Earliness and Tardiness Single Machine Scheduling with Proportional Weights

Wlodzimierz Szwarc
and
Samar K. Mukhopadhyay

School of Business Administration
University of Wisconsin-Milwaukee
P. O. Box 742
Milwaukee, WI 53201

Abstract

This paper presents the first efficient method to solve an earliness-tardiness single machine n job scheduling model with idle times permitted. In this model the earliness and tardiness penalties are proportional to the processing times of the jobs. A two stage decomposition process builds the optimal job arrangement as a sequence of single and multijob blocks. The arrangement of the jobs within each multijob block is unspecified since it depends on the start time of this block. This process is followed by a procedure that drastically reduces the number of candidate optimal n job sequences. Finally an available optimal timing algorithm is recommended and implemented to select the best schedule among those sequences. The solution procedure tested on a PC on 400 examples for n=40 and 50 proves to be very fast.

1. Introduction

Just-in-time production environment is being increasingly popular in manufacturing. The associated advantages like reduction in inventory are the prime motivation for implementing such systems. In this environment, jobs should be completed as close to the due date as possible. Penalties are associated with both tardiness and earliness of the jobs. A very relevant objective in this type of situation is to minimize the sum of the earliness and tardiness penalties of all jobs. Conceptually, the inclusion of earliness in the objective means that there could be inserted idle time in the schedule so that jobs are scheduled to finish close to their due dates. This non-regular measure of performance makes the problem very hard to solve.

This paper deals with a single machine n job earliness tardiness model (later called ET model). Garey et al. [2] consider a case with equal penalties for earliness and tardiness. They prove the NP-completeness of the ET model and also present an $O(n \log n)$ algorithm to find the best schedule for a given sequence. Yano and Kim [7] and Davis and Kanet [3] developed $O(n^3)$ optimal timing algorithms for arbitrary earliness and tardiness penalties.

This paper presents the first efficient method to solve an ET model with idle times permitted. This model assumes that the earliness and tardiness penalties of the jobs are proportional to their processing times. Yano and Kim [7] outlined a branch and bound method and used it to evaluate four heuristic procedures tested on 100 problems of the size ≤ 40.

The tardiness version of this model has been earlier explored by Arkin and Roundy [1] and Szwarc and Liu [5].

The solution procedure of the ET model rests on the following basic findings.

1. The ET model is decomposable into blocks as in the tardiness version. What is remarkable is that the arrangement of those blocks <u>does not change</u> when their start times are shifted. What <u>does</u> change is the arrangement of jobs within each block.
2. The precedence relations on which the decomposability is based remain valid even when idle times are permitted.

Based on these findings, we determine an initial list of potentially optimal sequences. Taking advantage of certain precedence relations, we develop a procedure that drastically reduces the initial list to a manageable number of sequences. Finally we apply a recently developed optimal timing algorithm of [6] to every potential candidate in the reduced list in order to select the optimal schedule of our ET model. (The optimal timing algorithm of [3] can also be applied.) Our solution procedure tested on 400 problems on a PC proves to be quite efficient.

2. The Precedence Matrix and Problem Decomposition (Stage I)

Throughout the paper, we use the following notations:

$I=(1,2,\ldots,n)$ - the set of n jobs,

p_k - processing time of job k, $k \in I$,

d_k - the due date of job k,

C_k - completion time of job k,

T_k - $\max(0, C_k-d_k)$ - the tardiness of job k,

E_k - $\max(0, d_k-C_k)$ - the earliness of job k,

αp_k - earliness penalty for job k, $\alpha>0$,

βp_k - tardiness penalty for job k, $\beta>0$.

For convenience, we arrange the jobs such that

$$p_1 \geq p_2 \geq \dots \geq p_n, \quad \text{and} \quad i<j \text{ if } p_i = p_j \text{ and } d_i \leq d_j.$$

The problem can be formally stated as: find a schedule S that minimizes

$$f(S) = \sum_{k=1}^{n} (\alpha p_k E_k + \beta p_k T_k). \tag{1}$$

Consider two schedules $S_1 = \sigma ij\pi$ and $S_2 = \sigma ji\pi$ where $i,j \in I$, $t = \sum_{k\in\sigma} p_k$ and σ and π are two disjoint subsequences of the remaining n-2 jobs. Then $\Delta_{ij}(t)=f(S_2)-f(S_1)$ is the cost of interchanging adjacent jobs i and j if their processing starts at t. Assume that there is <u>no idle time</u> between jobs i and j in S_1 and S_2. Then $\Delta_{ij}(t)$ can be presented in the following compact form:

$$\Delta_{ij}(t) = \begin{bmatrix}\alpha p_j\\ \beta p_j\end{bmatrix}[t+p_j-d_j] - \begin{bmatrix}\alpha p_i\\ \beta p_i\end{bmatrix}[t+p_i-d_i] +$$

$$+ \begin{bmatrix}\alpha p_i\\ \beta p_i\end{bmatrix}[t+p_i+p_j-d_i] - \begin{bmatrix}\alpha p_j\\ \beta p_j\end{bmatrix}[t+p_i+p_j-d_j], \tag{2}$$

once we use the symbol $\begin{bmatrix}x\\ y\end{bmatrix}[z]$ to denote the following function:

$$\begin{bmatrix}x\\ y\end{bmatrix}[z] = \begin{cases} -xz & \text{if } z<0 \\ yz & \text{if } z\geq 0 \end{cases}$$

This cost $\Delta_{ij}(t)$ does not depend on how the jobs are arranged in σ and π but depends on start time t of the pair. Let us define a value t_{ij} such that

$$t_{ij} = d_i - p_i - p_j + \frac{d_i-d_j}{p_i-p_j}\, p_j, \quad p_i \neq p_j. \tag{3}$$

Let $\Delta'_{ij}(t)$ be a special case of $\Delta_{ij}(t)$ where $\alpha=0$. The following result ties our model to the tardiness model of [5] (the proof is given in the Appendix).

<u>Lemma 1</u>: $\Delta_{ij}(t) = \frac{\alpha+\beta}{\beta}\Delta'_{ij}(t)$.

Reference [5] establishes the following property that specifies the optimal arrangement of adjacent jobs i and j, i<j, where there is no idle time between those jobs.

<u>Property 1</u>: (a) If $d_i-d_j \leq 0$ then $i\to j$ (i precedes j), $\forall t$.

(b) If $d_i-d_j \geq p_i-p_j$ then $j\to i$, $\forall t$.

(c) If $0 < d_i-d_j < p_i-p_j$ then $j \rightarrow i$ for $t<t_{ij}$ and $i \rightarrow j$ for $t \geq t_{ij}$.

where t_{ij} is defined by (3).

If $p_i=p_j$ and $d_i=d_j$ then $i \rightarrow j$, $\forall t$ because case a is checked prior to case b. The orderings for cases (a) and (b) are unconditional and will be called global orderings. The ordering of case (c) is conditional since it depends on t. This ordering changes direction for $t=t_{ij}$.

By Lemma 1, sgn $(\Delta_{ij}(t))$ = sgn $(\Delta'_{ij}(t))$, $\forall t$. Then Property 1 is valid for our model, if there is no idle time between jobs i and j.

Yano and Kim [7] also established ordering conditions (a) and (b) of Property 1. The scope of their conditions is limited to local pairs of adjacent jobs.

Next we show that Property 1 remains in force in our ET model by proving the following basic result.

Property 2: Property 1 is valid when idle times are permitted.

Proof: Let us first utilize two simple observations from [6] regarding any ET model.

(I) Consider a schedule where there is idle time between adjacent jobs r and s. If r is the earlier job and $d_r>d_s$, then one can find a better schedule without idle time between r and s by starting r later or (and) starting s earlier (by considering cases $C_s \leq d_s$ and $C_s>d_s$ where C_s is the completion time of job s in the schedule).

(II) If there is idle time in an optimal schedule between adjacent jobs r and s where r is the earlier job, then $d_s-d_r>p_s$.

To prove the theorem, consider an optimal schedule S where there is idle time between jobs i and j, i<j. Examine all three cases of Property 1.

Case (a): $d_i \leq d_j$: According to the optimality of S and observation I, j cannot precede i in S. Hence i must precede j.

Case (b): $d_i-d_j \geq p_i-p_j$: Then $d_i > d_j$. Again the optimality of S and observation I imply that j precedes i.

Case (c): $0 < d_i-d_j < p_i-p_j$: Due to $d_i > d_j$, the assumptions about S and observation I, job j precedes i. In view of observation II, condition $d_s-d_r>p_s$ reads as $d_i-d_j > p_i$, which is inconsistent with Case (c). Hence this case is ruled out. QED.

Since Property 1 of [5] is valid for our model we take advantage of some findings of [5] derived from Property 1 in order to decompose our problem. First define an upper triangular precedence matrix T for all $1 \le i < j \le n$ by placing the symbols in cell (i,j): "-" for case (a) of Property 1, "+" for case (b), and "t_{ij}" for case (c).

For illustration consider the following:

Example 1

$p_1,\ldots,p_{20}$=:	96, 93, 93, 89, 73, 70, 53, 44, 42, 35, 34, 32, 27, 27, 17, 14, 11, 10, 4, 3.
$d_1,\ldots,d_{20}$=:	411, 392, 463, 480, 246, 364, 493, 116, 93, 255, 605, 261, 134, 471, 209, 395, 340, 523, 343, 518.

The precedence matrix T is given in Figure 1.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
96)	++	--	--	++	++	--	++	++	++	--	++	++	--	++	304	313	--	314	--
	189)	--	--	++	++	--	++	++	++	--	++	++	--	++	--	295	--	297	--
		282)	--	++	++	--	++	++	++	--	++	++	--	++	368	++	--	++	--
			371)	++	++	--	++	++	++	--	++	++	368	++	++	++	--	++	--
				444)	--	--	++	++	--	--	--	++	--	167	--	--	--	--	--
					514)	--	++	++	++	--	++	++	--	++	--	287	--	291	--
						567)	++	++	++	--	++	++	436	++	++	++	--	++	--
							611)	++	--	--	--	--	--	--	--	--	--	--	--
								653)	--	--	--	--	--	--	--	--	--	--	--
									688)	--	--	++	--	++	--	--	--	--	--
										722)	++	++	++	++	++	++	++	++	++
											754)	++	--	++	--	--	--	--	--
												781)	--	--	--	--	--	--	--
													808)	++	++	++	--	++	--
														825)	--	--	--	--	--
															839)	++	--	++	--
																850)	--	--	--
																	860)	++	512
																		864)	--
																			867)

Figure 1: Precedence Matrix of Example 1

The numbers in cells (j,j), j=1,...,20, indicate $\sum_{k=1}^{j} p_k$. According to [5], the jobs can be divided into mutually exclusive blocks by defining a set Ω of cells (i,j) of T marked by "t_{ij}". Divide this set into mutually exclusive subsets $\Omega_1, \Omega_2, \ldots$ by using the following rule: once (i,j) is assigned to Ω_k, then (i,u) and (u,j) also belong to Ω_k. Next define a set of jobs B_k, called a multijob block, as

$$B_k = \{(i) \cup (j) \mid (i,j) \in \Omega_k\}.$$

Those blocks are shown to be mutually exclusive.

B=(k) is a single job block if all entries of T are "+" or "-" in row and column k.

In our example, $\Omega_1=\{(1,16),(1,17),(1,19),(2,17),(2,19),(3,16),(6,17),(6,19)\}$, $\Omega_2=\{(4,14),(7,14)\}$, $\Omega_3=\{(5,15)\}$, $\Omega_4=(18,20)$. Hence the list of the multijob and single job blocks is $B_1=(1,2,3,6,16,17,19)$, $B_2=(4,7,14)$, $B_3=(5,15)$, $B_4=(18,20)$, $B_5=(8)$, $B_6=(9)$, $B_7=(10)$, $B_8=(11)$, $B_9=(12)$, $B_{10}=(13)$.

To arrange the blocks we utilize the following powerful property of [5]: If <u>some</u> job of block B_u globally precedes <u>some</u> job of block B_v then <u>every</u> job of B_u globally precedes <u>every</u> job of B_v.

According to [5], the optimal arrangement of blocks is found by sequencing the first jobs of each block in a nondecreasing order of their due dates (if ties occur, place the job with a smaller d_k-p_k first). The optimal block sequence of Example 1 is

9,8,13,(5,15),10,12,(1,2,3,6,16,17,19),(4,7,14),(18,20),11,

where the arrangement of the jobs within each multijob block is unspecified. However, the arrangement of blocks <u>remains unchanged since it does not depend on their start or completion times</u>. Let t_B be the sum of the processing times of all jobs preceding block B. Then t_B is the <u>earliest start time</u> of block B. The precedence submatrices T_{B_1}, T_{B_2}, T_{B_3}, and T_{B_4} are given in Figure 2.

B_1

```
          1    2    3    6   16   17   19|
-----------------------------------------|
270)                                     |
              ++   --   ++  304  313  314|   1
                   --   ++   --  295  297|   2
                        ++  368   ++   ++|   3
                             --  287  291|   6
                                  ++   ++|  16
                                       --|  17
                                         |  19
```

B_2

```
           4    7   14 |
      -----------------|
651)                   |
               --  368 |  4
                   436 |  7
                       | 14
```

B_3

```
         5   15 |
      ----------|
113)            |
       167      |  5
                | 15
```

B_4

```
        18   20 |
     ---------- |
820)            |
            512 | 18
                | 20
```

Figure 2: Submatrices of Four Blocks of Example 1

The earliest start times are listed in the left upper corner of the submatrices. In the next section we complete the decomposition process and generate candidate sequences.

3. Block Partition (Stage II) and Candidate Sequences

Now that we have found the optimal block arrangement we know the earliest start time t_B for each block B. If $t_{ij} \le t_B$, then job i unconditionally precedes j in block B since their start time $t \ge t_B$. This unconditional ordering is called local since it is valid within a particular block. If $t_{ij} \le t_B$, then $t_{ij} \le t$ and each "t_{ij}" entry is updated to a "-" to indicate that $i \to j$ unconditionally. We call it "-" updating. According to Figure 2, entries (4,14) and (7,14) of B_2 and (18,20) of B_4 are marked by "-".

A "+" updating (to indicate that $j \to i$ unconditionally in B) would also be possible if the latest completion time of block B were known at the end of Stage I. This is the case for the tardiness version where processing starts at zero and goes on uninterrupted. Hence the start and completion times of each block are fixed for this case.

Consider a "-" updated submatrix T_B of block B. This block can be further decomposed by applying the following rule.

Rule 1: If all entries of T_B are "+" in column k and "-" in row k, then k is the first job of B.

(the "+" entries symbolize global orderings while the "-" symbolize global or local orderings).

If all "t_{ij}" entries of T_B are converted into "-" then, according to [5], block B can be completely decomposed by repetitive application of Rule 1. In our example, Rule 1 splits blocks B_2 and B_4 into sequences 4,7,14 and 18,20 respectively. As a result, the second stage decomposition produces sequence

9,8,13,(5,15),10,12,(1,2,3,6,16,17,19),4,7,14,18,20,11

where the arrangement of jobs of B_3 and B_1 is still unspecified.

We will show that the list of candidate sequences can be drastically reduced even further for m-job blocks B, $m \ge 3$, once some entries are "+" or "-" in their respective submatrices. Based on Property 1 and Rule 1 we offer a procedure that considerably reduces the number of candidate sequences.

1. For each job $r \in B$ make a list of all possible immediate successors $s \in B$ where, according to T_B, r precedes s unconditionally or conditionally. Formally s is a successor of r if

(a) For s<r, the entry of (s,r) is "+" or "t_{sr}".

(b) For s>r, the entry of (r,s) is "-" or "t_{rs}".

2. Generate all possible m job sequences of B, called B-sequences, using a branching procedure. Consider node σr (initially $\sigma=\phi$ and r is each of the m jobs of B) where σ is a sequence of jobs of B preceding r.

Before branching σr perform the following:

Routine 1:

(a) Do the "-" updating for subblock B-σr (where the jobs of σr are removed from B) assuming this subblock starts at $t_B + \sum_{k \in \sigma r} p_k$. Use Rule 1 to identify the first job, s, of the subblock. If Rule 1 is not applicable, use the branching routine (Routine 2).

(b) Terminate node σr if s is not on the successors list of r. Otherwise do the "-" updating for subblock B-σrs whose start time is $t_B + \sum_{k \in \sigma rs} p_k$.

If all "t_{ij}" entries become "-" identify via Rule 1 the m-job sequence that follows node σr and terminate node σr. Otherwise use Routine 2.

Routine 2 (Branching): Branch node σr by considering all possible successors s of r. Terminate node σrs if r>s and $t_{sr} \le t_B + \sum_{k \in \sigma} p_k$.

Relax Rule 1 as follows: We say that $k \in B$ is the semi-first job of B if all entries of T_B are "+" in column k and "t_{ij}" or "-" in row k.

It is obvious that: a) k is the first job of B once block B starts at $\max_{j \in B} t_{kj}$, j>k, b) There is exactly one semi-first job of B.

Remark: Routine 1 is especially powerful when t_B is relatively close to the largest t_{ij} of submatrix T_B or at least the largest t_{kj}. Consider block B_1 of Example 1 where $t_{B_1}=270$. Then 6 is the semi-first job of B_1 and $\max_{j \in B_1} t_{6j} = 291$. For any node $r \ne 6$ where $p_r \ge 291-270 = 21$, job 6 is the first job of subblock $B_1-(r)$. This eliminates jobs 1,2,3 from the first position of the B_1-sequence since 6 is not on their successors list. Job 16 is also eliminated from the first position since 1 and 3 are its only successors. The branching routine in this example is limited to single or two job nodes σr since subblocks $B_1-\sigma r$ are completely decomposable.

Although the developed properties are based on the assumption that the machine is not idle once it starts processing at time $t_0 \geq 0$, Property 2 guarantees that the optimal schedule of our model is to be found among the candidate sequences produced by the decomposition process, Routines 1 (with the relaxed Rule 1) and 2 and an optimal timing algorithm of, say, [3] or [6].

Consider block B_1 of Example 1. The successors list is given in Table 1.

Table 1
Successor List for Block B_1

Job	Successors
1	3, 16, 17, 19
2	1, 3, 16, 17, 19
3	16
6	1, 2, 3, 16, 17, 19
16	1, 3
17	1, 2, 3, 6, 16, 19
19	1, 2, 3, 6, 16

Routines 1 and 2 generate the following four B_1-sequences:

6,2,1,17,19,3,16; 17,6,2,1,19,3,16

17,19,6,2,1,3,16; 19,6,2,1,17,3,16.

The total number of candidate sequences is 8 since the jobs of B_3 can be arranged in order 5,15 and 15,5. Without using Routines 1 and 2 we would have to consider 10,080 candidate sequences (i.e., $2! \bullet 7!$). The optimal timing algorithm of [3] or [6] for $\alpha=1$ and $\beta=5$ produces an optimal schedule where the jobs of B_3 and B_1 are arranged in order 5,15 and 17,19,6,2,1,3,16 respectively. The cost of this schedule is 507697. For this particular schedule processing starts at 0 and is uninterrupted.

The number of candidate sequences may be even further reduced under certain conditions. Suppose that stage two of the decomposition procedure produced a sequence composed of single jobs and r multijob blocks $B_1, B_2, \ldots, B_r$ where $t_{B_1} < t_{B_2} < \ldots < t_{B_r}$. Then the total number of candidate sequences is $\prod_{s=1}^{r} m_s$ where m_s is the number of B_s-sequences. We will show how a delay of the start time of a job or block affects the number of B_s-sequences of subsequent blocks.

Consider an extreme situation where block B_1 starts not earlier than

$$t_0 = t_{B_1} + \max_{1\le s\le r} [\max_{i,j\in B_s} t_{ij} - t_{B_s}].$$

Then all r blocks decompose completely and the number of candidate sequences shrinks to one since there is only one B_s-sequence for each s. Let k be the semi-first job of B_s. One can further decrease the start time t_0 by replacing $\max_{i,j\in B_s} t_{ij} = u$ with $\max_{j\in B_s} t_{kj} = v$ for those B_s where $u-v\le p_k$ and still end up with a single candidate sequence.

* *

*

Consider a version of our model where, due to technological reasons or high idle costs, the processing on the machine <u>cannot be interrupted</u>. For this version the set of candidate sequences remains the same. Note that the start time t_0 of a given candidate sequence S uniquely defines the start times for each job of S and f(S) is a convex function of t_0. On that basis we formulate a very simple algorithm to find the optimal start time of S.

1. Start with $t_0=0$.
2. Find the set E of early jobs and set T of non-early jobs.
3. Calculate $A = \alpha\sum_{k\in E} p_k$ and $B = \beta\sum_{k\in T} p_k$.
4. Stop if $B-A\ge 0$.
5. If $B-A<0$ then increase t_0 by $\min_{k\in E}(d_k-C_k)$. Go to 2.

We apply the algorithm for every candidate sequence to find the optimal schedule.

Next consider the common due date version of our model treated by Rachamadugu [4]. It is known that the jobs are processed without interruption. Formula (2) becomes

$$\Delta_{ij}(t) = \begin{bmatrix}\alpha p_j\\ \beta p_j\end{bmatrix}[t+p_j-d] - \begin{bmatrix}\alpha p_i\\ \beta p_i\end{bmatrix}[t+p_i-d] + \begin{bmatrix}\alpha p_i-\alpha p_j\\ \beta p_i-\beta p_j\end{bmatrix}[t+p_i+p_j-d].$$

It is very easy to verify that $\Delta_{ij}(t)\ge 0$, $\forall t$ for each of the four intervals $t\le d-p_i-p_j$, $d-p_i-p_j < t\le d-p_i$, $d-p_i < t\le d-p_j$, and $d-p_j < t$.

Thus $i\to j$ unconditionally for every $i<j$ which means that $1,2,\ldots,n$ is the optimal

sequence. This confirms Rachamadugu's result about the optimality of the LPT sequence.

4. Computational Experience

We tested our decomposition method and Routines 1 and 2 on a number of randomly generated problems on IBM PS/2-70 using PASCAL. The integer processing times were drawn from a uniform distribution in a range [1, 100] while the due dates were generated from a uniform distribution of integers in a range [pa,pb] where $p = \sum_{k=1}^{n} p_k$. Thus the tardiness factor $T = 1 - \frac{a+b}{2}$ and the relative range R=b-a. For each of ten combinations of a,b and n, twenty test problems were generated. These data were used to solve ET problems for $\alpha=1$, $\beta=1$ and $\alpha=1$, $\beta=5$. The performance of our method that utilizes the optimal timing algorithm of [6] for 400 test problems where n=40 and 50 is summarized in Table 2.

a-b	n=40			n=50		
	$\alpha=1,\beta=1$	$\alpha=1,\beta=5$	# of cand seq	$\alpha=1,\beta=1$	$\alpha=1,\beta=5$	# of cand seq
0.1-0.5	42.18	32.96	20	51.41	53.17	20
0.1-0.9	31.97	35.04	20	46.46	44.38	21
0.1-1.3	32.90	33.50	21	42.57	43.34	31
0.1-1.7	53.61	50.91	177	52.57	50.37	75
0.1-2.1	50.87	62.21	111	122.32	115.34	261

Table 2. Performance of the Solution Procedure:
Cumulative CPU Time in Seconds for 20 Examples.

Notice that the set of candidate optimal sequences does not depend on α and β, while the actual optimal schedule does. The reduced sets turned out to be very small. In 79 out of 80 test problems for (a,b)=(0.1,0.5) and (0.1,0.9) there was a single sequence per example. We also solved all 100 examples of [7] using the optimal timing algorithms of [3] and [6]. The CPU times were 309.38 seconds and 251.04 respectively.

5. Final Remarks

This paper presents the first efficient method to solve an earliness-tardiness single machine scheduling model with idle time permitted without using any branch and bound method. In this model the earliness and tardiness penalties are proportional to the processing times of the jobs. The properties of adjacent job orderings provide tools to decompose the problem and reduce it to a small number of candidate sequences that are examined for optimality by an optimal timing algorithm. It would be interesting to explore whether some of those properties can be applied for the general earliness tardiness models.

Appendix

Proof of Lemma 1

Let $f_1(S) = \sum_{k=1}^{n} p_k E_k$ and $f_2(S) = \sum_{k=1}^{n} p_k T_k$.

According to Lemma 1 of [1], $f_1(S) = f_2(S)+K$, where K does not depend on S. Consequently, $f(S) = (\alpha+\beta)f_2(S) + \alpha K$. Let $T(S)=\beta f_2(S)$.

Then $\Delta_{ij}(t) = (\alpha+\beta)[f_2(S_2)-f_2(S_1)]$ while $\Delta'_{ij}(t) = \beta[f_2(S_2)-f_2(S_1)]$.

Hence $\dfrac{\Delta_{ij}(t)}{\alpha+\beta} = \dfrac{\Delta'_{ij}(t)}{\beta}$, QED.

References

[1] E.M. Arkin and R.O. Roundy, "Weighted Tardiness Scheduling on Parallel Machines with Proportional Weights," Operations Research, 39, 64-81 (1991).

[2] M.R. Garey, R.E. Tarjan, and G.T. Wilfong, "One Processor Scheduling with Symmetric Earliness and Tardiness Penalties," Mathematics of Operations Research 13, 330-348 (1988).

[3] J.S. Davis and J.J. Kanet, "Single-Machine Scheduling with Early and Tardy Completion Costs," Naval Research Logistics, 40 (1993), 85-101.

[4] R. Rachamadugu, "Scheduling Jobs Against a Common Due Date," Working Paper #631, School of Business Administration, The University of Michigan, Ann Arbor, Michigan 48109 (1990).

[5] W. Szwarc and J.J. Liu, "Weighted Tardiness Single Machine Scheduling with Proportional Weights," Management Science, 39, 626-632 (1993).

[6] W. Szwarc and S.K. Mukhopadhyay, "Optimal Timing Schedules in Earliness-Tardiness Single Machine Sequencing," to appear in Naval Research Logistics.

[7] C.A. Yano and Y-D Kim, "Algorithms for a Class of Single-Machine Weighted Tardiness and Earliness Problems," European Journal of Operational Research 52, 167-178 (1991).