Single-Machine Scheduling of Unit-Time Jobs with Earliness and Tardiness Penalties

Sushil Verma* Maged Dessouky†

January 7, 2000

Abstract

The problem of determining a schedule of jobs with unit-time lengths on a single machine that minimizes the total weighted earliness and tardiness penalties with respect to arbitrary rational due-dates is formulated as an integer programming problem. We show that if the penalties meet a certain criterion, called the *Dominance Condition,* then there exists an extremal optimal solution to the LP-relaxation that is integral, leading to a polynomial-time solution procedure. The general weighted symmetric penalty structure is one cost structure that satisfies the Dominance Condition; we point out other commonly found penalty structures that also fall in this category.

Keywords: Single-machine scheduling, earliness and tardiness penalties, unit-time jobs, linear programming.

1 Introduction

With the current emphasis on just-in-time (JIT) production, completing jobs earlier than their due dates should be discouraged as much as completing jobs later than their due dates. This dictates that the objective function of a scheduling problem should include both earliness and tardiness penalties, leading to an irregular measure of performance [Baker (1974)]. Baker and Scudder (1990) provide an excellent review of the single-machine scheduling problem with earliness and tardiness penalties. The single-machine weighted tardiness scheduling problem is known to be NP-complete [Lenstra et al. (1977)]. For this reason, many algorithms including dynamic programming, branch and bound, and heuristic rules, have been developed to solve the single-machine weighted tardiness scheduling problem (e.g., Fisher (1976), Lawler (1977), Baker and Schrage (1978), Schrage and Baker (1978), Rachamadugu and Morton (1983), Lenstra and Kan (1985), and Potts and Wassenhove (1985)). Recent research has focused on the weighted tardiness and earliness penalties (e.g., Ow and Morton (1989) and Yano and Kim (1991)). The main distinction between problems with tardiness penalties only and those with both earliness and tardiness penalties is that in the latter problem it might be necessary to insert idle time before processing a job to avoid earliness costs, while in the former problem there always exists an optimal solution in which there is no unforced idle time.

*This work was done while the author was at Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193

†Department of Industrial and Systems Engineering University of Southern California Los Angeles, CA 90089-0193

For single-machine problems with weighted tardiness and earliness penalties, there are polynomial-time algorithms known for only special cases of this problem. Examples are the algorithm by Kanet (1981) for the case with a common due date and one for both the tardiness and earliness weights and its extension by Kramer and Lee (1993) that considers a common time window instead of a single due date.

For identical processing times, the single-machine weighted tardiness scheduling problem can be formulated as an assignment problem [Lawler (1964)]. With identical processing times and only tardiness penalties, the set of busy periods consists of the first n time periods, where n is the number of jobs to schedule. Garey et al. (1988) develop an $O(n \log n)$ algorithm that determines the optimal schedule for identical processing times with the weights of tardiness and earliness of all jobs being the same. Hall and Posner (1991) develop another polynomial-time algorithm for the identical processing times case with the jobs having common due date but different weighted symmetric tardiness and earliness penalties. They also show that the same problem with non-identical processing times is NP-complete.

In this paper, we consider the single-machine weighted earliness and tardiness scheduling problem where all jobs have identical processing times and the due dates are not necessarily integer multiples of the processing time. Typically, the processing time is set to one and the due dates are represented as fractions of the processing times. If the due dates are integral, then the problem can be formulated as an assignment problem; the difficulty arises when the due dates are fractional. Our interest in fractional due dates arose from a study of a batch chemical manufacturing process. The bottleneck stage of the process was the mixing operation which produced non-identical batches (jobs) having the same processing time. The demand for batches was daily, and the mixing time of a batch (job) might be of the order of days so the dues dates in this case are not integer multiples of the processing time. The problem is described using the mathematical notation below.

Let $N = \{1, 2, ..., n\}$ be the set of jobs to schedule where n is the number of jobs. The due date of job i is $d_i \in \boldsymbol{Q}^+$. Let $\alpha_i \in \boldsymbol{Q}^+$ and $\beta_i \in \boldsymbol{Q}^+$ be the earliness and tardiness penalties of job i per unit time, respectively. We assume that the identical processing time of all jobs is 1. As is well known, identical processing time can be easily transformed into unit-time by redefining the time period to the common processing time.

Problem 1.1 Determine the completion times, C_i, for all jobs $i \in N$ that minimizes

$$\sum_{i \in N} (\alpha_i \max(0, d_i - C_i) + \beta_i \max(C_i - d_i, 0)) \tag{1.1}$$

The problem is first formulated as an integer linear programming (ILP) model. We show by example that the linear programming relaxation of the ILP need not have an integral optimal solution under general earliness and tardiness penalties. However, we prove that for a broad class of penalty structures there exists an extremal integral optimal solution to the LP-relaxation problem, thereby leading to a polynomial-time procedure. We define this class of problems by using the following condition.

Definition 1.1 Dominance Condition
Jobs can be indexed such that both $\alpha_1 \leq \alpha_2 \leq .... \leq \alpha_n$ and $\beta_1 \leq \beta_2 \leq .... \leq \beta_n$.

If we replace $\leq$ by $<$ in the above definition, we refer to it as the *strict* dominance condition.

Clearly, symmetric penalties, i.e., $\alpha_i = \beta_i$, $\forall i \in N$, satisfy the dominance condition. Garey et al. (1988) showed an $O(n \log n)$ algorithm for the special case of this problem where the earliness and tardiness penalties are identical across the jobs, i.e., $\alpha_i = \beta_i = 1, \forall i \in N$. Practically, the symmetric penalty structure represents just-in time scenarios where holding costs are as significant as lateness penalties. There are other asymmetric penalty structures which fall into this pattern. Some of these are as follows.

Identical α_i/β_i or $\alpha_i - \beta_i$: If all the jobs have different asymmetric penalties but either the ratio α_i/β_i or the difference $\alpha_i - \beta_i$ is the same for all $i \in N$, then the dominance condition is met. This case represents the situation where the holding cost is expressed as a fraction of the delay cost.

Identical Earliness Penalty: Often the holding costs for all the jobs are the same when the jobs are similar in nature, i.e., $\alpha_i = \alpha$ $\forall i \in N$. The tardiness costs might be different across the jobs if they meet demands of different customers. Such a penalty structure also meets the dominance condition.

Identical Tardiness Penalty: This case is just the reverse of the last example, i.e., $\beta_i = \beta$ $\forall i \in N$.

2 Problem Formulation

We first formulate Problem 1.1 as an integer linear programming model and then show that the optimal solution to the linear programming relaxation (LP-relaxation) of the model will be integer under certain conditions on the penalty weights.

Lemma 2.1 *There exists an optimal solution to Problem 1.1 in which job i is completed within the time interval* $[\max\{1, d_i - 2(n-1)\}, d_i + 2(n-1) + 1]$.

Proof : We will show that any schedule can be used to construct another schedule with no worse objective value and yet satisfying the above property. Consider a schedule in which job i is not completed within the claimed interval. Suppose that it completes prior to this interval. Let I be the subinterval $[\max\{1, d_i - 2(n-1)\}, d_i]$. Note that $d_i - 2(n-1) > 1$ otherwise job i cannot be completed before the interval I. Therefore the interval I is of length $2(n-1)$. Since there are no more than $n-1$ jobs of unit length processed in I, there must be an idle period of length at least one in this interval. If we reschedule job i to be processed within this idle period and therefore make it less early, it results in no degradation and possible improvement in the objective value. In this manner, all the jobs prior to this interval can be rescheduled in this interval with no worsening of the objective value. The case in which job i is processed after the claimed interval is completely analogous. The result follows. ■

Define K_i as the set of time points within the interval $[\max\{1, d_i - 2(n-1)\}, d_i + 2(n-1) + 1]$ with an integer distance from the due date d_i if $d_i \geq 1$. Let k in the following definition equal $\min\{2(n-1), \lfloor d_i - 1 \rfloor\}$.

$$K_i := \begin{cases} \{d_i - k, ..., d_i - 1, d_i, d_i + 1, ..., d_i + 2(n-1) + 1\}, & \text{if } d_i \geq 1 \\ \{\}, & \text{if } d_i < 1 \end{cases} \tag{2.1}$$

Also define,

$$K_0 := \{1, 2, 3, 4, .., n\}. \tag{2.2}$$

Define

$$J := \bigcup_{i=0}^{n} K_i. \tag{2.3}$$

The members of set J will be referred to as *locations* . Finally, define subsets of set J for $i \in N$.

$$S_i := J \cap [\max\{1, d_i - 2(n-1)\}, d_i + 2(n-1) + 1] \tag{2.4}$$

The significance of sets S_i is seen through the following proposition.

Proposition 2.1 *There exists an optimal solution to Problem 1.1 such that the optimal completion time for job i belongs to set S_i.*

Proof : Proof is by construction of an optimal solution with the desired property by suitably modifying any optimal solution to Problem 1.1. We start with an optimal solution in which the completion time of job i is inside the interval $[\max\{1, d_i - 2(n-1)\}, d_i + 2(n-1) + 1]$. Such an optimal solution is guaranteed to exist in light of Lemma 2.1. Suppose that this optimal solution has some completion times not belonging to set J. This implies that there exists a set of one or more contiguously scheduled jobs whose completion times are not a integer distance away from either time 1 or one of the due dates. Consider the largest such set, T. Note that none of the jobs in this set are completed on-time; they are either strictly early or strictly tardy. Refer to the jobs in T which are early and tardy as T_e and T_t respectively. Since T is a maximal contiguous set of jobs, there must be some idle time before the start time of the first job and the completion time of the last job in this set T.

Since each job in this set is either strictly tardy or strictly early, the objective function is locally linear with respect to the completion times of each of these jobs. Therefore, the local derivative of the objective function in the direction of equal changes in the completion times of the jobs in set T is $\sum_{i \in T_t} \beta_i - \sum_{i \in T_e} \alpha_i$. Since the current solution is optimal, this derivative must be zero. It implies that we can reduce the completion times of all the jobs in T by the same amount without changing the objective value, provided the early jobs do not become tardy and the tardy jobs do not become early. The amount of reduction in the completion times is further limited by the idle time present before the starting time of the first job in set T. The goal is to reduce the completion times of the jobs in set T by an amount which is just enough to make them members of set J, if such a reduction is possible. Otherwise, we reduce the completion times just enough to fill up the idle time lying just left of the first job in T which increases the size of set T by at least one. It is clear that in either case, the completion time of job i stays in the interval $[\max\{1, d_i - 2(n-1)\}, d_i + 2(n-1) + 1]$. We formally describe this process next.

Let the idle time before the start time of the first job in T be equal to ϵ. Let δ be the minimum decrease of completion time of all the jobs in this set such that resulting completion times are an integer distance away from either time 1 or one of the due dates. In mathematical notation, $\delta = \min\{C_i - j | i \in T,\ j \in J \text{ such that } C_i - j \geq 0\}$. Define $\Delta = \min(\delta, \epsilon)$. We reduce the completion time of all the jobs in set T by Δ. Since we are only reducing the completion times, the early jobs clearly do not become tardy. Moreover, the reduction is such that the tardy jobs do not become early. Since $\Delta \leq C_i - d_i$ for any tardy job $i \in T$, the new completion times ($C_i - \Delta$ for tardy job $i \in T$) would not be less than their due dates. We can therefore conclude that such a reduction will not have any effect on the objective function value. If $\Delta = \delta$, then it implies that the completion times of the jobs in set T now belong to set J. If $\Delta = \epsilon \neq \delta$, the start time of the first job in set T becomes equal to the completion time of some other job also not in set J.

We repeat the above process until all job completion times belong to set J. At each step of the above process, we either add some more job completion times to set J (in case $\Delta = \delta$) or the size of set T increases by at least one (in case $\Delta = \epsilon \neq \delta$). Moreover, at no step does the completion time of a job leave set J. Therefore, there are no more than n iterations. ■

We are now ready to present the integer linear programming formulation of Problem 1.1. The variables of the formulation are $x_{i,j}$ where $i \in N$ is the job number and $j \in S_i$ is the location index. $x_{i,j} = 1$ if $C_i = j$ and zero otherwise. Note that j is not necessarily an integer since the due dates, d_i, $i \in N$, can be non-integer. The coefficient of $x_{i,j}$ in the objective function is $c_{i,j}$ defined below.

$$c_{i,j} := \alpha_i \max(0, d_i - j) + \beta_i \max(j - d_i, 0) \tag{2.5}$$

To avoid job overlaps, we define the following set.

$$H_j := \{j' \in J |\ 0 \leq j - j' < 1\} \tag{2.6}$$

The set H_j represents the time points in J before j in which no other jobs may be scheduled to complete if a job is scheduled to be completed at time j. The complete ILP formulation of Problem 1.1 is

Problem 2.1

$$\min \quad z = \sum_{i \in N} \sum_{j \in S_i} c_{i,j} x_{i,j}$$

subject to

$$\sum_{j \in S_i} x_{i,j} = 1 \quad \text{for all } i \in N \tag{2.7}$$

$$\sum_{i \in N} \sum_{j' \in H_j \cap S_i} x_{i,j'} \leq 1 \text{ for all } j \in J \tag{2.8}$$

$$x_{i,j} \in \{0, 1\} \quad \text{for all } i \in N \text{ and for all } j \in S_i \tag{2.9}$$

Constraint 2.7 ensures that each job is assigned a completion time and Constraint 2.8 ensures that not more than one job is scheduled in conflicting production intervals.

We provide an example for the above constraints.

Example 2.1 $n = 2$, $d_1 = 1.5$, and $d_2 = 2$.

$$
\begin{aligned}
K_0 &= \{1,2\} \\
K_1 &= \{1.5, 2.5, 3.5, 4.5\} \\
K_2 &= \{1,2,3,4,5\} \\
J &= \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\} \\
S_1 &= \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5\} \\
S_2 &= \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\} \\
H_1 &= \{1\} \\
H_{1.5} &= \{1, 1.5\} \\
H_2 &= \{1.5, 2\} \\
H_{2.5} &= \{2, 2.5\} \\
H_3 &= \{2.5, 3\} \\
H_{3.5} &= \{3, 3.5\} \\
H_4 &= \{3.5, 4\} \\
H_{4.5} &= \{4, 4.5\} \\
H_5 &= \{4.5, 5\}
\end{aligned}
$$

The constraint matrix for Example 2.1 is as follows. The variables (columns) are indexed by a pair (i, j). The first two rows correspond to constraint set 2.7 and the remaining rows correspond to constraint set 2.8.

i	(1,1.0)	(1,1.5)	(1,2.0)	(1,2.5)	(1,3.0)	(1,3.5)	(1,4.0)	(1,4.5)	(2,1.0)	(2,1.5)	(2,2.0)	(2,2.5)	(2,3.0)	(2,3.5)	(2,4.0)	(2,4.5)	(2,5.0)
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
j																	
1.0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1.5	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
2.0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
2.5	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0
3.0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
3.5	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0
4.0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
4.5	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0
5.0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

It should be noted that if the due dates are integers, then $H_j = \{j\}$ for all j. In this case, the ILP formulation is identical to an assignment problem formulation and Problem 1.1 can be solved in time complexity $O(n^6)$ since there are $O(n^2)$ variables. However, this complexity can be reduced by recognizing that the set of candidate busy periods is of cardinality at most $2n$ and the problem can be formulated as an assignment problem of time complexity $O(n^3)$ [Dessouky et al. (1998)]. Note that with only tardiness penalties, the problem can be formulated as an assignment problem even when the due dates are non-integer [Lawler (1964)].

In case the penalty structure is strongly asymmetric, in the sense that it does not meet the dominance condition, then the LP-relaxation problem need not have an integral optimal solution. We show this result by example. Suppose there are two jobs to be scheduled with $d_1 = 2-\delta$ and $d_2 = 2$. The penalties are $\alpha_1 = 1+\epsilon$, $\alpha_2 = 1$, $\beta_1 = 1$, and $\beta_2 = 1+\epsilon$. If $\epsilon > \frac{\delta}{1-\delta}$ and $1 > \delta > 0$, then there exists a unique non-integral optimal solution. To illustrate, consider Example 2.1, which corresponds to $\delta = 1/2$. With the above penalty data, its (unique) optimal solution for any $\epsilon > 1$ is as follows: $x_{2,1.0} = x_{1,1.5} = x_{2,2.0} = x_{1,2.5} = 1/2$

and the remaining variables are zero. We believe that although in this example all the variables assume half-integral values, it is not always the case.

In the next section, we show that if the dominance condition is satisfied then there always exist an extremal optimal solution of the LP-relaxation of the above ILP that is integral. Moreover, if the strict dominance condition is satisfied then any extremal optimal solution of the LP-relaxation of the above ILP is integral.

3 Integrality Under Dominance

In this section we show that the concept of dominance plays a key role in the existence of integral optimal solutions of Problem 2.1. The main idea behind the proof of integrality is to divide the set of feasible solutions into two categories. One category is shown to be always sub-optimal under the strict dominance condition. The other category is shown to possess integral extremal points. We first define the two kinds of feasible solution.

Definition 3.1 Let $\boldsymbol{x}$ be a feasible solution to the LP-relaxation of Problem 2.1. We say that job i_2 is *nested* in job i_1 ($i_1 \neq i_2$), if there exist $j_k \in J$, $k \in \{1,2,3\}$ such that $j_1 < j_2 < j_3$ and x_{i_1,j_1}, x_{i_2,j_2} and x_{i_1,j_3} are all positive.

We define a relation $\prec_{\boldsymbol{x}}$ naturally based upon the above definition. We say that $i_2 \prec_{\boldsymbol{x}} i_1$, if job i_2 is nested in job i_1 in a feasible solution $\boldsymbol{x}$.

Recall the following from the theory of sets [Royden (1968), pp22-23]. A relation $\boldsymbol{R}$ is said to be *antisymmetric* on a set $\boldsymbol{X}$ if $x \ \boldsymbol{R}\ y$ and $y \ \boldsymbol{R}\ x$ imply $x = y$ for all x and y in $\boldsymbol{X}$. A relation $\boldsymbol{R}$ is said to be *transitive* on a set $\boldsymbol{X}$ if $x \ \boldsymbol{R}\ y$ and $y \ \boldsymbol{R}\ z$ imply $x \ \boldsymbol{R}\ z$ for all x, y and z in $\boldsymbol{X}$. A relation $\boldsymbol{R}$ is called a *partial ordering* of a set $\boldsymbol{X}$ if it is transitive and antisymmetric on $\boldsymbol{X}$.

Definition 3.2 Let $\boldsymbol{x}$ be a feasible solution to the LP-relaxation of Problem 2.1. We say that $\boldsymbol{x}$ is *nested* if $\prec_{\boldsymbol{x}}$ is a *partial ordering* of the set of jobs, N. Otherwise, $\boldsymbol{x}$ is called *non-nested* .

We will drop the subscript $\boldsymbol{x}$ in $\prec_{\boldsymbol{x}}$ when there is no danger of confusion. Also, in the rest of this section, we refer to a feasible solution $\boldsymbol{x}$ of the LP-relaxation of Problem 2.1 just as "a feasible solution $\boldsymbol{x}$".

Lemma 3.1 *If a feasible solution $\boldsymbol{x}$ is non-nested then there exist jobs i_1 and i_2, $i_1 \neq i_2$, which are nested in each other, i.e., $i_1 \prec i_2$ and $i_2 \prec i_1$.*

Proof : Suppose the feasible solution $\boldsymbol{x}$ is non-nested. Definition 3.2 dictates that $\prec$ does not define a partial ordering on set N. Therefore, $\prec$ is either not antisymmetric or not transitive. We show that in either case there exist two distinct jobs which are nested in each other.

If $\prec$ is not antisymmetric, then by definition there must exist some distinct jobs i_1 and i_2 such that $i_1 \prec i_2$ and $i_2 \prec i_1$. If $\prec$ is not transitive, then there must exist some distinct jobs i_1, i_2 and i_3 such that $i_1 \prec i_2$, $i_2 \prec i_3$ but $i_1 \not\prec i_3$. We prove that $i_3 \prec i_2$, implying that i_2 and i_3 are nested in each other. Since $i_i \prec i_2$, job i_1 has a positive allocation at a location (call it j) which lies strictly between the first and the last location with positive allocation of job i_2 (call them f_2 and l_2 respectively). Since $i_1 \not\prec i_3$, all the allocation of

job i_3 is either no later or no earlier than location j. We prove it for the case when all the allocation is no earlier than location j. The proof for the other case is along similar lines. Call the first location with a positive allocation of job i_3 as f_3. If $f_3 \geq l_2$, then i_2 cannot be nested in i_3. Therefore, $j \leq f_3 < l_2$, implying that job i_3 is nested in job i_2 and the result follows. ■

We exclude non-nested optimal solutions in the following theorem.

Theorem 3.1 *If the job penalties meet the strict dominance condition, then any optimal solution to the LP-relaxation of Problem 2.1 is nested.*

Proof : We prove by contradiction. Assume that we have an optimal solution which is non-nested. Therefore, according to Lemma 3.1 there exist distinct jobs i_1 and i_2 which are nested in each other. Let's assume without loss of generality that $i_2 < i_1$. Therefore, both the earliness and the tardiness penalties for job i_1 are strictly greater than that of job i_2. Moreover, since i_2 is nested in i_1, there must exist locations j_1, j_2 and j_3 all in set J such that $j_1 < j_2 < j_3$ and $x_{i_1,j_1}, x_{i_2,j_2}, x_{i_1,j_3}$ are positive. Let $\epsilon = \min(x_{i_1,j_1}, x_{i_2,j_2}, x_{i_1,j_3})$. We show that the current solution can be strictly improved in at least one of the following two ways.

Modification (a) Increase x_{i_2,j_1} and x_{i_1,j_2} by ϵ. Decrease x_{i_1,j_1} and x_{i_2,j_2} by ϵ.

Modification (b) Increase x_{i_2,j_3} and x_{i_1,j_2} by ϵ. Decrease x_{i_1,j_3} and x_{i_2,j_2} by ϵ.

Before we measure the effect of the above modifications on the objective function of the LP-relaxation problem, we divide the different due date structures into four cases.

Case (1) $d_{i_1}, d_{i_2} \leq j_2$

Case (2) $d_{i_1}, d_{i_2} \geq j_2$

Case (3) $d_{i_1} \leq j_2$, $d_{i_2} \geq j_2$

Case (4) $d_{i_2} \leq j_2$, $d_{i_1} \geq j_2$

Note that for any d_{i_1} and d_{i_2}, at least one of the above four conditions is satisfied. We consider only Case (1) and Case (3). We show that in both these cases modification (b) strictly improves the value of the objective function. It can be shown along similar lines that for Case (2) and Case (4) modification (a) strictly improves the value of the objective function. Note that $\epsilon > 0$. We will also frequently use the fact that $\beta_{i_1} > \beta_{i_2}$ under the strict dominance condition. We will explicitly compute the effect of modifications on the value of the objective function under the two cases.

Case(1): The change in the objective function is equal to $\epsilon(-\beta_{i_1}(j_3 - j_2) + \beta_{i_2}(j_3 - j_2)) = \epsilon(j_3 - j_2)(\beta_{i_2} - \beta_{i_1})$. Since $j_3 > j_2$ and $\beta_{i_2} < \beta_{i_1}$, the change is strictly negative.

Case(3): The change in the objective function is equal to $\epsilon(-\beta_{i_1}(j_3 - j_2) - \alpha_{i_2}(d_{i_2} - j_2) + \max(\beta_{i_2}(j_3 - d_{i_2}), \alpha_{i_2}(d_{i_2} - j_3)))$. If $j_3 \geq d_{i_2}$, the change equals $\epsilon(-\beta_{i_1}(j_3 - j_2) - \alpha_{i_2}(d_{i_2} - j_2) + \beta_{i_2}(j_3 - d_{i_2}))$. Since $j_3 - j_2 \geq j_3 - d_{i_2} \geq 0$, $j_3 > j_2$ and $\beta_{i_1} > \beta_{i_2} \geq 0$, the change is strictly negative. If $j_3 \leq d_{i_2}$, the change equals $\epsilon(-\beta_{i_1}(j_3 - j_2) - \alpha_{i_2}(j_3 - j_2))$. Since $j_3 > j_2$ and $\beta_{i_1} > \beta_{i_2} \geq 0$, the change is again strictly negative.

We have shown that $\boldsymbol{x}$ is not an optimal solution, which is a contradiction to our assumption. Therefore, any optimal solution to the LP-relaxation must be nested. ■

In case the job penalties meet the dominance condition, but not strictly, then the following corollary guarantees existence of a nested optimal solution. It uses the next lemma which quotes a well-known result from the theory of linear programming [Murthy (1983)].

Lemma 3.2 *For any real $n \times 1$ vectors $\boldsymbol{c}$ and $\bar{\boldsymbol{c}}$, $m \times 1$ vector $\boldsymbol{b}$ and $m \times n$ matrix $\boldsymbol{A}$, there exists $\bar{\epsilon} > 0$, such that for $0 \leq \epsilon \leq \bar{\epsilon}$ any optimal solution to the linear program $\min(\boldsymbol{c} + \epsilon\bar{\boldsymbol{c}})^T\boldsymbol{v}$ subject to $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{b},\ \boldsymbol{v} \geq 0$ is also optimal to the linear program $\min \boldsymbol{c}^T\boldsymbol{v}$ subject to $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{b},\ \boldsymbol{v} \geq 0$.*

Corollary 3.2 *If the job penalties meet the dominance condition, then there exists a nested extremal optimal solution to the LP-relaxation of Problem 2.1.*

Proof : Consider the perturbed job penalties $\bar{\alpha}_i = \alpha_i + i\epsilon$ and $\bar{\beta}_i = \beta_i + i\epsilon$. According to Lemma 3.2, any optimal solution to the LP-relaxation of Problem 2.1 with job penalties $(\bar{\alpha}_i, \bar{\beta}_i)$, is also optimal to the LP-relaxation with job penalties (α_i, β_i), provided ϵ is small enough. Moreover, since for any $\epsilon > 0$ the perturbed job penalties satisfy the strict dominance condition, all optimal solutions to the LP-relaxation with perturbed penalties are nested, as we proved in Theorem 3.1. Therefore, we have the result. ■

We are ready to present the main result of this paper which states that an extremal nested feasible solution is integral. It uses certain key properties of interval matrices. We first define an interval matrix and then present two lemmas about its properties. The proofs are either cited or are obvious.

Definition 3.3 An ***interval matrix*** is a 0-1 matrix with the property that under some linear ordering of its columns, all the 1's in each row are consecutive.

Lemma 3.3 (Nemhauser and Wolsey (1988), pp 544) *Interval matrices are totally unimodular.*

Lemma 3.4 *Suppose $\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix}$ is an interval matrix and vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are integral. Then, all the vertices of the polyhedron*

$$\begin{aligned} \boldsymbol{A}_1\boldsymbol{v} &= \boldsymbol{b}_1 \\ \boldsymbol{A}_2\boldsymbol{v} &\leq \boldsymbol{b}_2 \\ \boldsymbol{v} &\geq 0 \end{aligned}$$

are integral.

Proof : Since interval matrices are totally unimodular, the result follows from the property of totally unimodular matrices. ■

Lemma 3.5 *If $\boldsymbol{A}$ is an interval matrix and $\boldsymbol{a}_k$ is its kth column, then the matrices*

(i) $[\boldsymbol{a}_1, \boldsymbol{a}_2, .., \boldsymbol{a}_{k-1}, \boldsymbol{a}_k, \boldsymbol{a}_k, \boldsymbol{a}_{k+1}, ..\boldsymbol{a}_n]$, *and*

(ii) $[\boldsymbol{a}_1, \boldsymbol{a}_2, .., \boldsymbol{a}_{k-1}, \boldsymbol{a}_{k+1}, ..\boldsymbol{a}_n]$

are interval matrices.

Let $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ be the part of the constraint matrix corresponding to constraints (2.7) and (2.8) respectively. The system of constraints in terms of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ is the following. The vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ denote vectors of all 1's.

$$\begin{aligned} \boldsymbol{A}_1\boldsymbol{x} &= \boldsymbol{e}_1 \\ \boldsymbol{A}_2\boldsymbol{x} &\leq \boldsymbol{e}_2 \\ \boldsymbol{x} &\geq 0 \end{aligned}$$

The columns of these matrices can be indexed by the pair (i, j), where $i \in N$ denotes the job number and $j \in S_i$ denotes a location to which job i may be possibly allocated. The rows in the matrix $\boldsymbol{A}_1$ enforce the constraint that the total allocation of each job is a unit. Therefore, there are n rows in matrix $\boldsymbol{A}_1$ while matrix $\boldsymbol{A}_2$ contains $|J|$ rows. $\boldsymbol{A}_1$ has the following structure. Suppose its rows are indexed by r and columns are indexed, as per above, by (i, j). Let $a_{r,(i,j)}$ be the coefficient in the rth row and (i, j)th column in matrix $\boldsymbol{A}_1$. Then, $a_{r,(i,j)} = 1$ if and only if $r = i$, otherwise it is zero.

Suppose that columns are ordered such that for all i, the column (i, j_1) precedes column (i, j_2) if and only if $j_1 < j_2$. The columns $(i_1, j), (i_2, j), ..., \text{and } (i_n, j)$ are ordered arbitrarily for the moment for all j. In the rest of this section, we will denote the columns $(i_1, j), (i_2, j), ..., \text{and } (i_n, j)$ collectively as $(., j)$ for ease in presentation. We illustrate this new ordering on the data of Example 2.1.

$$
\begin{array}{cc}
 & \begin{array}{ccccccccccccccccc}
(1,1.0) & (2,1.0) & (1,1.5) & (2,1.5) & (1,2.0) & (2,2.0) & (1,2.5) & (2,2.5) & (1,3.0) & (2,3.0) & (1,3.5) & (2,3.5) & (1,4.0) & (2,4.0) & (1,3.5) & (2,4.5) & (2,5.0)
\end{array} \\
\boldsymbol{A}_1 = & \left[\begin{array}{ccccccccccccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1
\end{array}\right] \\
\boldsymbol{A}_2 = & \left[\begin{array}{ccccccccccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}\right]
\end{array}
$$

Notice in the above example that A_2 is an interval matrix. We next prove that this will always be the case under the above ordering.

Lemma 3.6 *The matrix $\boldsymbol{A}_2$ is an interval matrix. The 1's in each row of the matrix $\boldsymbol{A}_2$ are consecutive if the columns are ordered such that for all $i \in N$ and $j_1, j_2 \in S_i$, the column (i, j_1) precedes column (i, j_2) if and only if $j_1 < j_2$. Moreover, the 1's in each row of the matrix $\boldsymbol{A}_2$ are consecutive under all orderings of columns $(., j)$, for each j.*

Proof : First note that for any location j and jobs i_1 and i_2, columns (i_1, j) and (i_2, j) in the matrix $\boldsymbol{A}_2$ are identical if they both exist. Therefore, under light of Lemma 3.5 we need only prove that the column submatrix formed of the columns $\{(i, j') | j' \in S_i\}$ is an interval matrix for any given i. Refer to this submatrix as $\boldsymbol{A}_2^i$. It can be interpreted as the constraint matrix of the following set of imaginary constraints.

$$\sum_{j' \in H_j \cap S_i} x_{i,j'} \leq 1 \text{ for all } j \in J$$

Both the sets H_j and S_i consist of contiguous locations and hence the set $H_j \cap S_i$ also consists of only contiguous locations. Since the columns are ordered by their location index, the 1's in each row of this matrix must be consecutive. Therefore, $\boldsymbol{A}_2^i$ is an interval matrix. As we argued earlier, columns $(., j)$ are identical in the matrix $\boldsymbol{A}_2$. Therefore, their ordering leaves the matrix $\boldsymbol{A}_2$ unchanged. ■

We have shown that $\boldsymbol{A}_2$ is an interval matrix. However, $\boldsymbol{A}_1$ in its present form is not an interval matrix. We prove that after a suitable transformation and a certain ordering of columns $(., j)$, we can turn $\boldsymbol{A}_1$ into an interval matrix under certain special circumstances. The next two lemmas build the basis for the proposition.

Lemma 3.7 *Two jobs cannot have positive allocation at the same two locations in an extremal feasible solution, i.e., there do not exist locations j_1 and j_2 and jobs i_1 and i_2 such that x_{i_1,j_1}, x_{i_1,j_2}, x_{i_2,j_1}, x_{i_2,j_2} are all positive.*

Proof : The elements in column (i,j) in matrix $\boldsymbol{A}_1$ are independent of j since the column (i,j) contains one 1 in the ith row independent of j. The elements in column (i,j) in matrix $\boldsymbol{A}_2$ are independent of i. Therefore, the difference of columns (i_1,j_k) and (i_2,j_k) is independent of k in the complete matrix $\begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{bmatrix}$. The four columns (i_1,j_1), (i_1,j_2), (i_2,j_1), and (i_2,j_2) are therefore not independent and cannot be part of a basis. ■

Lemma 3.8 *Suppose $\boldsymbol{x}$ is a nested feasible solution and no two jobs have positive allocation at the same two locations. Then two jobs are incomparable under the partial ordering $\prec_{\boldsymbol{x}}$ if and only if they can be indexed i_1 and i_2 and there exists a location j such that $x_{i_1,j'} = 0$ $\forall j' > j$ and $x_{i_2,j'} = 0$ $\forall j' < j$.*

Proof : We first prove the "only if" part. If two jobs are incomparable under $\prec$, then neither is nested in the other. This relation implies that neither job has a positive allocation at a location which is strictly between the first and last location with a positive allocation of the other job. We are left with two possibilities. *Either* one of the jobs has its last location with positive allocation no later than the first location with positive allocation of the other job, *or* both the jobs have positive allocation at the same two locations. The latter is precluded by assumption. Therefore, we have the result.

To prove the "if" part, suppose that the two jobs are comparable under $\prec$. We can index them such that $i_1 \prec i_2$ implying that job i_1 has a positive allocation at a location (call it $\bar{j}$) which lies strictly between the first and the last location with positive allocation of job i_2 (call them j_1 and j_2 respectively). Note that $j_1 < \bar{j} < j_2$. Any location j for which $x_{i_1,j'} = 0$ $\forall j' > j$ and $x_{i_2,j'} = 0$ $\forall j' < j$ has to be such that $j \geq \bar{j}$ (implied by the first equation) and $j \leq j_1$ (implied by the second equation). The two inequalities cannot however be met at the same time since $\bar{j} > j_1$. Therefore, location j of the desired type cannot exist. ■

Given a nested feasible solution $\boldsymbol{x}$, we transform $\boldsymbol{A}_1$ and the right hand side $\boldsymbol{e}_1$ in the following fashion. Refer to the transformed matrix as $\bar{\boldsymbol{A}}_1$ and the transformed right hand side as $\bar{\boldsymbol{b}}$. Let $\boldsymbol{a}_i$ and $\bar{\boldsymbol{a}}_i$ be the ith row of $\boldsymbol{A}_1$ and $\bar{\boldsymbol{A}}_1$ respectively. Then,

$$\bar{\boldsymbol{a}}_i = \boldsymbol{a}_i + \sum_{i' \prec i} \boldsymbol{a}_{i'}, \tag{3.1}$$

$$\bar{b}_i = 1 + \sum_{i' \prec i} 1. \tag{3.2}$$

Lemma 3.9 *Suppose $\boldsymbol{x}$ is a nested feasible solution and $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{b}}$ are given by equations (3.1) and (3.2) respectively. Then, the system of equations $\bar{\boldsymbol{A}}\boldsymbol{v} = \bar{\boldsymbol{b}}$ is equivalent to the system of equations $\boldsymbol{A}_1\boldsymbol{v} = \boldsymbol{e}_1$.*

Proof : Equations (3.1) and (3.2) are equivalent to a series of row-operations on the system $\boldsymbol{A}_1\boldsymbol{v} = \boldsymbol{e}_1$ performed in a particular sequence. If $i_1 \prec i_2$, then the row-operation corresponding to the i_1th equations in (3.1) and (3.2) is performed only after the row-operation corresponding to the i_2th equations is performed. This is feasible for any i_1 and i_2 because $\prec$ defines a partial ordering on N. Since the row-operations leave the solution space of a system of equations unchanged, we have the result. ■

Proposition 3.1 *If $\boldsymbol{x}$ is a nested feasible solution such that no two jobs have positive allocations at the same two locations then the submatrix of $\bar{\boldsymbol{A}}_1$ achieved by dropping the column (i,j) if $x_{i,j}=0$, is an interval matrix. Suppose that the columns are ordered such that for all $i \in N$ and $j_1, j_2 \in S_i$, the column (i,j_1) precedes column (i,j_2) if and only if $j_1 < j_2$. Then, there exists an ordering of columns $(.,j)$ for each j, under which the 1's in each row of the above mentioned submatrix are consecutive.*

Proof : Since each column of $\boldsymbol{A}_1$ contains only one 1, the sum of any number of its rows is also a 0-1 vector. Therefore, $\bar{\boldsymbol{A}}_1$ is a 0-1 matrix.

With an arbitrary ordering of columns $(.,j)$, the 1's in each row of $\bar{\boldsymbol{A}}_1$ might not be consecutive. We order columns $(.,j)$ in the following fashion. Location j can either be the first, the last, or an intermediate location with positive allocation for some job i with $x_{i,j} > 0$. Refer to these jobs as of Type F, L and M respectively with respect to location j. For location j, let columns corresponding to jobs of Type L precede columns of Type M which precede columns of Type F. There can be at most one job of Type M because if there were two, they would clearly be nested in each other, a contradiction to our assumption. Within Types F and L all the jobs are comparable to each other under $\prec$, since according to Lemma 3.8 there cannot exist any location that separates the locations with positive allocation of any two jobs. Within Type F, let the column (i_1,j) precede (i_2,j) if $i_2 \prec i_1$. Within Type L, let the column (i_1,j) precede (i_2,j) if $i_1 \prec i_2$. Note that the column ordering is well defined because $\prec$ is a partial ordering on N.

Consider the submatrix of the matrix $\bar{\boldsymbol{A}}_1$ achieved by dropping each column (i,j) for which $x_{i,j}=0$. Consider the row i of this submatrix under the above ordering. There is a 1 in column (i',j) of this row if and only if $x_{i',j} > 0$ and either job $i' \prec i$ or $i'=i$. We show that these 1's are consecutive by proving that all the columns between the first and the last column with a 1, are of type (i',j) where $i' \prec i$ or $i'=i$.

First we show that if f and l are the first and the last locations with positive allocation for job i, then the first 1 in this row is at column (i,f) and the last 1 in this row is at column (i,l). It is clear that the first 1 in this row would be in column (i',f') where either $i' \prec i$ or $i'=i$, and $f' \leq f$ is the first location with positive allocation of any such job i'. Consider a job i' such that $i' \prec i$ implying that there exists a location j, $f < j < l$ and $x_{i',j} > 0$. We show that $f'=f$. Assume to the contrary that $f' < f$. Therefore, $f' < f < j$ and $x_{i',f'}, x_{i',j}, x_{i,f} > 0$, implying that $i \prec i'$. It thus contradicts the given assumption that $\boldsymbol{x}$ is a nested feasible solution. Therefore, f' must be equal to f. But this implies that job i' and job i both are of Type F with respect to location f, and therefore column (i,f) would precede (i',f) under the column ordering prescribed previously. The proof that the last 1 in this row is at column (i,l) is along analogous lines.

We now prove that all the columns between (i,f) and (i,l) are of type (i',j) where $i' \prec i$ or $i'=i$. Note that any column which could possibly be between (i,f) and (i,l) must be of the type (i',j), where $f \leq j \leq l$. Suppose $i' \not\prec i$, implying that j cannot be strictly between f and l. Then, without loss of generality, assume that $j=f$. If i' is of Type L or Type M with respect to location f, then the column (i',f) would precede column (i,f). If i' is of Type F with respect to location f, then job i must be nested in job i', because they cannot be incomparable in light of Lemma 3.8, and $i' \not\prec i$ by assumption. Therefore, under the above ordering, column (i',f) must precede column (i,f). The result follows. ■

The main theorem follows.

Theorem 3.3 *Any extremal nested feasible solution to the LP-relaxation of Problem 2.1 is integral.*

Proof : Suppose that $\boldsymbol{x}$ is a nested feasible solution to the LP-relaxation of Problem 2.1 that is non-integral. We show that it must be non-extremal.

In this solution there might be two jobs assigned at the same two locations. According to Lemma 3.7 such a $\boldsymbol{x}$ could not be extremal. Therefore, we need only to consider the case where two jobs share at most one location.

We show that $\boldsymbol{x}$ is a feasible solution to a set of constraints which suitably transformed satisfy the conditions of Lemma 3.4. This set of constraints is just the set of constraints of Problem 2.1 without the columns (i, j) such that $x_{i,j} = 0$. Since the system $\boldsymbol{A}_1\boldsymbol{v} = \boldsymbol{e}_1$ is equivalent to the system $\bar{\boldsymbol{A}}_1\boldsymbol{v} = \bar{\boldsymbol{b}}$ (Lemma 3.9), we can rewrite the constraints of Problem 2.1 as the following.

$$\begin{aligned} \bar{\boldsymbol{A}}_1\boldsymbol{v} &= \bar{\boldsymbol{b}} \\ \boldsymbol{A}_2\boldsymbol{v} &\leq \boldsymbol{e}_2 \\ \boldsymbol{v} &\geq 0 \end{aligned} \tag{3.3}$$

Suppose the columns of the above system are ordered by location, i.e., for all $i \in N$ and $j_1, j_2 \in S_i$, the column (i, j_1) precedes column (i, j_2) if and only if $j_1 < j_2$. Moreover, for all j, the columns $(., j)$ are ordered such that the matrix $\bar{\boldsymbol{A}}_1$ has the consecutive 1's property. We proved in Proposition 3.1 that such an ordering exists as long as we drop the columns with no allocation, and no two jobs have a positive allocation at the same two locations. Both the conditions are met here. Also, under the above ordering, the matrix $\boldsymbol{A}_2$ also has the consecutive 1's property (Lemma 3.6). Therefore $\boldsymbol{A}_2$, with some columns dropped, is still an interval matrix (Lemma 3.5). Lastly, $\bar{\boldsymbol{b}}$ is an integral vector and $\boldsymbol{e}_2$ is a vector of all ones. Therefore, according to Lemma 3.4, the feasible solution $\boldsymbol{x}$ must be a non-extremal feasible solution to the transformed set of constraints (3.3). Since the solution set of the transformed constraints is just a lower dimension face of the solution set of the original constraints, $\boldsymbol{x}$ is also an extremal feasible solution to the LP-relaxation. ■

Corollary 3.4 *If the job penalties meet the strict dominance condition, then all extremal optimal solutions to the LP-relaxation of Problem 2.1 are integral. Furthermore, if the job penalties meet the dominance condition, then there exists an extremal integral optimal solution to the LP-relaxation of Problem 2.1.*

Proof : The first statement follows immediately from Theorem 3.1, which prohibits non-nested solutions under the strict dominance condition, and Theorem 3.3, which assures that nested solutions are either non-extremal or integral. The second statement follows from Corollary 3.2, which proves existence of extremal nested solutions under the dominance condition, and Theorem 3.3. ■

Given that the optimal face contains an extremal integral solution, there are a variety of techniques that can be used to identify it. One technique is based upon perturbation of the job penalties of the type described in the proof of Theorem 3.1.

$$\bar{\alpha}_i = \alpha_i + i\epsilon \tag{3.4}$$

$$\bar{\beta}_i = \beta_i + i\epsilon \tag{3.5}$$

For any $\epsilon > 0$, all the optimal solutions of the LP-relaxation of Problem 2.1 will be integral, because the perturbed penalties satisfy the strict dominance condition. For small enough ϵ, these optimal solutions will be also optimal to Problem 2.1 with unperturbed penalties.

The following proposition prescribes an upper bound on the parameter ϵ. Define L to be the length of the binary encoding of the unperturbed rational penalties and the rational due dates. Recall that the length of binary encoding of an integer p is $1 + \lceil \log_2(|p| + 1) \rceil$ [Grötschel et al. (1980)].

Proposition 3.2 *If $\epsilon < 2^{-L}/2n^3$, then an optimal solution to Problem 2.1 with job penalties $(\bar{\alpha}_i, \bar{\beta}_i)$ is also optimal to Problem 2.1 with job penalties (α_i, β_i).*

Proof : Because of the perturbation specified in equations (3.4) and (3.5), the maximum change in the coefficient $c_{i,j}$ is $\epsilon i|d_i - j| < \epsilon n(2n) = 2n^2\epsilon$. Therefore, the maximum change in the objective value of an integral feasible solution due to perturbation in the penalties is $n2n^2\epsilon = 2n^3\epsilon$. On the other hand, it can be easily shown that the minimum non-zero difference between objective values of two integral feasible solutions to Problem 2.1 is bounded below by 2^{-L}. Therefore, if $2(2n^3/\epsilon) < 2^{-L}$ (in other words, $\epsilon < 2^{-L}/4n^3$), any optimal solution to the perturbed problem is also optimal to the original problem. ■

We are ready with the main result of the paper.

Corollary 3.5 *If the job penalties meet the dominance condition, Problem 1.1 can be solved in polynomial time.*

Proof : In order to get a polynomial-time algorithm, we need to first perturb the penalties so that the new penalties satisfy the strict dominance condition. If we perturb the penalties in accordance to equations (3.4) and (3.5) with $\epsilon < 2^{-L}/4n^3$, the new penalties satisfy the dominance condition. Moreover, the encoding size of the new penalties remains polynomial in the encoding size of the original penalties. Therefore, in light of Corollary 3.4 and Proposition 3.2, we need only find an extremal optimal solution to the LP-relaxation of Problem 2.1 with the perturbed penalties. The linear programming problem is well-known to be solvable in time polynomial of its encoding size. Since the encoding size of the linear programming formulation (2.1) is polynomial in the encoding size of the original Problem 1.1, we have the result. ■

4 Conclusions

The problem of determining a schedule of jobs with unit-time lengths on a single-machine that minimizes the total weighted earliness and tardiness penalties was formulated as an integer programming problem. If the due dates are integral, then the problem can be simply solved as an assignment problem. However, if the due dates are rational, the problem ceases to be straightforward. We showed that if the penalties meet a certain criterion called the *Dominance Condition* then there exists an extremal optimal solution to the LP-relaxation that is integral, making the problem solvable in polynomial time under these conditions. The general weighted symmetric penalty structure is one cost structure that satisfies the Dominance Condition; we pointed out other commonly found penalties that also fall in this category.

The complexity of Problem 1.1 when the asymmetry is of the non-dominant type is still an open question. It would also be interesting to find a combinatorial algorithm which directly implies the integrality property proved in this paper.

Acknowledgments The authors are grateful to comments and suggestions provided by two anonymous referees and the associate editor which led to significant improvement in the presentation.

References

Baker, K. (1974). *Introduction to Sequencing and Scheduling.* John Wiley and Sons, New York.

Baker, K. and Schrage, L. (1978). Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks. *Operations Research*, **26** 111-120.

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

Dessouky, M., Kijowski, B., and Verma, S. (1998). Simultaneous Batching and Scheduling for Chemical Processing with Earliness and Tardiness Penalties. Technical Report Number 1998-01, University of Southern California, Los Angeles, CA.

Fisher, M. (1976). A dual algorithm for the one-machine scheduling problem. *Mathematical Programming*, **11** 229–251.

Garey, M., Tarjan, R., and Wilfong, G. (1988). One-process scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, **13** 330–348.

Grötschel, M., Lovász, L., and Schrijver, A. (1980). *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, Berlin.

Hall, N. and Posner, M. (1991). Earliness-tardiness scheduling problems i: Weighted deviation of completion times about a common due date. *Operations Research*, **39** 836–846.

Kanet, J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, **28** 643–651.

Kramer, F.-J. and Lee, C.-Y. (1993). Common due-window scheduling. *Production and Operations Management*, **2** 262–275.

Lawler, E. (1964). On scheduling problems with deferral costs. *Management Science*, **11** 280–288.

Lawler, E. (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, **1** 331–342.

Lenstra, J. and Kan, A. R. (1985). *Sequencing and Scheduling. In Combinatorial Optimization: Annotated Bibliographies, O'hEighertaigh et al.* John Wiley & Sons, New York.

Lenstra, J., Kan, A. R., and Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, **1** 343–362.

Murthy, K. G. (1983). *Linear Programming.* John Wiley and Sons.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* John Wiley & Sons.

Ow, P. and Morton, T. (1989). The single machine early/tardy problem. *Management Science*, **35** 177–191.

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

Rachamadugu, R. and Morton, T. (1983). Myopic heuristics for the single machine weighted tardiness problem. working paper 30-82-83. Technical report, Carnegie Mellon University, Pittsburgh, PA.

Royden, H. (1968). *Real Analysis*. Macmillan Publishing Co., Inc.

Schrage, L. and Baker, K. (1978). Dynamic programming solution of sequencing problems with precedence constraints. *Operations Research*, **26** 444–449.

Yano, C. and Kim, Y. (1991). Algorithms for single machine scheduling problems minimizing tardiness and earliness. *European Journal of Operational Research*, **52** 167–178.