Minimizing Weighted Earliness and Tardiness Penalties in Single-Machine Scheduling with Idle Time Permitted

Jaw-Yeh Chen, Sheng-Fuu Lin

Department of Electrical and Control Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 30010, Republic of China

Received 17 July 2000; revised 21 December 2001; accepted 1 March 2002

Abstract: In this paper, a single-machine scheduling problem with weighted earliness and tardiness penalties is considered. Idle time between two adjacent jobs is permitted and due dates of jobs could be unequal. The dominance rules are utilized to develop a relationship matrix, which allows a branch-and-bound algorithm to eliminate a high percentage of infeasible solutions. After combining this matrix with a branching strategy, a procedure to solve the problem is proposed. © 2002 Wiley Periodicals, Inc. Naval Research Logistics 49: 760–780, 2002; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/nav.10039

Keywords: scheduling theory; branch-and-bound; earliness/tardiness; idle time; dominance rule

1. INTRODUCTION

In recent times, the use of JIT, or just-in-time delivery, which aims to provide the customer with an order at a precisely desired time, has become a much-valued consideration. In the scheduling field, this is an earliness/tardiness (E/T) model [2, 9].

Many researchers have addressed either heuristic procedures [7] or branch-and-bound schemas [1, 3, 9, 11], or in some cases both. Abdul-Razag and Potts [1] looked at problems of job-independent weighted earliness and tardiness penalties without inserted idle time. The solution they proposed is a branch-and-bound scheme for which they developed a special relaxed dynamic programming procedure to obtain the lower bounds of the scheme.

For job-dependent weighted E/T penalty problems, Ow and Morton [7] described a family of heuristic dispatch rules, but in doing so they considered only schedules without inserted idle time. They also utilized dominance conditions for adjacent jobs to obtain optimal solutions for two special cases. In the first, a WSPT (weighted shortest processing time) sequence results in a schedule that has no early jobs; in the second, a WLPT (weighted longest processing time) sequence results in a schedule that has no tardy jobs. Szwarc [9] proposed a single-machine n job earliness-tardiness model with job-independent penalties. He arrived at results on orderings for adjacent jobs that showed that dominance ordering in an optimal schedule varied according to a critical value of start times. However, he considered only the case where idle time is not

Correspondence to: S.-F. Lin.

© 2002 Wiley Periodicals, Inc.

permitted. Many researchers [2, 5, 6, 8, 10] have pointed out that the assumption of no idle time is inconsistent with the E/T model, since if idle time is permitted in a sequence, then it is always better to insert it in the schedule than not to do so.

Also recently, the E/T model with idle time permitted has attracted much attention. Given a job sequence, optimal insertion of idle time in the E/T model has been studied [4, 10]. In the case where earliness and tardiness penalties are equal, Garey, Tarjan and Wilfong [4] proposed an $O(n \log n)$ complexity procedure, where n is the number of jobs, for inserting idle time optimally. Szwarc and Mukhopadhyay [10], using a cluster concept to develop the properties of the E/T model, proposed an algorithm of complexity $O(mn)$, where m is the number of clusters. Idle time is easily inserted optimally if the job sequence is determined. Therefore, the most important consideration is to find the best sequence. However, this two-stage procedure (that is, of finding a good job sequence into which to insert idle time) may not produce an optimal schedule. Consider, for example, a three-job problem where the processing time for each job is 3, 7, and 14, respectively, and the due date for each job is 10, 30, and 35, respectively. At the same time, assume both the earliness and tardiness penalties to be 1. If no idle time is permitted, the optimal sequence is jobs 1, 3, and 2, starting at time 7 with the cost being 12. But, if idle time is permitted, the optimal sequence is jobs 1, 2, and 3, with starting times 7, 14, and 21, respectively, then the cost is 9. In fact, there is more than one optimal schedule. For example, the sequence of jobs 1, 2, and 3, with starting times 7, 23, and 30, respectively, is also an optimal schedule, with the cost being 9. Going back to the sequence jobs 1, 3, and 2, which is the best sequence when idle time is not permitted, then the optimal cost is 12 when idle time is permitted. Hence, if the job sequence is scheduled without idle time in the first stage but inserted in the second stage, then the schedule may not be optimal.

Davis and Kanet [3] proposed a TIMETABLER procedure for adjusting idle time, and used this to establish a lower bound for branch-and-bound methods. Since this procedure involves computing a TIMETABLER in each partial schedule, the computations are extensive.

To deal with lower bound computation, Hoogeveen and Van de Velde [11] presented a fivefold approach after which they used branch-and-bound methods to solve the E/T model. They changed the form of the problem from that of being an E/T model into one in which the goal is to minimize the function that is dependent on completion time and total earliness.

A further problem arises with the E/T model. Koulamas [6] discussed the single-machine earliness/tardiness problem with arbitrary time windows (STW), dividing it into two subproblems. He first found a good job sequence, then optimally inserted idle time into the sequence. The results were based on heuristics developed for other single-machine scheduling problems, such as apparent urgency (AU), and adjacent pairwise interchange (API), with some modifications. As the above example shows, this two-stage procedure may not obtain the optimal job schedule. A recent survey of the literature on single-machine earliness and tardiness problems can be found in Baker and Scudder [2].

This study considers a single-machine scheduling problem with weighted earliness and tardiness penalties, and with unequal due dates and idle time permitted. As pointed out by Garey, Tarjan and Wilfong [4], as well as by Baker and Scudder [2], where the due dates are treated as decision variables, the problem turns out to be relatively simple. However, if the due dates are given and unequal, then the problem is NP-complete and only branch-and-bound schemata or heuristic procedures are possible. We choose to amalgamate the two stages simultaneously. First, dominance rules are utilized to develop a matrix that is referred to as the *relationship matrix* [2, 9]. Then, this matrix is combined with a branching strategy. Computations have been greatly reduced, as shown in the simulation results.

2. NOTATIONS

Let us consider a single-machine job-independent weighted earliness and tardiness penalties scheduling problem in which idle time inserted between two adjacent jobs is permitted. That is, given n jobs, $J_1, J_2, \ldots, J_n$, each with a processing time p_k and a due date d_k, for a job completed before its due date the penalty is α per unit time, while for a job completed after its due date the penalty is β per unit time and, between any two adjacent jobs, idle time is permitted. The objective of this problem is to find an optimal schedule such that the total weighted earliness and tardiness penalties can be minimized. The following notations and symbols will be used throughout this study.

J_i = job i,
p_k = processing time of J_k,
d_k = due date of J_k,
C_k = completion time of J_k,
s_k = starting time of J_k,
α = earliness penalty per unit time,
β = tardiness penalty per unit time,
$E_k = max(d_k - C_k, 0)$ = earliness of J_k,
$T_k = max(C_k - d_k, 0)$ = tardiness of J_k,
P, Q, PJ_iJ_j = a subschedule,
$S = PJ_iJ_jQ$ = a schedule,
$C(P)$ = completion time of subschedule P,
$S(Q)$ = starting time of subschedule Q,
$f(S) = \sum_{k=1}^{n} (\alpha E_k + \beta T_k)$ = E/T objective function for schedule S,
J_iJ_j = a subschedule in which J_i and J_j are adjacent jobs, and idle time may exist between J_i and J_j,
$V_k(i, j)$ = kth critical interval (see Section 3) of subschedule J_i, J_j in which J_i must precede J_j (i.e., $J_i \rightarrow J_j$) if $C(P)$ is within this interval, where $k \leq 5$ [that is, if $C(P) \in V_k(i, j)$, then J_i must precede J_j and if $C(P) \in V_k(j, i)$, then J_j must precede J_i].

The problem can then be stated as follows. Find schedule S that minimizes $f(S)$, in which idle time may be inserted between any two adjacent jobs.

3. THE RELATIONSHIP MATRIX

In this section, we discuss a method for determining the *critical interval set* of adjacent jobs, J_i and J_j, $\{V_k(i, j) | k \leq 5\}$, so that the first will precede the other if the completion time of the job that comes immediately before J_i and J_j, which we shall term the latest job, falls within an interval. The element m_{ij} in the *relationship matrix* $M = [m_{ij}]_{n \times n}$ represents the critical interval set $\{V_k(i, j)\}$. When jobs J_i and J_j are scheduled as adjacent, some conditions must be met to verify whether J_i must precede J_j, according to the latest job completion time, the starting time of the subschedule next to J_i and J_j, their due dates, processing times, and weighted penalties. These conditions are referred to as the *dominance properties* [5, 9] of two adjacent jobs. To derive the relationship matrix for the E/T model, we must first discuss the dominance properties of two adjacent jobs. Consider two adjacent jobs, J_i and J_j. Let PJ_iJ_jQ and PJ_jJ_iQ be the two

schedules. Also assume the completion time for subschedule P is $C(P)$, and the starting time for subschedule Q is $S(Q)$. Define $\delta_{ij} = f(PJ_jJ_iQ) - f(PJ_iJ_jQ)$ as the change in cost due to interchanging jobs J_i and J_j. A study of the properties of schedule PJ_iJ_jQ will allow us to classify it into five *schedule patterns,* which can be used to describe how the starting times of two adjacent jobs J_i and J_j are arranged:

(i) $d_j - p_i - p_j \geq d_i - p_i \geq C(P)$: According to $S(Q)$, there are three situations:

- **(ia)** $S(Q) \geq d_j$: Both J_i and J_j are arranged as on-time jobs; hence, $s_i = d_i - p_i$ and $s_j = d_j - p_j$.
- **(ib)** $d_j > S(Q) \geq d_i + p_j$: Since $S(Q)$ is less than d_j, J_j cannot be scheduled on time; hence, $s_i = d_i - p_i$ and $s_j = S(Q) - p_j$.
- **(ic)** $d_i + p_j > S(Q) \geq C(P) + p_i + p_j$: $S(Q)$ is so small that not even J_i can be scheduled on time; hence, $s_i = S(Q) - p_i - p_j$ and $s_j = S(Q) - p_j$. Note that $S(Q)$ must be greater than $C(P) + p_i + p_j$, otherwise, J_i and J_j cannot be scheduled between subschedules P and Q.

(ii) $d_j - p_i - p_j \geq C(P) > d_i - p_i$: According to $S(Q)$, there are two situations:

- **(iia)** $S(Q) \geq d_j$: Since J_i cannot be completed on time, to get minimum tardiness, J_i is released at time $C(P)$. But J_j can be arranged as an on-time job. Then the pattern is $s_i = C(P)$ and $s_j = d_j - p_j$.
- **(iib)** $d_j > S(Q) \geq C(P) + p_i + p_j$: Neither J_i and J_j can be scheduled on time; hence, $s_i = C(P)$ and $s_j = S(Q) - p_j$.

(iii) $C(P) > max(d_i - p_i, d_j - p_i - p_j)$: Since J_i cannot be completed on time, to obtain minimum tardiness, J_i is released at time $C(P)$. In this case, J_j must be a tardy job. Again, to obtain minimum tardiness, J_j must start its processing at time C_i. The pattern is $s_i = C(P)$ and $s_j = C_i$. Note that $S(Q)$ does not affect the pattern.

(iv) $d_i - p_i \geq C(P) > d_j - p_i - p_j$ and $\alpha \leq \beta$: In this case, it is impossible for both J_i and J_j to be on-time jobs. Since $\alpha \leq \beta$, to minimize the cost, J_i is completed early and J_j tardy with $s_i = C(P)$ and $s_j = C_i$. Also $S(Q)$ does not affect the pattern.

(v) $d_i - p_i > d_j - p_i - p_j$, $d_i - p_i \geq C(P)$ and $\alpha > \beta$: According to $S(Q)$, there are three situations:

- **(va)** $S(Q) \geq d_i + p_j$: In this case, it is impossible for both J_i and J_j to be on-time jobs. Since $\alpha > \beta$, to minimize the cost, J_i is arranged as an on-time job, and J_j as a tardy one with $s_j = d_i$. Hence, the pattern is $s_i = d_i - p_i$ and $s_j = d_i$.
- **(vb)** $d_i + p_j > S(Q) \geq max(d_j, C(P) + p_i + p_j)$: Both jobs must be shifted earlier than (va) to satisfy the constraint of $S(Q)$; hence, $s_i = S(Q) - p_i - p_j$ and $s_j = S(Q) - p_j$.
- **(vc)** $d_j > S(Q) \geq C(P) + p_i + p_j$: The cost is different from (vb), but the pattern is the same; hence, $s_i = S(Q) - p_i - p_j$ and $s_j = S(Q) - p_j$.

(vi) $d_i - p_i > d_j - p_i - p_j \geq C(P)$ and $\alpha \leq \beta$: According to $S(Q)$, there are two situations:

(via) $S(Q) \geq d_j$: In this case, it is impossible for both J_i and J_j to be on-time jobs. Since $\alpha \leq \beta$, to minimize the cost, J_i is completed early with $s_i = d_j - p_i - p_j$ and J_j on time. Then, the pattern is $s_i = d_j - p_i - p_j$ and $s_j = d_j - p_j$.

(vib) $d_j > S(Q) \geq C(P) + p_i + p_j$: Both jobs must be shifted earlier than (via) to satisfy the constraint of $S(Q)$; hence, $s_i = S(Q) - p_i - p_j$ and $s_j = S(Q) - p_j$.

Similarly, schedule PJ_jJ_iQ can also be classified into five schedule patterns:

(vii) $d_i - p_i - p_j \geq d_j - p_j \geq C(P)$:

(viia) $S(Q) \geq d_i$: The pattern is $s_j = d_j - p_j$ and $s_i = d_i - p_i$.

(viib) $d_i > S(Q) \geq d_j + p_i$: The pattern is $s_j = d_j - p_j$ and $s_i = S(Q) - p_i$.

(viic) $d_j + p_i > S(Q) \geq C(P) + p_i + p_j$: The pattern is $s_j = S(Q) - p_i - p_j$ and $s_i = S(Q) - p_i$.

(viii) $d_i - p_i - p_j \geq C(P) > d_j - p_j$:

(viiia) $S(Q) \geq d_i$: The pattern is $s_j = C(P)$ and $s_i = d_i - p_i$.

(viiib) $d_i > S(Q) \geq C(P) + p_i + p_j$: The pattern is $s_j = C(P)$ and $s_i = S(Q) - p_i$.

(ix) $(C(P) > max(d_j - p_j, d_i - p_i - p_j))$: The pattern is $s_j = C(P)$ and $s_i = C_j$.

(x) $d_j - p_j \geq C(P) > d_i - p_i - p_j$ and $\alpha \leq \beta$: In this case, the pattern is $s_j = C(P)$ and $s_i = C_j$.

(xi) $d_j - p_j > d_i - p_i - p_j$, $d_j - p_j \geq C(P)$ and $\alpha > \beta$:

(xia) $S(Q) \geq d_j + p_i$: The pattern is $s_j = d_j - p_j$ and $s_i = d_j$.

(xib) $d_j + p_i > S(Q) \geq max(d_i, C(P) + p_i + p_j)$: The pattern is $s_j = S(Q) - p_i - p_j$ and $s_i = S(Q) - p_i$.

(xic) $d_i > S(Q) \geq C(P) + p_i + p_j$: The pattern is $s_j = S(Q) - p_i - p_j$ and $s_i = S(Q) - p_i$.

(xii) $d_j - p_j > d_i - p_i - p_j \geq C(P)$ and $\alpha \leq \beta$:

(xiia) $S(Q) \geq d_i$: The pattern is $s_j = d_i - p_i - p_j$ and $s_i = d_i - p_i$.

(xiib) $d_i > S(Q) \geq C(P) + p_i + p_j$: The pattern is $s_j = S(Q) - p_i - p_j$ and $s_i = S(Q) - p_i$.

Let us consider a pair of adjacent jobs J_i and J_j. To obtain a better schedule, we must compare the cost of PJ_iJ_jQ and PJ_jJ_iQ. A total of 36 combinations formed from the two sets of schedule patterns must be checked. But some combinations can be ignored, for example, case (i) with case (vii). Since $d_j - p_i - p_j \geq d_i - p_i$ and $d_i - p_i - p_j \geq d_j - p_j$, it implies $d_j - p_i - p_j > d_j - p_j$, indicating that the intersection of the two considered domains is disjointed. Finally, we have 20 possible combinations as shown in Table 1. For example, (1) is in column (i) and row (x) which means that case (1) is a combination of patterns (i) and (x). In other words, in case (1), when J_i must precede J_j, schedule pattern (i) should be adopted, and when J_j must precede J_i, schedule pattern (x) should be adopted. The details are discussed in the following:

Table 1.

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
(vii)				(9)	(14)	(18)
(viii)				(10)	(15)	
(ix)			(6)	(11)	(16)	
(x)	(1)	(4)	(7)	(12)		(19)
(xi)	(2)	(5)	(8)		(17)	
(xii)	(3)			(13)		(20)

(1) For the combination of cases (i) and (x), the assumptions are $d_j - p_i - p_j \geq d_i - p_i \geq C(P) > d_i - p_i - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are three situations:

- **(1a)** $S(Q) \geq d_j$: Since $\delta_{ij} = \alpha(d_j - C(P) - p_j) + \beta(C(P) + p_i + p_j - d_i) > 0$, J_i must precede J_j.
- **(1b)** $d_j > S(Q) \geq d_i + p_j$: Since $\delta_{ij} = \alpha(S(Q) - C(P) - p_j) + \beta(C(P) + p_i + p_j - d_i) > 0$, J_i must precede J_j.
- **(1c)** $d_i + p_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(2S(Q) - C(P) - 2p_j - d_i) + \beta(C(P) + p_i + p_j - d_i)$.

(2) For the combination of cases (i) and (xi), the assumptions are $d_j - p_i - p_j \geq d_i - p_i \geq C(P)$ and $\alpha > \beta$. According to $S(Q)$, there are five situations:

- **(2a)** $S(Q) \geq d_j + p_i$: Since $\delta_{ij} = \beta(d_j + p_i - d_i) > 0$, J_i must precede J_j.
- **(2b)** $d_j + p_i > S(Q) \geq d_j$: Since $\delta_{ij} = \beta(S(Q) - d_i) + \alpha(d_j + p_i - S(Q)) > 0$, J_i must precede J_j.
- **(2c)** $d_j > S(Q) \geq d_i + p_j$: Since $\delta_{ij} = \beta(S(Q) - d_i) + \alpha p_i > 0$, J_i must precede J_j.
- **(2d)** $d_i + p_j > S(Q) \geq max(d_i, C(P) + p_i + p_j)$: $\delta_{ij} = \beta(S(Q) - d_i) + \alpha(p_i - p_j - d_i + S(Q))$.
- **(2e)** $d_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

(3) For the combination of cases (i) and (xii), the assumptions are $d_j - p_i - p_j \geq d_i - p_i > d_i - p_i - p_j \geq C(P)$ and $\alpha \leq \beta$. According to $S(Q)$, there are four situations:

- **(3a)** $S(Q) \geq d_j$: Since $\delta_{ij} = \alpha(d_j + p_i - d_i) > 0$, J_i must precede J_j.
- **(3b)** $d_j > S(Q) \geq d_i + p_j$: Since $\delta_{ij} = \alpha(S(Q) + p_i - d_i) > 0$, J_i must precede J_j.
- **(3c)** $d_i + p_j > S(Q) \geq d_i$: $\delta_{ij} = \alpha(2S(Q) + p_i - 2d_i - p_j)$.
- **(3d)** $d_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

(4) For the combination of cases (ii) and (x), the assumptions are $d_j - p_i - p_j \geq C(P) > d_i - p_i$ and $\alpha \leq \beta$. According to $S(Q)$, there are two situations:

- **(4a)** $S(Q) \geq d_j$: Since $\delta_{ij} = \alpha(d_j - C(P) - p_j) + \beta P_j > 0$, J_i must precede J_j.
- **(4b)** $d_j > S(Q) \geq C(P) + p_i + p_j$: Since $\delta_{ij} = \alpha(S(Q) - C(P) - p_j) + \beta P_j > 0$, J_i must precede J_j.

(5) For the combination of cases (ii) and (xi), the assumptions are $d_j - p_i - p_j \geq C(P) > d_i - p_i$ and $\alpha > \beta$. According to $S(Q)$, there are three situations:

(5a) $S(Q) \geq d_j + p_i$: Since $\delta_{ij} = \beta(d_j - C(P)) > 0$, J_i must precede J_j.
(5b) $d_j + p_i > S(Q) \geq d_j$: Since $\delta_{ij} = \beta(S(Q) - C(P) - p_i) + \alpha(d_j + p_i - S(Q)) > 0$, J_i must precede J_j.
(5c) $d_j > S(Q) \geq C(P) + p_i + p_j$: Since $\delta_{ij} = \beta(S(Q) - C(P) - p_i) + \alpha p_i > 0$, J_i must precede J_j.

(6) For the combination of cases (iii) and (ix), the assumption is $C(P) > max(d_i - p_i, d_j - p_j)$, which results in $\delta_{ij} = \beta(p_j - p_i)$.
(7) For the combination of cases (iii) and (x), the assumptions are $d_j - p_j \geq C(P) > max(d_i - p_i, d_j - p_i - p_j)$ and $\alpha \leq \beta$, which results in $\delta_{ij} = \alpha(d_j - C(P) - p_j) + \beta(d_j - C(P) - p_i)$.
(8) For the combination of cases (iii) and (xi), the assumptions are $d_j - p_j \geq C(P) > max(d_i - p_i, d_j - p_i - p_j)$ and $\alpha > \beta$. According to $S(Q)$, there are two situations:

(8a) $S(Q) \geq d_j + p_i$: $\delta_{ij} = \beta(2d_j - 2C(P) - p_i - p_j)$.
(8b) $d_j + p_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \beta(S(Q) + d_j - 2C(P) - 2p_i - p_j) + \alpha(d_j + p_i - S(Q))$.

(9) For the combination of cases (iv) and (vii), the assumptions are $d_i - p_i - p_j \geq d_j - p_j \geq C(P) > d_j - p_i - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are three situations:

(9a) $S(Q) \geq d_i$: Since $\delta_{ij} = -\alpha(d_i - C(P) - p_i) - \beta(C(P) + p_i + p_j - d_j) < 0$, J_j must precede J_i.
(9b) $d_i > S(Q) \geq d_j + p_i$: Since $\delta_{ij} = -\alpha(S(Q) - C(P) - p_i) - \beta(C(P) + p_i + p_j - d_j) < 0$, J_j must precede J_i.
(9c) $d_j + p_i > S(Q) \geq C(P) + p_i + p_j$: Since $\delta_{ij} = -\alpha(2S(Q) - C(P) - 2p_i - d_j) - \beta(C(P) + p_i + p_j - d_j) < 0$, J_j must precede J_i.

(10) For the combination of cases (iv) and (viii), the assumptions are $d_i - p_i - p_j \geq C(P) > d_j - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are two situations:

(10a) $S(Q) \geq d_i$: Since $\delta_{ij} = -\alpha(d_i - C(P) - p_i) - \beta p_i < 0$, J_j must precede J_i.
(10b) $d_i > S(Q) \geq C(P) + p_i + p_j$: Since $\delta_{ij} = -\alpha(S(Q) - C(P) - p_i) - \beta p_i < 0$, J_j must precede J_i.

(11) For the combination of cases (iv) and (ix), the assumptions are $d_i - p_i \geq C(P) > max(d_j - p_j, d_i - p_i - p_j)$ and $\alpha \leq \beta$, which results in $\delta_{ij} = -\alpha(d_i - C(P) - p_i) + \beta(C(P) + p_j - d_i)$.
(12) For the combination of cases (iv) and (x), the assumptions are $min(d_i - p_i, d_j - p_j) \geq C(P) > max(d_i - p_i - p_j, d_j - p_i - p_j)$ and $\alpha \geq \beta$, which results in $\delta_{ij} = \alpha(d_j - d_i + p_i - p_j) + \beta(d_j - d_i)$.
(13) For the combination of cases (iv) and (xii), the assumptions are $d_j - p_j > d_i - p_i - p_j \geq C(P) > d_j - p_i - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are two situations:

(13a) $S(Q) \geq d_i$: $\delta_{ij} = \alpha(d_j - 2d_i + 2p_i + C(P)) - \beta(C(P) + p_i + p_j - d_j)$.

(13b) $d_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(d_j - 2S(Q) + 2p_i + C(P)) - \beta(C(P) + p_i + p_j - d_j)$.

(14) For the combination of cases (v) and (vii), the assumptions are $d_i - p_i - p_j \geq d_j - p_j \geq C(P)$ and $\alpha > \beta$. According to $S(Q)$, there are five situations:

(14a) $S(Q) \geq d_i + p_j$: Since $\delta_{ij} = -\beta(d_i + p_j - d_j) < 0$, J_j must precede J_i.
(14b) $d_i + p_j > S(Q) \geq d_i$: Since $\delta_{ij} = -\beta(S(Q) - d_j) - \alpha(d_i + p_j - S(Q)) < 0$, J_j must precede J_i.
(14c) $d_i > S(Q) \geq d_j + p_i$: Since $\delta_{ij} = -\beta(S(Q) - d_j) - \alpha p_j < 0$, J_j must precede J_i.
(14d) $d_j + p_i > S(Q) \geq max(d_j, C(P) + p_i + p_j)$: $\delta_{ij} = \alpha(d_j - S(Q) + p_i - p_j) - \beta(S(Q) - d_j)$.
(14e) $d_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

(15) For the combination of cases (v) and (viii), the assumptions are $d_i - p_i - p_j \geq C(P) > d_j - p_j$ and $\alpha > \beta$. According to $S(Q)$, there are three situations:

(15a) $S(Q) \geq d_i + p_j$: Since $\delta_{ij} = \beta(C(P) - d_i) < 0$, J_j must precede J_i.
(15b) $d_i + p_j > S(Q) \geq d_i$: Since $\delta_{ij} = \beta(C(P) + p_j - S(Q)) - \alpha(d_i + p_j - S(Q)) < 0$, J_j must precede J_i.
(15c) $d_i > S(Q) \geq C(P) + p_i + p_j$: Since $\delta_{ij} = \beta(C(P) + p_j - S(Q)) - \alpha p_j < 0$, J_j must precede J_i.

(16) For the combination of cases (v) and (ix), the assumptions are $d_i - p_i \geq C(P) > max(d_j - p_j, d_i - p_i - p_j)$ and $\alpha > \beta$. According to $S(Q)$, there are two situations:

(16a) $S(Q) \geq d_i + p_j$: $\delta_{ij} = \beta(2C(P) + p_i + p_j - 2d_i)$.
(16b) $d_i + p_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \beta(2C(P) + p_i + 2p_j - d_i - S(Q)) - \alpha(d_i + p_j - S(Q))$.

(17) For the combination of cases (v) and (xi), the assumptions are $min(d_i - p_i, d_j - p_j) \geq C(P)$, $d_i - p_i > d_j - p_i - p_j$, $d_j - p_j > d_i - p_i - p_j$ and $\alpha > \beta$. The situations are as follows.

(17a) $S(Q) \geq max(d_i + p_j, d_j + p_i)$: $\delta_{ij} = \beta(2d_j - 2d_i + p_i - p_j)$.

If $d_i + p_j \geq d_j + p_i$ then the following two terms are adopted:

(17b1) $d_i + p_j > S(Q) \geq d_j + p_i$: $\delta_{ij} = \beta(2d_j + p_i - d_i - S(Q)) - \alpha(d_i + p_j - S(Q))$.
(17c1) $d_j + p_i > S(Q) \geq max(d_i, d_j, C(P) + p_i + p_j)$: $\delta_{ij} = \beta(d_j - d_i) + \alpha(d_j - d_i + p_i - p_j)$.

Else the following two terms are adopted:

(17b2) $d_j + p_i > S(Q) \geq d_i + p_j$: $\delta_{ij} = \beta(S(Q) - 2d_i - p_j + d_j) + \alpha(d_j + p_i - S(Q))$.

(17c2) $d_i + p_j > S(Q) \geq max(d_i, d_j, C(P) + p_i + p_j)$: $\delta_{ij} = \beta(d_j - d_i) + \alpha(d_j - d_i + p_i - p_j)$.

If $C(P) + p_i + p_j > d_i$ and $C(P) + p_i + p_j > d_j$; then the following four branches vanish. If not, the branches appear:

Branch 1:

(17d1) $d_i > S(Q) \geq d_j \geq C(P) + p_i + p_j$: $\delta_{ij} = -\beta(S(Q) - d_j) + \alpha(d_j + p_i - p_j - S(Q))$.
(17e1) $d_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

Branch 2:

(17d2) $d_i > S(Q) \geq C(P) + p_i + p_j > d_j$: $\delta_{ij} = -\beta(S(Q) - d_j) + \alpha(d_j + p_i - p_j - S(Q))$.

Branch 3:

(17d3) $d_j > S(Q) \geq d_i \geq C(P) + p_i + p_j$: $\delta_{ij} = \beta(S(Q) - d_i) + \alpha(S(Q) + p_i - p_j - d_i)$.
(17e3) $d_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

Branch 4:

(17d4) $d_j > S(Q) \geq C(P) + p_i + p_j > d_i$: $\delta_{ij} = \beta(S(Q) - d_i) + \alpha(S(Q) + p_i - p_j - d_i)$.

(18) For the combination of cases (vi) and (vii), the assumptions are $d_i - p_i - p_j \geq d_j - p_j > d_j - p_i - p_j \geq C(P)$ and $\alpha \leq \beta$. According to $S(Q)$, there are four situations:

(18a) $S(Q) \geq d_i$: Since $\delta_{ij} = -\alpha(d_i - d_j + p_j) < 0$, J_j must precede J_i.
(18b) $d_i > S(Q) \geq d_j + p_i$: Since $\delta_{ij} = -\alpha(p_j - d_j + S(Q)) < 0$, J_j must precede J_i.
(18c) $d_j + p_i > S(Q) \geq d_j$: $\delta_{ij} = \alpha(2d_j + p_i - p_j - 2S(Q))$.
(18d) $d_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

(19) For the combination of cases (vi) and (x), the assumptions are $d_i - p_i > d_j - p_i - p_j \geq C(P) > d_i - p_i - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are two situations:

(19a) $S(Q) \geq d_j$: $\delta_{ij} = \alpha(2d_j - d_i - 2p_j - C(P)) + \beta(C(P) + p_i + p_j - d_i)$.
(19b) $d_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(2S(Q) - C(P) - 2p_j - d_i) + \beta(C(P) + p_i + p_j - d_i)$.

(20) For the combination of cases (vi) and (xii), the assumptions are $min(d_i - p_i - p_j, d_j - p_i - p_j) \geq C(P)$, $d_i - p_i > d_j - p_i - p_j$, $d_j - p_j > d_i - p_i - p_j$ and $\alpha \leq \beta$. According to $S(Q)$, there are three situations:

If $d_i \geq d_j$ then the three situations are:

(20a1) $S(Q) \geq d_i$: $\delta_{ij} = \alpha(2d_j - 2d_i + p_i - p_j)$.
(20b1) $d_i > S(Q) \geq d_j$: $\delta_{ij} = \alpha(2d_j + p_i - p_j - 2S(Q))$.

(20c1) $d_j > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

Else the three situations are:

(20a2) $S(Q) \geq d_j$: $\delta_{ij} = \alpha(2d_j - 2d_i + p_i - p_j)$.
(20b2) $d_j > S(Q) \geq d_i$: $\delta_{ij} = \alpha(p_i - 2d_i - p_j + 2S(Q))$.
(20c2) $d_i > S(Q) \geq C(P) + p_i + p_j$: $\delta_{ij} = \alpha(p_i - p_j)$.

The fact that cases (1)–(20) cover all the situations can be easily verified. Note that once d_i, d_j, p_i, and p_j are given, only one of cases (1)–(20) can match. Although cases (1)–(20) describe sufficiently the dominance properties of two adjacent jobs J_i and J_j, they cannot be used conveniently. Hence some rearrangement of them is made. Now the jobs are numbered such that

$$p_1 \leq p_2 \leq \cdots \leq p_n,$$

and i is placed in front of j whenever $p_i = p_j$ and $d_i < d_j$. For cases (1)–(20), it can be seen that $C(P)$ always falls in an interval. The end points of the interval may be one of the following four points: $d_i - p_i$, $d_j - p_j$, $d_i - p_i - p_j$, and $d_j - p_i - p_j$. The schedule pattern will vary according to the magnitude sequence of these four points. Note that $d_j - p_j > d_i - p_i > d_i - p_i - p_j > d_j - p_i - p_j$, $d_i - p_i = d_j - p_j > d_i - p_i - p_j > d_j - p_i - p_j$, and $d_j - p_j > d_i - p_i > d_i - p_i - p_j = d_j - p_i - p_j$ are all impossible cases. For example, if $d_j - p_j > d_i - p_i > d_i - p_i - p_j > d_j - p_i - p_j$, then $d_j - p_j > d_i - p_i$ implies $d_j > d_i$ because $p_i \leq p_j$; and if $d_i - p_i - p_j > d_j - p_i - p_j$ implies $d_i > d_j$, then a contradiction occurs. The other two conditions are similar. Then cases (1)–(20) are rearranged as follows:

CASE A: If $d_i - p_i > d_j - p_j > d_i - p_i - p_j > d_j - p_i - p_j$, there are two conditions:

A1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (16)**),
- $d_j - p_j \geq C(P)$, (i.e., **case (17)**),

A2. If $\alpha \leq \beta$, then five possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (11)**),
- $d_j - p_j \geq C(P) > d_i - p_i - p_j$, (i.e., **case (12)**),
- $d_i - p_i - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (13)**),
- $d_j - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE B: If $d_i - p_i > d_j - p_j > d_j - p_i - p_j > d_i - p_i - p_j$, there are two conditions:

B1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (16)**),

- $d_j - p_j \geq C(P)$, (i.e., **case (17)**),

B2. If $\alpha \leq \beta$, then five possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (11)**),
- $d_j - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (12)**),
- $d_j - p_i - p_j \geq C(P) > d_i - p_i - p_j$, (i.e., **case (19)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE C: If $d_i - p_i > d_i - p_i - p_j > d_j - p_j > d_j - p_i - p_j$, there are two conditions:

C1. If $\alpha > \beta$, then four possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (16)**),
- $d_i - p_i - p_j \geq C(P) > d_j - p_j$, (i.e., **case (15)**),
- $d_j - p_j \geq C(P)$, (i.e., **case (14)**),

C2. If $\alpha \leq \beta$, then five possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (11)**),
- $d_i - p_i - p_j \geq C(P) > d_j - p_j$, (i.e., **case (10)**),
- $d_j - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (9)**),
- $d_j - p_i - p_j \geq C(P)$, (i.e., **case (18)**).

CASE D: If $d_j - p_j > d_i - p_i > d_j - p_i - p_j > d_i - p_i - p_j$, there are two conditions:

D1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),
- $d_j - p_j \geq C(P) > d_i - p_i$, (i.e., **case (8)**),
- $d_i - p_i \geq C(P)$, (i.e., **case (17)**),

D2. If $\alpha \leq \beta$, then five possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),
- $d_j - p_j \geq C(P) > d_i - p_i$, (i.e., **case (7)**),
- $d_i - p_i \geq C(P) > d_j - p_i - p_j$, (i.e., **case (12)**),
- $d_j - p_i - p_j \geq C(P) > d_i - p_i - p_j$, (i.e., **case (19)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE E: If $d_j - p_j > d_j - p_i - p_j > d_i - p_i > d_i - p_i - p_j$, there are two conditions:

E1. If $\alpha > \beta$, then four possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),

- $d_j - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (8)**),
- $d_j - p_i - p_j \geq C(P) > d_i - p_i$, (i.e., **case (5)**),
- $d_i - p_i \geq C(P)$, (i.e., **case (2)**),

E2. If $\alpha \leq \beta$, then five possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),
- $d_j - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (7)**),
- $d_j - p_i - p_j \geq C(P) > d_i - p_i$, (i.e., **case (4)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (1)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (3)**).

CASE F: If $d_i - p_i > d_j - p_j > d_i - p_i - p_j = d_j - p_i - p_j$, there are two conditions:

F1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (16)**),
- $d_j - p_j \geq C(P)$, (i.e., **case (17)**),

F2. If $\alpha \leq \beta$, then four possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (11)**),
- $d_j - p_j \geq C(P) > d_i - p_i - p_j$, (i.e., **case (12)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE G: If $d_i - p_i > d_j - p_j = d_i - p_i - p_j > d_j - p_i - p_j$, there are two conditions:

G1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (16)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (14)**),

G2. If $\alpha \leq \beta$, then four possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_j$, (i.e., **case (11)**),
- $d_j - p_j \geq C(P) > d_j - p_i - p_j$, (i.e., **case (9)**),
- $d_j - p_i - p_j \geq C(P)$., (i.e., **case (18)**).

CASE H: If $d_i - p_i = d_j - p_j > d_i - p_i - p_j = d_j - p_i - p_j$, there are two conditions:

H1. If $\alpha > \beta$, then two possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P)$, (i.e., **case (17)**),

H2. If $\alpha \leq \beta$, then three possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (12)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE I: If $d_i - p_i = d_j - p_j > d_j - p_i - p_j > d_i - p_i - p_j$, there are two conditions:

I1. If $\alpha > \beta$, then two possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P)$, (i.e., **case (17)**),

I2. If $\alpha \leq \beta$, then four possibilities occur:

- $C(P) > d_i - p_i$, (i.e., **case (6)**),
- $d_i - p_i \geq C(P) > d_j - p_i - p_j$, (i.e., **case (12)**),
- $d_j - p_i - p_j \geq C(P) > d_i - p_i - p_j$, (i.e., **case (19)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (20)**).

CASE J: If $d_j - p_j > d_i - p_i = d_j - p_i - p_j > d_i - p_i - p_j$, there are two conditions:

J1. If $\alpha > \beta$, then three possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),
- $d_j - p_j \geq C(P) > d_i - p_i$, (i.e., **case (8)**),
- $d_i - p_i \geq C(P)$, (i.e., **case (2)**),

J2. If $\alpha \leq \beta$, then four possibilities occur:

- $C(P) > d_j - p_j$, (i.e., **case (6)**),
- $d_j - p_j \geq C(P) > d_i - p_i$, (i.e., **case (7)**),
- $d_i - p_i \geq C(P) > d_i - p_i - p_j$, (i.e., **case (1)**),
- $d_i - p_i - p_j \geq C(P)$, (i.e., **case (3)**).

Note that once d_i, d_j, p_i and p_j are given, only one of the 20 cases, A1, A2, . . . , J1, and J2 is true. For convenience, we denote $V_k(i, j)$ (or $V_k(j, i)$) as the kth *critical interval* of subschedule J_iJ_j (or J_jJ_i), where $k \leq 5$. Then the relationship matrix can be constructed easily as shown in the following example.

EXAMPLE 1: Let us consider a weighted total earliness/tardiness cost problem with five jobs having data as follows:

$$p_1 = 2, \quad p_2 = 6, \quad p_3 = 8, \quad p_4 = 12, \quad \text{and} \quad p_5 = 20;$$

$$d_1 = 25, \quad d_2 = 12, \quad d_3 = 40, \quad d_4 = 30, \quad \text{and} \quad d_5 = 23;$$

$$\alpha = 1 \quad \text{and} \quad \beta = 1.$$

$$M = \begin{array}{|c|c|c|c|c|c|}
\hline
 & J_1 & J_2 & J_3 & J_4 & J_5 \\
\hline
J_1 & & {}_6[24,\infty)_* \\ {}_{11}[21,23]_* \\ {}_{10}[\phi]_* \\ {}_9[\phi]_* \\ {}_{18}[\phi]_* & {}_6[33,\infty)_* \\ {}_7[31,32]_* \\ {}_4[24,30]_* \\ {}_1[16,23]^{\infty}_{max(28,C(P)+10)} \\ {}_3[0,15]^{\infty}_{28} & {}_6[24,\infty)_* \\ {}_{11}[19,23]_* \\ {}_{12}[17,18]_* \\ {}_{19}[12,16]^{\infty}_{30} \\ {}_{20}[0,11]^{\infty}_{30} & {}_6[24,\infty)_* \\ {}_{11}[14,23]_* \\ {}_9[\phi]_* \\ {}_{18}[\phi]_* \\
\hline
J_2 & {}_6[\phi]_* \\ {}_{11}[18,21]_* \\ {}_{10}[7,17]_* \\ {}_9[5,6]_* \\ {}_{18}[0,4]_* & & {}_6[33,\infty)_* \\ {}_7[27,32]_* \\ {}_4[7,26]_* \\ {}_1[0,6]_* \\ {}_3[\phi]_* & {}_6[19,\infty)_* \\ {}_7[13,18]_* \\ {}_4[7,12]_* \\ {}_1[0,6]_* \\ {}_3[\phi]_* & {}_6[7,\infty)_* \\ {}_{11}[4,6]_* \\ {}_{12}[0,3]_* \\ {}_{19}[\phi]_* \\ {}_{20}[\phi]_* \\
\hline
J_3 & {}_6[\phi]_* \\ {}_7[\phi]_* \\ {}_4[\phi]_* \\ {}_1[16,23]^{28}_{C(P)+10} \\ {}_3[0,15]^{28}_{C(P)+10} & {}_6[\phi]_* \\ {}_7[\phi]_* \\ {}_4[\phi]_* \\ {}_1[\phi]_* \\ {}_3[\phi]_* & & {}_6[33,\infty)_* \\ {}_{11}[30,32]_* \\ {}_{10}[\phi]_* \\ {}_9[\phi]_* \\ {}_{18}[\phi]_* & {}_6[33,\infty)_* \\ {}_{11}[26,32]_* \\ {}_{10}[\phi]_* \\ {}_9[\phi]_* \\ {}_{18}[\phi]_* \\
\hline
J_4 & {}_6[\phi]_* \\ {}_{11}[\phi]_* \\ {}_{12}[17,18]_* \\ {}_{19}[12,16]_* \\ {}_{20}[0,11]_* & {}_6[\phi]_* \\ {}_7[\phi]_* \\ {}_4[\phi]_* \\ {}_1[\phi]_* \\ {}_3[\phi]_* & {}_6[\phi]_* \\ {}_{11}[21,30]_* \\ {}_{10}[19,20]_* \\ {}_9[11,18]_* \\ {}_{18}[0,10]_* & & {}_6[19,\infty)_* \\ {}_{11}[14,18]_* \\ {}_{12}[\phi]_* \\ {}_{13}[\phi]_* \\ {}_{20}[\phi]_* \\
\hline
J_5 & {}_6[\phi]_* \\ {}_{11}[4,14]_* \\ {}_9[2,3]_* \\ {}_{18}[0,1]_* & {}_6[\phi]_* \\ {}_{11}[\phi]_* \\ {}_{12}[\phi]_* \\ {}_{19}[\phi]_* \\ {}_{20}[\phi]_* & {}_6[\phi]_* \\ {}_{11}[13,26]_* \\ {}_{10}[4,12]_* \\ {}_9[0,3]_* \\ {}_{18}[\phi]_* & {}_6[\phi]_* \\ {}_{11}[4,14]_* \\ {}_{12}[0,3]_* \\ {}_{13}[\phi]_* \\ {}_{20}[\phi]_* & \\
\hline
\end{array},$$

Figure 1. The relationship matrix.

From cases A1, A2, ..., J1, and J2, we can derive the *critical interval set* for each subschedule J_iJ_j, where $i \neq j$. Hence, the *relationship matrix* is shown in Figure 1, where $M = [m_{ij}]_{5\times5}$, $m_{ij} = \{V_1(i, j), V_2(i, j), V_3(i, j), V_4(i, j), V_5(i, j)\}$, $\varnothing$ is an empty set, and ${}_x[a, b]^z_y$ indicates that if $a \leq C(P) \leq b$ and $z \geq S(Q) \geq y$ [if $z = \infty$, then this equation will become $z > S(Q) \geq y$], then we use case (x) to schedule J_i and J_j; ${}_x[a, b]_*$ indicates that if $a \leq C(P) \leq b$ and $\infty > S(Q) \geq C(P) + p_i + p_j$, then we use case (x) to schedule J_i and J_j. As an illustration, let us examine pair $m_{35} = \{V_1(3, 5), V_2(3, 5), V_3(3, 5), V_4(3, 5), V_5(3, 5)\}$ and $m_{53} = \{V_1(5, 3), V_2(5, 3), V_3(5, 3), V_4(5, 3), V_5(5, 3)\}$, where $i = 3$ and $j = 5$. Since $p_3 = 8$, $p_5 = 20$, $d_3 = 40$, and $d_5 = 23$, we have $d_i - p_i = 32 > d_i - p_i - p_j = 12 > d_j - p_j = 3 > d_j - p_i - p_j = -5$ and $\alpha \leq \beta$, and, therefore, case C2 is qualified. Then, m_{35} and m_{53} have the following critical intervals, respectively:

- If $C(P) > 32$, then case (6) is adopted to schedule J_3 and J_5 when they are arranged adjacently. From case (6), J_3 should precede J_5; hence, $V_1(3, 5) =_6 [33, \infty)_*$ and $V_1(5, 3) =_6 [\varnothing]_*$.
- Consider $32 \geq C(P) > 12$, then case (11) is adopted to schedule J_3 and J_5 when they are arranged adjacently. From case (11), it can be seen that $\delta_{35} = -(d_3 -$

$C(P) - p_3) + (C(P) + p_5 - d_3) = 2C(P) - 52$. Hence, if $C(P) \geq 26$, then $\delta_{35} \geq 0$; if $C(P) \leq 26$, then $\delta_{35} \leq 0$. Therefore, when $32 \geq C(P) \geq 26$, J_3 should precede J_5, and when $26 \geq C(P) \geq 13$, J_5 should precede J_3. Therefore, $V_2(3, 5) =_{11} [26, 32]_*$ and $V_2(5, 3) =_{11} [13, 26]_*$.

- If $12 \geq C(P) > 3$, then case (10) is adopted to schedule J_3 and J_5 when they are arranged adjacently. From case (10), J_5 should precede J_3, hence, $V_3(3, 5) =_{10} [\varnothing]_*$ and $V_3(5, 3) =_{10} [4, 12]_*$.
- Consider $3 \geq C(P) > -5$, then case (9) is adopted to schedule J_3 and J_5 when they are arranged adjacently. From case (9), when $3 \geq C(P) > -5$, J_5 should precede J_3. Since all jobs are assumed to be available for processing at time 0, then $V_4(3, 5) =_9 [\varnothing]_*$ and $V_4(5, 3) =_9 [0, 3]_*$.
- If $-5 \geq C(P)$, then case (18) is adopted to schedule J_3 and J_5 when they are arranged adjacently. Since all jobs are available for processing at time 0, $C(P)$ must be a nonnegative integer and it cannot be smaller than 0. Hence, $V_5(3, 5) =_{18} [\varnothing]_*$ and $V_5(5, 3) =_{18} [\varnothing]_*$.

Every pair of m_{ij} and m_{ji} can be obtained by similar processes. This example will be continued in Example 2 to complete the schedule. □

4. BRANCHING SCHEME

In this section, a *branching scheme* based on the relationship matrix is discussed. In the beginning, $C(P) = 0$, all possible initial jobs are determined using the following rule: Job J_i is a possible initial job if there exists a J_j such that J_i could precede J_j. That is, there exists k such that $0 \in V_k(i, j)$. Define $L_0 = \{\text{all possible initial jobs}\}$. A tree corresponding to the initial job J_i can be constructed. Apply the relationship matrix to determine all the possible successors of J_i, and collect them into a set L_i. Select a job J_j from L_i. Apply the relationship matrix to determine all the possible successors of J_j, and collect them into a set L_{ij}.

If $L_{ij} = \varnothing$, then remove J_j from L_i and move up one level. Select another job from L_i if $L_i \neq \varnothing$, and repeat the same process. If $L_i = \varnothing$ after J_j has been removed, then remove J_i from L_0 and move up one level. Then select another possible initial job from L_0.

If $L_{ij} \neq \varnothing$, select a job J_k from L_{ij}. Apply the relationship matrix to determine all the possible successors of J_k, collect them into a set L_{ijk}, and repeat the same process.

Note that if a subschedule is not feasible, then it cannot reach the lowest level of the tree; whereas if it is feasible, then it will reach the lowest level of the tree, in which case we should move up one level and repeat the above-described process. Using these processes recursively, we can find feasible schedules for the first initial job, and likewise for all possible initial jobs. We then select the one with the smallest cost. In summary, the steps can be described as follows.

STEP 1: Check row J_l, $l = 1, 2, \ldots, n$, in the relationship matrix.

If there is a critical interval such that it contains 0 then J_l is a possible initial job.
Let $L_0 = \{\text{all possible initial jobs}\}$.

STEP 2: If $L_0 = \varnothing$ then exit.
Else select a job J_i from L_0.

STEP 3: Let $C(P) = 0$.

Check row J_i in the matrix.
For $j = 1, 2, \ldots, n, j \neq i$
{
If a critical interval $V_l(i, j)$ $(l = 1, 2, \ldots, 5)$ contains $C(P) = 0$ then

i. check row J_j.
ii. If a critical interval contains $C(P) + p_i = p_i$ in row J_j then add J_j to L_i.

}
If there is no critical interval for all j then remove J_i from L_0 and go to Step 2.

STEP 4: If $L_i = \varnothing$ then remove J_i from L_0 and go to Step 2.

Else select a job J_j from L_i.

STEP 5: Check element m_{ij} in the matrix and check to which critical interval it belongs.

Then, a schedule pattern can be determined and we can obtain s_i, C_i, s_j and C_j.

STEP 6: let $C(P) = C_i$.

Check row J_j in the matrix.
For $k = 1, 2, \ldots, n, k \neq i, k \neq j$
{

If a critical interval $V_l(j, k)$ $(l = 1, 2, \ldots, 5)$ contains $C(P) = C_i$ then
i. check row J_k.
ii. If a critical interval contains $C(P) + p_j$ in row J_k then add J_k to L_{ij}.

}
If there is no critical interval for all k then remove J_j from L_i and go to Step 4.

STEP 7: If $L_{ij} = \varnothing$ then remove J_j from L_i and go to Step 4.

Else select a job J_k from L_{ij}.

Note that Step 7 is similar to Step 4 and hence Step 8 will be similar to Step 5. Step 9 will be similar to Step 6, and so on. Three steps are added when a level is added. If a schedule reaches the lowest level of the tree, then we call it a feasible schedule and calculate its cost. When all feasible schedules are found, then we select the one with the smallest cost.

EXAMPLE 2: Let us now consider the five-job problem illustrated in Example 1 again, but this time using the *branching rule* to solve the problem. In this example, $L_0 = \{1, 2, 3, 4, 5\}$ because $V_5(1, 3) =_3 [0, 15]^{\infty}_{28}$, $V_5(2, 1) =_{18} [0, 4]_*$, $V_5(3, 1) =_3 [0, 15]^{28}_{C(P)+10}$, $V_5(4, 3) = [0, 10]_{18}$ and $V_4(5, 1) = [0, 1]_{18}$ all contain 0.

Select J_1 from L_0 as the initial job. Then $L_1 = \{3, 4\}$ since $V_5(1, 3) =_3 [0, 15]_{28}^{\infty}$ contains 0 and J_3 has a successor after time 2, $V_5(1, 4) =_{20} [0, 11]_{30}^{\infty}$ contains 0 and J_4 has a successor after time 2. Select J_3 from L_1 to form subschedule J_1J_3 and adopt case (3). Next, check situations (3a), (3b), and (3c), which should indicate that the schedule pattern is situations (ia), (ib), and part of (ic). Next, consider situation (ia). When $S(Q) \geq 40$, then we have $s_1 = 23$, $C_1 = 25$, $s_3 = 32$, and $C_3 = 40$. Now, $C(P) = 25$, and $L_{13} = \emptyset$, since no interval of m_{32}, m_{34}, and m_{35} contains 25. Next, consider situation (ib). When $40 > S(Q) \geq 33$, then we have $s_1 = 23$, $C_1 = 25$, $s_3 = S(Q) - 8$, and $C_3 = S(Q)$. Now, $C(P) = 25$, and $L_{13} = \emptyset$ since no interval of m_{32}, m_{34}, and m_{35} contains 25. Next, consider situation (ic). Observe $V_5(3, 1)$ and the condition of (ic), $S(Q)$ must satisfy $33 > S(Q) \geq 28$ and the schedule is $s_1 = S(Q) - 10$, $C_1 = S(Q) - 8$, $s_3 = S(Q) - 8$, and $C_3 = S(Q)$. Now $C(P) = S(Q) - 8$ and hence $25 > C(P) \geq 20$ and $L_{13} = \emptyset$, since no interval of m_{32}, m_{34}, and m_{35} could contains $C(P)$. Remove J_3 from L_1, move up one level (then $C(P) = 0$ again), select another job J_4 from L_1 to form subschedule J_1J_4, and adopt situations (20a2), according to which the schedule pattern should be case (va). Hence, we have $s_1 = 16$, $C_1 = 18$, $s_4 = 18$, and $C_4 = 30$. Then, $C(P) = 18$, and $L_{14} = \{J_3\}$ because $V_4(4, 3) =_9 [11, 18]_*$ contains 18 and J_3 has a successor after time $C(P) + p_4 = 30$, $V_2(4, 5) = [14, 18]_{11}$ contain 18, but J_5 has no successor after time 30, and no interval of m_{42} contains 18. Take J_3 from L_{14} to form subschedule $J_1J_4J_3$ and adopt case (9). Check situations (9a), (9b), and (9c), the schedule pattern should be situations (via), (vib), and (vic). Consider situation (via). When $S(Q) \geq 40$, then we have $s_4 = 18$, $C_4 = 30$, $s_3 = 32$, and $C_3 = 40$. Now $C(P) = 30$ and $L_{143} = \emptyset$ because no interval of m_{32} contains 30 and although $V_2(3, 5)$ contains 30, no interval of m_{52} contains time greater than $30 + p_3 = 38$. Consider situation (vib). When $40 > S(Q) \geq 38$, then we have $s_4 = 18$, $C_4 = 30$, $s_3 = S(Q) - 8$, and $C_3 = S(Q)$. Now $C(P) = 30$ and $L_{143} = \emptyset$ because no interval of m_{32} contains 30 and although $V_2(3, 5)$ contains 30, no interval of m_{52} contains time greater than $30 + p_3 = 38$. Now consider situation (vic). The $S(Q)$ must satisfy $38 > S(Q) \geq C(P) + p_i + p_j = 18 + 8 + 12 = 38$, and therefore situation (vic) must be discarded.

Remove J_3 from L_{14}; then $L_{14} = \emptyset$. Move up one level, remove J_4 from L_1; then $L_1 = \emptyset$; remove J_1 from L_0, move up one level, and select another job J_2 from L_0 as an initial job.

If J_2 is the initial job, following the above processes, we find seven feasible schedules:

1. $J_2J_1J_4J_3J_5$ with $s_2 = 6$, $C_2 = 12$, $s_1 = 16$, $C_1 = 18$, $s_4 = 18$, $C_4 = 30$, $s_3 = 30$, $C_3 = 38$, $s_5 = 38$, and $C_5 = 58$, at cost 44.
2. $J_2J_3J_1J_4J_5$ with $s_2 = 6$, $C_2 = 12$, $s_3 = 14$, $C_3 = 22$, $s_1 = 22$, $C_1 = 24$, $s_4 = 24$, $C_4 = 36$, $s_5 = 36$, and $C_5 = 56$, at cost 58.
3. $J_2J_3J_1J_4J_5$ with $s_2 = 6$, $C_2 = 12$, $s_3 = 15$, $C_3 = 23$, $s_1 = 23$, $C_1 = 25$, $s_4 = 25$, $C_4 = 37$, $s_5 = 37$, and $C_5 = 57$, at cost 58.
4. $J_2J_4J_1J_3J_5$ with $s_2 = 6$, $C_2 = 12$, $s_4 = 12$, $C_4 = 24$, $s_1 = 24$, $C_1 = 26$, $s_3 = 26$, $C_3 = 34$, $s_5 = 34$, and $C_5 = 54$, at cost 44.
5. $J_2J_4J_1J_5J_3$ with $s_2 = 5$, $C_2 = 11$, $s_4 = 11$, $C_4 = 23$, $s_1 = 23$, $C_1 = 25$, $s_5 = 25$, $C_5 = 45$, $s_3 = 45$, and $C_3 = 53$, at cost 43.
6. $J_2J_4J_1J_3J_5$ with $s_2 = 5$, $C_2 = 11$, $s_4 = 11$, $C_4 = 23$, $s_3 = 23$, $C_3 = 31$, $s_1 = 31$, $C_1 = 33$, $s_5 = 33$, and $C_5 = 53$, at cost 55.
7. $J_2J_5J_1J_4J_3$ with $s_2 = 0$, $C_2 = 6$, $s_5 = 6$, $C_5 = 26$, $s_1 = 26$, $C_1 = 28$, $s_4 = 28$, $C_4 = 40$, $s_3 = 40$, and $C_3 = 48$, at cost 30.

If J_3 is the initial job, following the above processes, we find one feasible schedule $J_3J_1J_2J_4J_5$ with $s_3 = 15$, $C_3 = 23$, $s_1 = 23$, $C_1 = 25$, $s_2 = 25$, $C_2 = 31$, $s_4 = 31$, $C_4 = 43$, $s_5 = 43$ and $C_5 = 63$, at cost 89.

If J_4 is the initial job, following the above processes, we find three feasible schedules:

1. $J_4J_1J_2J_3J_5$ with $s_4 = 11$, $C_4 = 23$, $s_1 = 23$, $C_1 = 25$, $s_2 = 25$, $C_2 = 31$, $s_3 = 31$, $C_3 = 39$, $s_5 = 39$, and $C_5 = 59$, at cost 63.
2. $J_4J_3J_1J_2J_5$ with $s_4 = 4$, $C_4 = 16$, $s_3 = 16$, $C_3 = 24$, $s_1 = 24$, $C_1 = 26$, $s_2 = 26$, $C_2 = 32$, $s_5 = 32$, and $C_5 = 52$, at cost 80.
3. $J_4J_3J_1J_2J_5$ with $s_4 = 3$, $C_4 = 15$, $s_3 = 15$, $C_3 = 23$, $s_1 = 23$, $C_1 = 25$, $s_2 = 25$, $C_2 = 31$, $s_5 = 31$, and $C_5 = 51$, at cost 79.

If J_5 is the initial job, following the above processes, we find five feasible schedules:

1. $J_5J_1J_2J_3J_4$ with $s_5 = 3$, $C_5 = 23$, $s_1 = 23$, $C_1 = 25$, $s_2 = 25$, $C_2 = 31$, $s_3 = 31$, $C_3 = 39$, $s_4 = 39$, and $C_4 = 51$, at cost 41.
2. $J_5J_1J_2J_3J_4$ with $s_5 = 2$, $C_5 = 22$, $s_1 = 22$, $C_1 = 24$, $s_2 = 24$, $C_2 = 30$, $s_3 = 30$, $C_3 = 38$, $s_4 = 38$, and $C_4 = 50$, at cost 42.
3. $J_5J_1J_2J_4J_3$ with $s_5 = 2$, $C_5 = 22$, $s_1 = 22$, $C_1 = 24$, $s_2 = 24$, $C_2 = 30$, $s_4 = 30$, $C_4 = 42$, $s_3 = 42$, and $C_3 = 50$, at cost 42.
4. $J_5J_3J_1J_2J_4$ with $s_5 = 3$, $C_5 = 23$, $s_3 = 23$, $C_3 = 31$, $s_1 = 31$, $C_1 = 33$, $s_2 = 33$, $C_2 = 39$, $s_4 = 39$, and $C_4 = 51$, at cost 65.
5. $J_5J_3J_1J_2J_4$ with $s_5 = 0$, $C_5 = 20$, $s_3 = 20$, $C_3 = 28$, $s_1 = 28$, $C_1 = 30$, $s_2 = 30$, $C_2 = 36$, $s_4 = 36$, and $C_4 = 48$, at cost 62.

Finally, there are 16 feasible schedules, the optimal one of which is $J_2J_5J_1J_4J_3$ with $s_2 = 0$, $C_2 = 6$, $s_5 = 6$, $C_5 = 26$, $s_1 = 26$, $C_1 = 28$, $s_4 = 28$, $C_4 = 40$, $s_3 = 40$, and $C_3 = 48$, having the minimum cost of 30. □

5. SIMULATION RESULTS

Let us consider a single-machine job-independent weighted earliness and tardiness penalties scheduling problem in which idle time inserted between two adjacent jobs is permitted. We did the following simulations with similar test data to Szwarc [9].

The integer processing times were drawn from a uniform distribution in the range [1, 100]. The integer earliness penalty α and tardiness penalty β were drawn from a uniform distribution in the range [1, 10]. The due dates were generated from a uniform integer distribution in the range $[up, vp]$, where $p = \sum_{k=1}^{n} p_k$ and $0.1 \leq u \leq v \leq 0.9$.

The general impression is that more feasible schedules result where the due dates interval $[up, vp]$ is wider, which, in view of the discussion in Section 3, could be expected. A wider due dates interval $[up, vp]$ implies that considerable idle time is involved, whereas a narrower interval $[up, vp]$ implies more difficulty with job scheduling and that many adjacent jobs cannot have idle time between them. A narrow interval induces a reduction in the schedule pattern of those adjacent jobs, resulting in fewer feasible schedules.

To verify the above descriptions and see the effects of the due dates interval $[up, vp]$, 20-job problems were tested with 10 combinations of u and v, that is, $(u, v) = (0.1, 0.9)$, $(0.1, 0.7)$, $(0.1, 0.5)$, $(0.1, 0.3)$, $(0.3, 0.9)$, $(0.3, 0.7)$, $(0.3, 0.5)$, $(0.5, 0.9)$, $(0.5, 0.7)$, and $(0.7, 0.9)$. The results are as follows.

In the case $u = 0.1$ and $v = 0.9$ (the worst case in Szwarc [9]), 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 9{,}999$	$10^4 \sim 99{,}999$	$10^5 \sim 399{,}999$	$4 \times 10^5 \sim 999{,}999$	$10^6 \sim 10^8$
Problems	80	109	103	85	123

with a maximum 83,197,565 and an average 4,727,375.

In the case $u = 0.1$ and $v = 0.7$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 9{,}999$	$10^4 \sim 99{,}999$	$10^5 \sim 299{,}999$	$3 \times 10^5 \sim 999{,}999$	$10^6 \sim 2 \times 10^7$
Problems	83	123	109	108	77

with a maximum 16,673,453 and an average 2,889,747.

In the case $u = 0.1$ and $v = 0.5$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 99$	$100 \sim 999$	$10^3 \sim 4{,}999$	$5 \times 10^3 \sim 199{,}999$	$2 \times 10^5 \sim 2 \times 10^6$
Problems	62	85	89	130	134

with a maximum 1,445,789 and an average 124,224.

In the case $u = 0.1$ and $v = 0.3$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 9$	$10 \sim 49$	$50 \sim 199$	$200 \sim 499$	$500 \sim 7 \times 10^4$
Problems	70	123	88	86	133

with a maximum 69,937 and an average 4023.

In the case $u = 0.3$ and $v = 0.9$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 9{,}999$	$10^4 \sim 99{,}999$	$10^5 \sim 299{,}999$	$3 \times 10^5 \sim 999{,}999$	$10^6 \sim 2 \times 10^7$
Problems	96	113	116	112	63

with a maximum 19,456,345 and an average 2,244,305.

In the case $u = 0.3$ and $v = 0.7$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 99$	$100 \sim 999$	$10^3 \sim 4{,}999$	$5 \times 10^3 \sim 199{,}999$	$2 \times 10^5 \sim 3 \times 10^6$
Problems	27	43	72	211	147

with a maximum 3,114,317 and an average 199,308.

In the case $u = 0.3$ and $v = 0.5$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 999$	$10^3 \sim 1{,}999$	$2 \times 10^3 \sim 2{,}999$	$3 \times 10^3 \sim 5{,}999$	$6 \times 10^3 \sim 2 \times 10^5$
Problems	109	101	92	108	90

with a maximum 170,211 and an average 8328.

In the case $u = 0.5$ and $v = 0.9$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 999$	$10^3 \sim 9{,}999$	$10^4 \sim 29{,}999$	$3 \times 10^4 \sim 199{,}999$	$2 \times 10^5 \sim 4 \times 10^6$
Problems	64	98	107	142	89

with a maximum 3,295,456 and an average 188,441.

In the case $u = 0.5$ and $v = 0.7$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 499$	$500 \sim 999$	$10^3 \sim 1{,}999$	$2 \times 10^3 \sim 4{,}999$	$5 \times 10^3 \sim 2 \times 10^5$
Problems	93	97	98	122	90

with a maximum 125,927 and an average 5913.

In the case $u = 0.7$ and $v = 0.9$, 500 problems are scheduled for which the experimental results are as follows:

Feasible Schedules	$0 \sim 499$	$500 \sim 999$	$10^3 \sim 1{,}999$	$2 \times 10^3 \sim 4{,}999$	$5 \times 10^3 \sim 2 \times 10^5$
Problems	162	91	95	103	49

with a maximum 123,773 and an average 5,349.

As the above data shows, the branching scheme based on the relationship matrix can solve idle-time-permitted E/T problems. Indeed, the number of feasible schedules increases as the due dates interval $[up, vp]$ widens. When the interval is narrow, i.e., $[up, vp] = [0.1, 0.3]$, 73% of the problems have less than 500 feasible schedules with a maximum 69,937 and an average 4023. In this case, there are fewer feasible schedules because the flexibility is small and many of them are reduced to a no-idle-time situation. In contrast, a wide due dates interval (for example, $[up, vp] = [0.1, 0.9]$) gives greater flexibility and an increase in the number of feasible schedules, 75% of the problems having less than 10^6 feasible schedules with a maximum 83,197,565 and an average 4,727,375. If $v - u$ is fixed and u is set at several values, e.g., $[u, v] = [0.1, 0.3]$, $[0.3, 0.5]$, $[0.5, 0.7]$, or $[0.7, 0.9]$, then the experiments produce similar results. Hence, we can fix u and change only the value of v.

6. CONCLUSIONS

In this paper, a single-machine scheduling problem with unequal earliness and tardiness penalties has been considered. Idle time between two adjacent jobs was permitted, and due dates of jobs could be unequal. The dominance rules for two adjacent jobs were used to construct a relationship matrix that was used for idle-time-weighted earliness/tardiness penalty problems, especially where the due dates interval was small. The relationship matrix allowed us to produce a branching scheme to solve the idle-time-permitted E/T model problems. Simulation results showed that the procedure solved those problems and generated schedules that could not be improved by adjacent job interchanges. In fact, where idle time was not permitted, the number of schedule patterns decreased from 12 (i.e., pattern (i), (ii), . . . , (xii)) to 4 (i.e., pattern (iii), (iv), (ix), and (x)). Hence, the number of feasible schedules also decreased. The same procedure can be used to solve problems like the above.

Problems in which earliness and tardiness weights were job-independent were considered. The results we have gathered here may be extended to the more general case in which tardiness and earliness weights are job-dependent. The extension from a single-machine to multiple-machines is also noteworthy. It would be of further interest to ascertain whether our results could be extended to the time window case.

ACKNOWLEDGMENTS

We wish to thank Professor W. Szwarc, Associate Editor, and anonymous referees for their many suggestions, which have greatly helped to improve our paper. This study was supported, in part, by the National Science Council, Republic of China, under contract number: NSC 90-2213-E-009-106.

REFERENCES

[1] T.S. Abdul-Razaq and C.N. Potts, Dynamic programming state-space relaxation for single machine scheduling, J Oper Res Soc 39 (1988), 141–152.
[2] K.R. Baker and G.D. Scudder, Sequencing with earliness and tardiness penalties: A review, Oper Res 38 (1990), 22–36.
[3] J.S. Davis and J.J. Kanet, Single-machine scheduling with early and tardy completion costs, Nav Res Logistics 40 (1993), 85–101.
[4] M.R. Garey, R.E. Tarjan, and G.T. Wilfong, One-processor scheduling with symmetric earliness and tardiness penalties, Math Oper Res 13 (1988), 330–348.
[5] Y.C. Kim and C.A. Yano, Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates, Nav Res Logistics 41 (1994), 913–933.
[6] C. Koulamas, Single-machine scheduling with time windows and earliness/tardiness penalties, Eur J Oper Res 91 (1996), 190–202.
[7] P.S. Ow and T.E. Morton, The single machine early/tardy problems, Manage Sci 35 (1989), 177–191.
[8] M. Pinedo, Scheduling, theory, algorithm, and systems, Prentice Hall, Englewood Cliffs, NJ, 1996.
[9] W. Szwarc, Adjacent orderings in single-machine scheduling with earliness and tardiness penalties, Nav Res Logistics 40 (1993), 229–243.
[10] W. Szwarc and S.K. Mukhopadhyay, Optimal timing schedules in earliness–tardiness single machine sequencing, Nav Res Logistics 42 (1995), 1109–1114.
[11] J.A. Hoogeveen and S.L. Van de Velde, A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time, INFORMS J Comput 8 (1996), 402–412.