, P.J., 2005. Algorithm for minimizing weighted earliness penalty in single-machine problem. European Journal of Operational Research 161, 780–796.

Potts, C.N., van Wassenhove, L.N., 1985. A branch and bound algorithm for the total weighted tardiness problem. Operations Research 33, 363–377.

Sidney, J.B., 1973. An extension of Moore's due date algorithm. In: Elmaghraby, S.E. (Ed.), Symposium on the Theory of Scheduling and Its Applications. Springer-Verlag, New York, pp. 393–398.

Sridharan, V., Zhou, Z., 1996. A decision theory based scheduling procedure for single-machine weighted earliness and tardiness problems. European Journal of Operational Research 94, 292–301.

Vairaktarakis, G.L., Lee, C.Y., 1995. The single machine scheduling to minimize total tardiness subject to minimum number of tardy jobs. IIE Transactions 27, 250–256.

Wan, G., Yen, B.P.C., 2002. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. European Journal of Operational Research 142, 271–281.

Yen, B.P.C., Wan, G., 2003. Single machine bicriteria scheduling: A survey. Industrial Engineering: Theory Applications and Practice 10, 222–231.