Preemption in Single Machine Earliness/Tardiness Scheduling

Kerem Bülbül*
Philip Kaminsky
Candace Yano
Industrial Engineering and Operations Research
University of California, Berkeley, CA

October 2001

Abstract

We consider a single machine earliness/tardiness scheduling problem with general weights, ready times and due dates. Using completion time information obtained from the optimal solution to a preemptive relaxation, we generate feasible solutions to the original non-preemptive problem. We report extensive computational results demonstrating the speed and effectiveness of this approach.

1 Introduction

In the single machine earliness/tardiness (E/T) problem, a set of jobs, each with an associated due date, has to be sequenced on a single machine. Each job has a penalty per unit time associated with completing before its due date, and a penalty per unit time associated with completing after its due date. E/T problems have been popular since the early 1980's because they reflect the JIT philosophy where early shipments are not allowed and result in inventory holding costs in addition to the tardiness penalties associated with the loss of customer goodwill. Although it is a useful problem in its own right, our interest in the single machine E/T problem has been further motivated by its appearance as a subproblem in solution approaches for more complex scheduling problems (see Ovacik and Uzsoy (1997) and Bülbül et al. (2001)). In these more general contexts, it is frequently necessary to solve many such problems very rapidly. As solution speed is very important, we focus on the development of fast, effective heuristics for this problem.

A survey of the early research on earliness/tardiness problems appears in Baker and Scudder (1990). More recent research can be found in Kanet and Sridharan (2000) and references therein. Our approach

*Corresponding author. E-mail: bulbul@ieor.berkeley.edu.

is based on the solution to a preemptive version of this problem. The majority of research on E/T problems has emphasized the single machine problem without preemption, i.e., jobs cannot be interrupted once started. There appear to be two primary reasons for the emphasis on non-preemptive schedules. First, in the traditional scheduling literature, the vast majority of objective functions are *regular*, i.e., the penalty is a non-decreasing function of the completion time of each job. For such objectives, preemption cannot improve the objective function value (OFV) if all ready times are equal. Second, in some settings, preemption is not practical. In other instances, however, a modest degree of preemption can be accommodated, with little or no penalty due to the preemption itself, by simply dividing customer orders into smaller processing batches in an appropriate way.

The complexity and usefulness of preemptive E/T models depend to a large extent on the type of preemption allowed. When preemption is allowed at arbitrary times, the problem turns out to be intractable, as we explain in more detail in the next section. Before doing so, we formally introduce the problem of interest, the single machine weighted E/T problem with distinct due dates, and explain why it motivated us to study preemption in E/T problems.

Consider a non-preemptive single machine scheduling problem with n jobs. Associated with each job $j, j = 1, \ldots, n$ are several parameters: p_j, the processing time for job j; r_j, the ready time for job j; d_j, the due date for job j; ϵ_j, the earliness cost per unit time if job j completes processing before d_j; and π_j, the tardiness cost per unit time if job j completes processing after d_j. We assume that the processing times, ready times and due dates are integral, and without loss of generality, that the ready time of each job is no larger than its due date. The objective is to minimize the sum of costs for all jobs. In particular, let s_j be the time at which job j starts processing. Then, our problem is:

(P1)

$$\min_{\mathbf{s}} \quad \sum_{j=1}^{n} \epsilon_j * \max(0, d_j - (s_j + p_j)) + \pi_j * \max(0, (s_j + p_j) - d_j) \tag{1.1}$$

s.t.

$$s_j \geq r_j \qquad \forall j \tag{1.2}$$

$$s_i + p_i \leq s_j \quad \textbf{or} \quad s_j + p_j \leq s_i \qquad \forall i,\ j \tag{1.3}$$

The objective is to minimize the total weighted earliness and tardiness. The constraints ensure that jobs start at or after their respective ready times and that jobs do not overlap.

In classifying scheduling problems, we follow the three field notation of Graham et al. (1979). Problem P1 is represented as $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$ where in the first field, 1 indicates that there is a single machine, and the entry r_j in the second field denotes that the ready times may be unequal. P1 is $\mathcal{NP}$-hard because the special case obtained by setting $\epsilon_j = 0\ \forall j$, the problem $1/r_j/\sum \pi_j T_j$, is known to be $\mathcal{NP}$-hard (Lenstra et al. (1977)). Therefore, most approaches for P1 are either branch and bound algorithms or dispatch heuristics based on dominance properties of the job sequence. Two important drawbacks of branch and

bound procedures are that they are typically not effective for problems of more than 40 to 50 jobs, and these approaches are computationally quite slow. While dispatch heuristics for this problem are typically quite fast, they frequently provide poor quality solutions.

In this paper, we develop a tight lower bound for P1 based on a preemptive solution that is easy to compute and reveals as much information as possible about the optimal solution to P1. Our primary goal is to use the optimal preemptive solution in developing a good, non-preemptive solution for P1. A secondary goal is to obtain a tight lower bound for specific instances of P1 that can be found quickly, in order to serve as a useful lower bound benchmark for heuristic procedures.

In the next section, we review some general properties of preemptive E/T models that will lead us to an appropriate preemption strategy for P1. In Section 3, we develop a penalty scheme for a preemptive version of model P1 in which each unit-duration portion of a job may incur a non-zero cost. We show that a formulation based on this penalty scheme leads to an excellent lower bound for P1 that can be computed in pseudo-polynomial time. In Section 4, we derive some properties of our new formulation and penalty scheme. In Section 5, we present a heuristic that uses the information from the solution to the preemptive problem to create a feasible solution to P1. Finally, in Section 6, we present the results of computational experiments which demonstrate the effectiveness of our lower bound, and of our heuristic. We conclude in Section 7.

2 Properties of Preemption in E/T Problems

The complexity of a preemptive scheduling problem depends heavily on the penalty scheme applied to the different segments of each job. Assume that job j is preempted $k-1$ times, i.e., it is completed in k portions where the m^{th} portion is completed at time C_{jm}. Then, the total cost incurred by job j is given by $\sum_{m=1}^{k} g_j(C_{jm})$ where g_j is the cost function of job j. Traditionally, the function g_j is chosen such that $g_j(C_{jm}) = 0$ unless $m = k$, i.e., only the last portion of job j incurs a non-zero cost, and hence $\sum_{m=1}^{k} g_j(C_{jm}) = g_j(C_{jk})$.

Below, we discuss general properties of optimal preemptive schedules for single machine problems with *distinct* due dates, in instances where only the last portion of each job may incur a non-zero cost. We first examine the problem with the objective of minimizing only (weighted) tardiness penalties, and then return to the E/T problem.

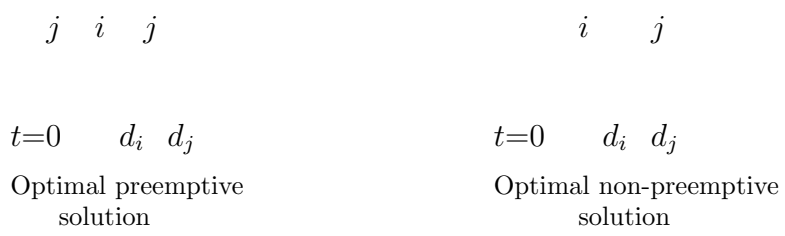
McNaughton (1959) demonstrates that preemption is not useful for obtaining a lower bound in a tardiness problem with equal ready times. He shows that in $1/prmp/\sum_j g(T_j)$ where g is a non-decreasing function of job tardiness, for every preemptive schedule, there exists a non-preemptive schedule whose objective value is no larger. The preemptive schedule is converted into a non-preemptive schedule by completing jobs non-

preemptively in non-decreasing order of the time as which their last portion is completed. This conversion process does not make any job tardier than it was, and thus from the monotonicity of g, it follows that the OFV does not deteriorate. This conversion from a preemptive schedule into a non-preemptive schedule can be carried out in polynomial time if the number of preemptions is finite and the ready times of the jobs are not restrictive.

This polynomial conversion scheme suggests that if we had a polynomial-time algorithm for $1/prmp/\sum \pi_j T_j$ then it would be possible to solve $1//\sum \pi_j T_j$ in polynomial time as well. However, $1//\sum \pi_j T_j$ (Lenstra et al. (1977)), $1/r_j/\sum \pi_j T_j$ (Lenstra et al. (1977)) and $1/r_j, prmp/\sum \pi_j T_j$ (Labetoulle et al. (1984)) are all known to be $\mathcal{NP}$-hard.

These observations imply that conventional preemptive problems whose objectives are non-decreasing in job tardiness are as hard as their non-preemptive counterparts. Furthermore, the optimal objective function values of corresponding preemptive and non-preemptive problems are the same except when the ready times are restrictive.

In contrast, for an objective function of the form $\sum_j [g_{1j}(E_j) + g_{2j}(T_j)]$, preemption may decrease the objective function even if all ready times are equal. Consider an example with $r_i = r_j = 0$, $d_i = 2$, $d_j = 3$, $p_i = 1$, $p_j = 2$, and with the objective of minimizing $\sum_j (E_j + T_j)$. The Gannt charts in Figure 1 show that the optimal preemptive schedule has a total cost of zero whereas the optimal non-preemptive schedule has a total cost of 1.



Optimal preemptive solution

Optimal non-preemptive solution

Figure 1: Preemption might lower OFV in E/T models

Now consider a preemptive E/T problem in which only the last portion of a job may incur a non-zero cost. If all due dates are distinct, then for any early job j, it is possible to schedule an arbitrarily small portion of the job of duration $\delta > 0$ such that it completes at d_j, and job j incurs a cost of zero (Mosheiov (1996)). If there are many early jobs with the same due date, then the total cost from earliness cannot be zero, but can be made arbitrarily small. This implies that one can treat a preemptive E/T problem as a preemptive tardiness problem, but as mentioned above, all preemptive tardiness problems of interest to us are $\mathcal{NP}$-hard, except for $1/r_j, p_j = 1/\sum \pi_j T_j$ which is equivalent to a transportation problem. Note that this last problem is polynomially solvable due to the *unit processing times*.

These observations imply that a preemptive E/T problem in which only the last portion of a job may incur a non-zero cost is unlikely to provide a tight lower bound on P1. Also encouraged by the observation that $1/r_j, p_j = 1/\sum \pi_j T_j$ has a polynomial time complexity because all processing times are equal, we next explore a preemptive lower bound for P1 in which jobs are divided into unit intervals and each portion of a job may incur a non-zero cost.

3 A Lower Bound For P1

Our approach for constructing a lower bound builds upon the ideas of Gelders and Kleindorfer (1974) for the single machine weighted tardiness problem and the results of Verma and Dessouky (1998) and Dessouky et al. (1999) for the non-preemptive single machine problem with the objective of minimizing weighted earliness and tardiness. To obtain a lower bound, Gelders and Kleindorfer divide each job into unit-duration jobs and associate a cost with each unit job. We utilize their idea of unit-duration jobs. Their other results, however, cannot be used directly because of the differences between the objective functions and the fact that non-delay schedules are optimal in weighted tardiness problems, but may be suboptimal when earliness costs are considered. We discuss these differences in more detail after presenting our approach.

We first present some properties of the optimal solution to P1 which will lead us to a transportation problem whose optimal objective value is a lower bound on the optimal objective of P1. The following result is similar to Proposition 2.1 in Verma and Dessouky (1998), who characterize optimal non-preemptive schedule of jobs with unit processing times on a single machine when the objective is to minimize total weighted earliness and tardiness.

Property 3.1 *There exists an optimal solution to P1 in which all job completion times are integral if processing times, due dates and ready times are integral.*

Proof. Observe that any feasible schedule for P1 consists of one or more blocks of jobs such that jobs within a block are scheduled without inserted idle time, and there is idle time between blocks, if more than one block exists. For any arbitrary block, the associated objective value is a piecewise linear, convex function of the start time of the first job in the block. Moreover, because due dates and processing times are integral, the slope of the objective function changes only at a subset of the integer time points. Thus, the objective function for the block achieves a minimum either at a unique integral time point, or at two or more consecutive integral time points and at all points in between if the solution is not unique. (If the resultant solutions for the individual blocks lead to concatenation or overlapping of blocks, the blocks are merged and the same results apply to the merged block.) ∎

Property 3.1 indicates that, without loss of optimality, the solution space can be restricted to integer start

times. Intuitively, an approximation to P1 could be obtained by dividing each job j into p_j unit-duration jobs, associating a penalty with each unit job (rather than only with the last one completed) and planning for a horizon consisting of an appropriate number of integral time periods. Before doing so, we first restate P1 as a binary integer program that is equivalent to P1. We then discuss modifications to the problem that lead to a relaxation from which we derive a lower bound. In the discretized versions of this problem, a period k spans from time $k-1$ to k, hence a job j with ready time r_j can first be processed in period $r_j + 1$.

Let $X_{jk} = 1$ if job j is processed in period k, 0 otherwise, and $(x)^+ = \max(x,\ 0)$. Then the problem becomes:

$$\min \quad \sum_j \sum_{k \in H} [\epsilon_j (d_j - k)^+ + \pi_j (k - d_j)^+](X_{jk} - X_{jk+1})^+$$

s.t.

(P2)

$$\sum_{k \in H} X_{jk} = p_j \qquad \forall j \tag{3.1}$$

$$\sum_j X_{jk} \leq 1 \qquad \forall k \in H \tag{3.2}$$

$$\sum_{k \in H} (X_{jk} - X_{jk+1})^+ = 1 \qquad \forall j \tag{3.3}$$

$$X_{jk} \text{ binary} \qquad \forall j,\ k \in H$$

Let $t_{min} = \min_j\ r_j + 1$, $t_{max} = \max_j\ d_j + P$, where $P = \sum_j p_j$ is the sum of processing times, and define the planning horizon $H = \{k | k \in \mathbb{Z},\ t \in [t_{min}, t_{max}]\}$. Thus the maximum value of k in H is greater than or equal to the latest period in which any portion of any job could conceivably be processed, accounting for potential idle time. X_{jk+1} is defined as zero if k is the last time period in the planning horizon, so the term in square brackets in the objective function is the cost incurred if job j finishes in period k. In the non-preemptive problem, the expression $(X_{jk} - X_{jk+1})^+$ takes on the value 1 only if k is the completion time of the job and is zero otherwise. Constraints (3.3), referred to as contiguity constraints, together with constraints (3.1), ensure that each job j is processed in p_j consecutive periods. The machine can process at most one job in any given period as specified by constraints (3.2).

The formulation of our lower bounding problem is obtained by omitting the contiguity constraints and replacing the objective function with a simpler expression in which each unit-duration portion of each job has an associated cost coefficient. With this new objective, the model has the form of a transportation problem, so the binary constraints do not need to be made explicit, and the problem can be solved very efficiently. We first present the formulation, then discuss the cost coefficients.

$$\min \quad \sum_{j} \sum_{\substack{k \in H \\ k \geq r_j + 1}} c_{jk} X_{jk} \tag{3.4}$$

s.t.

(TR1)

$$\sum_{\substack{k \in H \\ k \geq r_j + 1}} X_{jk} = p_j \quad \forall j \tag{3.5}$$

$$\sum_{j,\ k \geq r_j + 1} X_{jk} \leq 1 \quad \forall k \in H \tag{3.6}$$

$$X_{jk} \geq 0 \qquad \forall j,\ k \in H,\ k \geq r_j + 1 \tag{3.7}$$

The idea underlying our proposed cost structure is intuitive. If job j finishes exactly at d_j, and if we think of each of the p_j unit-duration segments in job j as different jobs, then the average completion time of those unit-duration segments is

$$\frac{(d_j - p_j + 1) + ... + d_j}{p_j} = d_j - \frac{1 + ... + (p_j - 1)}{p_j} = d_j - \frac{(p_j - 1)p_j}{2p_j} = d_j - \left(\frac{p_j}{2} - \frac{1}{2}\right)$$

We treat $d_j - \left(\frac{p_j}{2} - \frac{1}{2}\right)$, this average time, as the common due date for all unit jobs within job j, and assign the following cost coefficients:

$$c_{jk} = \begin{cases} \frac{\epsilon_j}{p_j}\left[(d_j - \frac{p_j}{2}) - (k - \frac{1}{2})\right] & \text{if } k \leq d_j \\ \frac{\pi_j}{p_j}\left[(k - \frac{1}{2}) - (d_j - \frac{p_j}{2})\right] & \text{if } k > d_j \end{cases} \tag{3.8}$$

Let S_P represent a feasible schedule for problem P with a total cost of $C(S_P)$. An optimal schedule is denoted by an asterisk. Now, although it may not be immediately obvious, we prove that:

Theorem 3.2 *The optimal objective value $C(S^*_{TR1})$ of TR1 is a lower bound on the optimal objective value $C(S^*_{P1})$ of P1.*

To show this, we prove that for any optimal solution S^*_{P1} of P1, the *same* schedule S_{TR1} is feasible for TR1 and has a cost that is less than or equal to that of S^*_{P1}. Thus, the cost of the optimal solution for TR1 is also a lower bound on the cost of S^*_{P1}. To show the first cost relationship, we consider each job j separately and show that for each possible completion time C_j of job j, the cost incurred in S_{TR1} is less than or equal than that incurred in S^*_{P1}. The critical result is that unless job j finishes in the time interval $[d_j + 1,\ d_j + p_j - 1]$, the cost job j incurs in S^*_{P1} and S_{TR1} are equal to each other. The detailed proof is in the appendix.

As noted earlier, Gelders and Kleindorfer (1974) use a related approach to find a lower bound for the non-preemptive weighted tardiness problem. In contrast to their model, however, we must account for the possibility that in the solution of TR1, for job j of duration $p_j > 1$, some unit jobs may be early, and some others may be tardy, which complicates the task of finding an appropriate cost structure.

Also, their objective function is regular, and consequently, they can restrict their attention to non-delay schedules without loss of optimality. Thus, their planning horizon is of length $C_{max} = P = \sum_j p_j$ if

$r_j = 0,\ \forall j$. However, in TR1, we may need a planning horizon up to t_{max}, even if all ready times are equal, because of the possibility of idle time in the optimal schedule. (It is possible to obtain tighter bounds on the horizon length, but this bound suffices for our current purposes.)

In TR1, there are $n * (t_{max} - t_{min} + 1)$ variables and $n + (t_{max} - t_{min} + 1)$ constraints. It is possible, however, to reduce the size of the problem significantly based on the following lemma which is similar to Lemma 2.1 in Verma and Dessouky (1998).

Lemma 3.3 *There exists an optimal solution to TR1 such that* $\forall j, X_{jk} = 0$ *if* $k \notin H_j$ *where* $H_j = \{k | k \in \mathbb{Z},\ k \in [\max(r_j + 1,\ d_j - P + 1),\ d_j + P]\}$.

The proof is in the appendix.

Lemma 3.3 implies that the following transportation problem, TR2, is equivalent to TR1 and has at most $n + 2nP$ constraints and at most $2nP$ variables.

(TR2)

$$\min \quad \sum_j \sum_{H_j} c_{jk} X_{jk} \tag{3.9}$$

s.t.

$$\sum_{H_j} X_{jk} = p_j \qquad \forall j \tag{3.10}$$

$$\sum_{j,\ k \in H_j} X_{jk} \leq 1 \quad \forall k \in H = \cup_j H_j \tag{3.11}$$

$$X_{jk} \geq 0 \qquad \forall j,\ k \in H_j$$

Theorem 3.4 *The optimal objective value of TR2 is a lower bound on the optimal objective of P1.*

Proof. Follows from Theorem 3.2 and Lemma 3.3. ∎

Thus, it is possible to obtain a lower bound for P1 from a transportation problem that is generated in pseudo-polynomial time from P1 but can be solved optimally in polynomial time once it is constructed. Because TR1 and TR2 lead to equivalent solutions we refer to these formulations collectively as TR when it is not necessary to make a distinction.

We observe that the form of preemption that we consider is often implemented in practical applications due to the nature of the manufacturing process or the structure of the work schedule. Thus, although our study was motivated partly by theoretical considerations, the results may be useful in practice.

4 Properties of the Lower Bound

In this section, we partially characterize the effectiveness of our proposed lower bound, $C(S^*_{TR})$. Observe that any gap between the cost of the optimal transportation solution S^*_{TR} and the optimal solution S^*_{P1} is due to both preemption in the transportation solution, and the different cost structures. In this section, we explore the effect of different cost structures. In order to do this, we first identify a pathological case, for which the cost of the optimal transportation solution is arbitrarily smaller than the cost of the optimal solution to the original problem, and make a sufficient assumption so that this case cannot occur. Next, we characterize the largest possible objective function difference between two identical schedules for problems TR and P1, and then we prove that the cost function that we have chosen for TR is in some sense the best possible cost function for this problem.

4.1 A Pathological Case

Consider a single job instance of problem P1. Clearly, if there is only a *single* job j and $r_j + p_j \leq d_j$ then the optimal solution to P1, S^*_{P1}, is to schedule job j in the interval $[d_j - p_j,\ d_j]$ with a total cost of zero for all possible values of $\epsilon_j > 0$ and $\pi_j > 0$. However, this is not necessarily the case for the optimal transportation solution to this single job instance. In the next lemma, for this very simple instance of P1, we characterize the conditions under which the optimal solutions of P1 and TR differ. Refer to the appendix for the proof.

Lemma 4.1 *Consider a single job instance of P1, with job j, $p_j > 1$, and $r_j + p_j \leq d_j$. Let $\epsilon_j = \beta * \pi_j$. If $\beta \leq 1$, then for all $\epsilon_j > 0$, we have $\pi_j > 0$, and the optimal solution S^*_{TR} to TR is the same as S^*_{P1} with $C(S^*_{TR}) = C(S^*_{P1}) = 0$. If $\beta > 1$ and $\frac{\epsilon_j - \pi_j}{2*(\epsilon_j + \pi_j)} * p_j \geq 1$, then $C(S^*_{TR}) < 0$.*

A direct consequence of this lemma is that one can create single job instances of P1 in which one can make the difference $C(S^*_{P1}) - C(S^*_{TR})$ arbitrarily large by increasing β. Indeed, we characterize this difference as β goes to infinity in the following corollary, the proof of which can be found in the appendix:

Corollary 4.2 *For the single job instance described above, let $\epsilon_j = \beta * \pi_j$. If p_j is odd then*

$$\lim_{\beta \to \infty} C(S^*_{TR}) = \frac{\pi_j}{8p_j}\beta(1 - p_j^2) + \frac{\pi_j}{8p_j} * (1 + 3p_j^2 - 4p_j).$$

If p_j is even then

$$\lim_{\beta \to \infty} C(S^*_{TR}) = \frac{\pi_j p_j}{8} * (3 - \beta).$$

To avoid these pathological cases, we make the following easily justifiable assumption for the rest of the paper.

Assumption 4.3 *For each job j, $\epsilon_j \leq \pi_j$.*

Under this assumption the cost of loss of customer goodwill per unit time is at least as expensive as the finished good inventory holding cost per unit time.

4.2 Cost Function Difference for Identical Schedules

In Section 3, we proved that $C(S^*_{TR})$ is a lower bound on $C(S^*_{P1})$. In this section, we give two characterizations of the portion of the gap attributable to differences between the objective functions of P1 and TR. To motivate the discussion, suppose that we develop an algorithm that converts S^*_{TR} into a non-preemptive schedule S_{P1}, and we are interested in computing a bound on the worst case difference of $C(S_{P1})$ and $C(S^*_{TR})$. To accomplish this, one would need to account for both the increase in the objective of TR while constructing S_{P1} from S^*_{TR} and the increase in cost from applying the original objective function to S_{P1}. The following two lemmas characterize the latter increase. The first characterizes the absolute difference, and the second characterizes the relative difference in cost when the TR and P1 objective functions are applied to the same (not necessarily optimal) non-preemptive schedule. The proofs are in the appendix.

Lemma 4.4 *Let S_{P1} be a feasible schedule to the original non-preemptive problem P1 with cost $C(S_{P1})$, and let S_{TR} be the corresponding transportation solution with cost $C(S_{TR})$. Then,*

$$C(S_{P1}) - C(S_{TR}) \leq \frac{1}{8}\sum_{j=1}^{n}(\pi_j + \epsilon_j)p_j.$$

Lemma 4.5 *Let S_{P1} be a feasible schedule to the original non-preemptive problem P1 with cost $C(S_{P1})$, and let S_{TR} be the corresponding transportation solution with cost $C(S_{TR})$. Then, under Assumption 4.3,*

$$0 \leq C(S_{TR}) \leq C(S_{P1}) \leq p_{max} * C(S_{TR}).$$

Observe that according to Lemma 4.5, the optimal objective function value of TR is the same as that of P1 (see Verma and Dessouky (1998)) if all jobs are of unit length because $p_{max} = 1$ in that case. Also, the bound given in Lemma 4.5 has the nice property that it is independent of the earliness and tardiness costs. This property is not true of the bound on the absolute difference $C(S_{P1}) - C(S_{TR})$ presented in Lemma 4.4.

4.3 Optimal Linear Cost Coefficient Function

In the previous subsection, we observed that the difference between the cost function of TR and the cost function of P1 can be relatively large in the worst case. A natural question to ask is whether we can find cost coefficients that will decrease the difference between the costs incurred by the same non-preemptive

schedule in P1 and in TR, and thereby improve the bounds given in Lemma 4.4 and Lemma 4.5. We restrict ourselves to *linear* cost coefficient functions as defined below. This choice has a natural appeal for three reasons. First, the original problem P1 also has a linear penalty structure. Second, the lower bound obtained from TR based on the piecewise linear cost structure in (3.8), and the non-preemptive schedules adapted from S^*_{TR} perform very well, as we demonstrate in the computational experiments. Finally, linearity enables us to use linear programming duality to prove our result. Specifically, we look for cost coefficients c_{jk} for job j that satisfy the following conditions where m_E and m_T represent the slope of the cost function before and after the due date d_j, respectively.

$$c_{jk} = c_{jd_j} + (d_j - k) * m_E \quad \forall k \leq d_j \tag{4.1}$$

$$c_{jk} = c_{jd_j+1} + (k - d_j - 1) * m_T \quad \forall k \geq d_j + 1 \tag{4.2}$$

$$m_E,\ m_T \geq 0 \tag{4.3}$$

Next, we need to choose an appropriate objective function that measures how *good* the cost coefficients are. Note that all we know about the completion time C_j of job j in an optimal non-preemptive schedule is that it lies within some interval H (see section 3). This implies that we need cost coefficients that perform well over a range of possible completion times rather than for a specific completion time. Now, assume that C_j lies in some interval $[t^j_{min},\ t^j_{max}]$ that includes d_j. As alternatives for the objective function, we consider minimizing the maximum or the total difference of $C_j(S_{P1}) - C_j(S_{TR})$ across all C_j values in the interval $[t^j_{min},\ t^j_{max}]$ where $C_j(S_P)$ denotes the cost incurred by job j in problem P. We observed computationally that for the minimax criterion the optimal cost coefficients are dependent on the choices of t^j_{min} and t^j_{max}. However, as we prove in this section, under some minor conditions the optimal cost coefficients for the minisum objective are not contingent on the specific values of these two parameters. Additionally, by solving several instances, we also observed that the optimal solution for the minimax criterion approaches the optimal solution of the minisum criterion when $t^j_{max} - t^j_{min}$ increases. Therefore, we use the minisum criterion, and characterize the structure of its optimal solution in the following discussion.

The problem of finding cost coefficients that satisfy (4.1) through (4.3) and minimize the sum of differences, $\sum_{C_j \in [t^j_{min},\ t^j_{max}]} [C_j(S_{P1}) - C_j(S_{TR})]$, over all possible completion times C_j of job j when it is scheduled non-preemptively to finish at the same time both in S_{P1} and S_{TR}, can be represented as a linear program. Let a_E and a_T denote c_{jd} and c_{jd+1}, respectively, and D_t be $C_j(S_{P1}) - C_j(S_{TR})$ if job j finishes processing at time t. Also, t^j_{min} and t^j_{max} are the smallest and largest possible completion times of job j, respectively. In general, we allow $t^j_{min} \geq r_j + p_j$ and $t^j_{max} \leq d_j + P$, and later in the discussion we identify some minor conditions to be imposed on t^j_{min} and t^j_{max}. In the following linear program we drop the job index j for notational convenience.

$$
\begin{array}{lll}
\min & \sum_{t=t_{min}}^{t_{max}} D_t & \\
\text{s.t.} & & \\
(\gamma_t) & D_t + \sum_{k=t-p+1}^{t} (a_E + (d-k)m_E) = \epsilon(d-t) & t = t_{min}, \ldots, d \quad (4.4) \\
(\delta_t) & D_t + \sum_{k=t-p+1}^{d} (a_E + (d-k)m_E) + \sum_{k=d+1}^{t} (a_T + (k-d-1)m_T) = \pi(t-d) & t = d+1, \ldots, d+p-1 \quad (4.5) \\
(\mu_t) & D_t + \sum_{k=t-p+1}^{t} (a_T + (k-d-1)m_T) = \pi(t-d) & t = d+p, \ldots, t_{max} \quad (4.6) \\
 & a_E,\ a_T \quad \text{unrestr.} & \\
 & m_E,\ m_T \geq 0 & \\
 & D_t \geq 0 & t = t_{min}, \ldots, t_{max} \quad (4.7)
\end{array}
$$

(CP)

The constraints (4.4) through (4.6) define the differences D_t between the cost functions of TR and P1 for different completion times of job j. By (4.7), $C_j(S_{P1}) - C_j(S_{TR})$ is guaranteed to be nonnegative for all possible completion times of job j. Thus, $C(S^*_{TR})$ yields a lower bound on $C(S^*_{P1})$ for all objective function coefficients c_{jk} generated from any feasible solution of CP. The Greek letters on the left are the dual variables associated with the constraints. Now, we show that the optimal solution to CP corresponds to the cost coefficients (3.8) when $t_{min} \leq d - p$, and $t_{max} \geq d + 2p$. In practice, these conditions are automatically satisfied for the vast majority of problems. From the definition of the planning horizon H, it follows that all we need is $r_j + p_j \leq d_j - p_j$ and $d_j + P \geq d_j + 2p_j$, i.e., $d_j - r_j \geq 2p_j$ and $P \geq 2p_j$.

Theorem 4.6 *If $t_{min} \leq d - p$, and $t_{max} \geq d + 2p$, then the optimal solution to CP is given by $m_E = \frac{\epsilon}{p}$, $m_T = \frac{\pi}{p}$, $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, and $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$ which correspond to the cost coefficients*

$$
c_k = \begin{cases} \frac{\epsilon}{p}\left[(d - \frac{p}{2}) - (k - \frac{1}{2})\right] & \text{if } k \leq d \\ \frac{\pi}{p}\left[(k - \frac{1}{2}) - (d - \frac{p}{2})\right] & \text{if } k > d \end{cases}
$$

given in (3.8). *The optimal objective function value is $\frac{1}{12}(\epsilon + \pi)(p^2 - 1)$.*

Proof. Our strategy is to find a dual feasible solution for which the dual objective value equals the objective value given above. It follows directly from Theorem 3.2 that $m_E = \frac{\epsilon}{p} \geq 0$, $m_T = \frac{\pi}{p} \geq 0$, $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, and $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$ and the corresponding D_t values are feasible for CP. The resulting objective function value is $\sum_{t=t_{min}}^{t_{max}} D_t = \sum_{x=1}^{x=p-1}(\pi x - f(x)) = \frac{1}{12}(\epsilon + \pi)(p^2 - 1)$ using the identities $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$, $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$ and the expression $f(x) = \frac{1}{2}\left(\frac{\epsilon_j}{p_j} + \frac{\pi_j}{p_j}\right)x^2 + \frac{1}{2}(\pi_j - \epsilon_j)\,x$ which was derived for Lemma 4.1.

The dual of **CP** is:

(DCP)

$$\max \quad \sum_{t=t_{min}}^{d} \epsilon(d-t)\gamma_t + \sum_{t=d+1}^{d+p-1} \pi(t-d)\delta_t + \sum_{t=d+p}^{t_{max}} \pi(t-d)\mu_t$$

s.t.

$$(D_t) \quad \gamma_t \leq 1 \qquad t = t_{min}, \ldots, d \tag{4.8}$$

$$(D_t) \quad \delta_t \leq 1 \qquad t = d+1, \ldots, d+p-1 \tag{4.9}$$

$$(D_t) \quad \mu_t \leq 1 \qquad t = d+p, \ldots, t_{max} \tag{4.10}$$

$$(a_E) \quad \sum_{t=t_{min}}^{d} p\gamma_t + \sum_{t=1}^{p-1}(p-t)\delta_{d+t} = 0 \tag{4.11}$$

$$(a_T) \quad \sum_{t=1}^{p-1} t\delta_{d+t} + \sum_{t=d+p}^{t_{max}} p\mu_t = 0 \tag{4.12}$$

$$(m_E) \quad \sum_{t=t_{min}}^{d} \sum_{k=t-p+1}^{t} (d-k)\gamma_t + \sum_{t=d+1}^{d+p-1} \sum_{k=t-p+1}^{d} (d-k)\delta_t \leq 0 \tag{4.13}$$

$$(m_T) \quad \sum_{t=d+1}^{d+p-1} \sum_{k=d+1}^{t} (k-d-1)\delta_t + \sum_{t=d+p}^{t_{max}} \sum_{k=t-p+1}^{t} (k-d-1)\mu_t \leq 0 \tag{4.14}$$

$$\gamma_t \quad \text{unrestr.} \quad t = t_{min}, \ldots, d$$

$$\delta_t \quad \text{unrestr.} \quad t = d+1, \ldots, d+p-1$$

$$\mu_t \quad \text{unrestr.} \quad t = d+p, \ldots, t_{max}$$

The corresponding primal variable for each constraint is indicated in the parentheses on the left. In the appendix, we prove the following Lemma:

Lemma 4.7 *If $t_{min} \leq d-p$, and $t_{max} \geq d+2p$, then the following solution is feasible for DCP.*

$$\gamma_{t_{min}} = -\gamma_d - \frac{p-1}{2} - K + 1 \qquad \text{where } K = d - t_{min} \tag{4.15}$$

$$\gamma_t = 1 \qquad t_{min}+1 \leq t \leq d-1 \tag{4.16}$$

$$\gamma_d = \frac{-(p+1)(p-1)}{12K} - \frac{K+p}{2} + 1 \tag{4.17}$$

$$\delta_t = 1 \qquad d+1 \leq t \leq d+p-1 \tag{4.18}$$

$$\mu_{d+p} = \frac{\frac{N^2}{2} + N(\frac{-p}{2} - 1) + (p-1)(\frac{p+13}{12}) + 1}{-N+p} \qquad \text{where } N = t_{max} - d \tag{4.19}$$

$$\mu_t = 1 \quad d+p+1 \leq t \leq t_{max}-1 \tag{4.20}$$

$$\mu_{t_{max}} = -\mu_{d+p} - \frac{p-1}{2} - N + p + 1 \tag{4.21}$$

Next, we compute the objective function value for this dual feasible solution to complete the proof. First we compute the following three expressions.

$$\begin{aligned}
\sum_{t=t_{min}}^{d} (d-t)\gamma_t &= (d-t_{min})\gamma_{t_{min}} + \sum_{t=1}^{d-t_{min}-1} t \\
&= K\left(\frac{(p+1)(p-1)}{12K} + \frac{K+p}{2} - 1 - \frac{p-1}{2} - K + 1\right) + \frac{K(K-1)}{2} \\
&= \frac{(p+1)(p-1)}{12}
\end{aligned}$$

$$\begin{aligned}
\sum_{t=d+1}^{d+p-1}(t-d)\delta_t &= \sum_{t=1}^{p-1} t = \frac{p(p-1)}{2} \\
\sum_{t=d+p}^{t_{max}}(t-d)\mu_t &= p\mu_{d+p} + (p+1) + (p+2) + \ldots + (N-1) + N\mu_{t_{max}} \\
&= p\mu_{d+p} + (N-p-1)p + \sum_{t=1}^{N-p-1} t + N(-\mu_{d+p} - \frac{p-1}{2} - (N-1-p)) \\
&= \frac{(p+1)(p-1)}{12} - \frac{p(p-1)}{2}
\end{aligned}$$

Therefore, the objective function value for the dual feasible solution (4.15)-(4.21) is

$$\begin{aligned}
\sum_{t=t_{min}}^{d}\epsilon(d-t)\gamma_t &+ \sum_{t=d+1}^{d+p-1}\pi(t-d)\delta_t + \sum_{t=d+p}^{t_{max}}\pi(t-d)\mu_t \\
&= \epsilon\frac{(p+1)(p-1)}{12} + \pi\left(\frac{p(p-1)}{2} + \frac{(p+1)(p-1)}{12} - \frac{p(p-1)}{2}\right) \\
&= \frac{1}{12}(\epsilon+\pi)(p+1)(p-1) = \frac{1}{12}(\epsilon+\pi)(p^2-1)
\end{aligned}$$

which is equal to the objective function value of the primal feasible solution $m_E = \frac{\epsilon}{p}$, $m_T = \frac{\pi}{p}$, $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$ and the corresponding D_t values. ∎

5 The Switch Heuristic (SW)

Optimal schedules for E/T problems often contain unforced idle time due to the presence of earliness costs in their objective functions, and consequently even if the job processing sequence is given, one needs a procedure to determine the optimal job start times. This is commonly referred to as *timetabling*. However, it turns out that timetabling is a rather easy problem, and the total cost is predominantly determined by the job sequence. Therefore, in this section, we focus on an algorithm to construct a feasible solution with a good job processing sequence for P1 starting from an optimal solution to TR, and discuss timetabling in more detail in section 6.1. Before stating the algorithm, we first derive some properties of an optimal solution to TR.

As in the original non-preemptive problem P1, any solution to TR can be characterized by "blocks" of processing with idle time between adjacent blocks. In the context of TR, we define a block B as a sequence of *unit* jobs processed contiguously. Below, we state a simple but important property regarding the block structure of an optimal solution to TR.

Property 5.1 *In an optimal solution S^*_{TR} to TR, if a unit job u_j of job j is processed in block B that starts at t_B and finishes at $t_B + l_B$, then $t_B \leq d_j \leq t_B + l_B$.*

Proof. By contradiction. Assume $d_j < t_B$. Then u_j is tardy and can be moved into the interval $[t_B - 1,\ t_B]$. This would decrease the cost of schedule S^*_{TR} contradicting its optimality. The case $d_j > t_B + l_B$ follows analogously. ∎

Thus, in S^*_{TR}, all p_j unit jobs of job j will be processed in the same block. One can therefore construct a feasible solution to P1 by re-arranging the unit jobs within each block independently of the other blocks and without changing the start and completion times of the blocks.

Trivially, if all ready times are equal, any sequence of jobs in B, scheduled non-preemptively and without idle time starting at t_B, yields a feasible schedule to P1. However, we would like to utilize as much information as possible from S^*_{TR} while creating a non-preemptive schedule S_{P1}.

Algorithm 1 The Switch Heuristic

Require: Given an optimal transportation schedule S^*_{TR}, run this algorithm for all blocks B.

Initialize:
For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^{TR^*}_{j_i}$.
Let Q_B be the set of split jobs in B.
Main routine:
while $Q_B \neq \emptyset$ **do**
  $k = \arg\min_{j \in Q_B}\{C_{j_{[1]}}\}$.
  SHIFT-EARLY(B, k), let the resulting schedule be S^E_{TR}. {Try early and late shifts.}
  SHIFT-LATE(B, k), let the resulting schedule be S^L_{TR}.
  $\Delta^E = C(S^E_{TR}) - C(S^*_{TR})$ {Compute cost increases.}
  $\Delta^L = C(S^L_{TR}) - C(S^*_{TR})$
  if $\Delta^E \leq \Delta^L$ **then**
    For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^E_{j_i}$.
  else
    For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^L_{j_i}$.
  end if
  Update Q_B. {A single shift can bring together several split jobs.}
end while

We have observed from the optimal solutions of S^*_{P1} and S^*_{TR} for many instances of this problem, that within a block, those jobs that contribute most to the overall cost are scheduled very close to the same time in both schedules, while less expensive jobs are frequently spread out in S^*_{TR}. This observation motivates the Switch Heuristic (see Algorithm 1, as well as the subroutines Algorithms 2 and 3). In particular, this heuristic chooses a job that is preempted at least once by another job (called a *split* job) in the *current* schedule, and rearranges its unit jobs while attempting to minimize the increase in the total cost of the block from its original value in S^*_{TR}. For a split job, all of its unit jobs are moved either next to its first or last unit

job completed. The algorithm tends to alternate between split jobs that are shifted early and late in order to increase the total cost of the block as little as possible. Thus, we allow less expensive, more spread out jobs in S^*_{TR} to be shuffled around, while in the final non-preemptive schedule more expensive jobs remain close to their original locations in S^*_{TR}.

Algorithm 2 Subroutine SHIFT-EARLY(B, k)

Initialize:
For each job $j \in B$ and for each segment j_i associated with j, $C^E_{j_i} = C_{j_i}$.
$t_s = C_{k_{[1]}},\ t_f = C_{k_{[p_j]}}$
Main routine:
for $t = t_s + 1$ **to** $t_f - 1$ **do** {add the non-k jobs to the queue in the correct order}
  if $J(t) \neq k$ **then**
    Enqueue(K, B(t))
  end if
end for
for $t = 1$ **to** $p_j - 1$ **do** {Shift job earlier}
  $C^E_{k_{[t+1]}} \leftarrow t_s + t$
end for
for $t = t_s + p_j$ **to** t_f **do** {remove the non-k jobs from the queue in the correct order}
  $j_i =$ **Dequeue(K)**
  $C^E_{j_i} = t$
end for

In the Switch Heuristic, for all jobs in block B, unit job $j_{[i]}$ is the i^{th} unit job associated with job j, and $C^{TR^*}_{j_{[i]}}$ is the completion time of the i^{th} segment of job j in schedule S^*_{TR}. In particular, this means that $C^{TR^*}_{j_{[1]}}$ represents the completion time of the earliest segment of job j scheduled in S^*_{TR}, while $C^{TR^*}_{j_{[p_j]}}$ is the completion time of the latest segment of job j. At each iteration of the algorithm, we select a split job, and consider either moving all of the segments of the job later to immediately precede its last unit job, or moving all of the segments of the job earlier to immediately follow its first unit job. In the former case the last unit job, and in the latter case the first unit job, is kept fixed at its current location. In both cases, the job is no longer split.

The two subroutines of the Switch Heuristic, Algorithm 2: Shift-Early and Algorithm 3: Shift-Late, shift a job adjacent to its earliest unit job, or its latest unit job, respectively. We define a temporary queue K, where the sorting discipline in the queue is first in, first out, and note that $Enqueue(K, j_i)$ adds unit job j_i to the queue, and $Dequeue(K)$ removes the first unit job in the queue. In addition, let $B(t)$ be the unit job j_i that is processed during the time interval $[t-1,\ t]$ in the current schedule, and let $J(t)$ be the associated job j. Note that the SHIFT-LATE procedure needs to check whether the schedules of the other jobs remain

feasible.

The algorithm will terminate in at most $\mid Q_B \mid$ iterations for each block because each call to the shift subroutines brings at least one split job together and never splits a job that has already been scheduled non-preemptively. Finally, when the Switch Heuristic terminates, we recover the job processing sequence and input it into an optimal timetabling algorithm to obtain the final non-preemptive schedule.

Algorithm 3 Subroutine SHIFT-LATE(B, k)

Initialize:
For each job $j \in B$ and for each segment j_i associated with j, $C^L_{j_i} = C_{j_i}$.
$t_s = C_{k_{[1]}},\ t_f = C_{k_{[p_j]}}$
Main routine:
for $t = t_s + 1$ **to** $t_f - 1$ **do** {add the non-k jobs to the queue in the correct order}
  if $J(t) \neq k$ **then**
    Enqueue(K, B(t))
  end if
end for
for $t = 1$ **to** $p_j - 1$ **do** {Shift job later}
  $C^L_{k_{[p_j - t]}} \leftarrow t_f - t$
end for
for $t = t_s$ **to** $t_f - p_j$ **do** {remove the non-k jobs from the queue in the correct order}
  $j_i =$ **Dequeue(K)**
  if feasible **then** {Is moving the job earlier feasible?}
    $C^L_{j_i} = t$
  else
    $C^L_{j_i} = \infty$
  end if
end for

6 Computational Experiments

We performed a variety of computational experiments in order to evaluate the performance of both the lower bound and a set of heuristics for P1. The set of heuristics consists of the Switch Heuristic, described in Section 5, as well as the following simple heuristics, each of which constructs a sequence based on elements of the optimal solution to the transportation problem, which, in general, is a preemptive schedule. To each sequence we apply an optimal timetabling algorithm to determine the final schedule (see section 6.1).

- Heuristic LCT: Sequence jobs in non-decreasing order of the completion times of their last unit jobs, where the completion time of the last unit job of job j is defined as $\max_{s.t.\ X_{jk}=1} k$.
- Heuristic ACT: Sequence jobs in non-decreasing order of the average completion time of their unit jobs.
- Heuristic MCT: Sequence jobs in non-decreasing order of the median completion time of their unit jobs.

We designed our computational experiments with two main objectives. First of all, we would like to demonstrate that our algorithms find good solutions and find them rapidly although the size of the transportation problem is potentially quite large because it depends on the sum of the processing times (section 3).

Our other major concern is the robustness of our algorithms. In particular, we use the information from a preemptive relaxation to generate non-preemptive schedules, and among the factors that affect preemption in the transportation model are the processing times and the number of jobs. Thus, we are interested in the behavior of our lower bound and the heuristics as processing times and the number of jobs increase.

In section 4, we identified some theoretical drawbacks of our approach using a single job example where the earliness cost is greater than the tardiness cost. However, our computational results indicate that these pathological cases do not happen in practice when there are many jobs competing for the same time periods. To observe the effect of the cost structure we use the data sets from Mazzini and Armentano (2001), and we also benchmark our results against theirs.

Finally, we compare our lower bound to another lower bound, the linear programming relaxation of P1. We argue in section 6.2 that the bound obtained from the LP relaxation of P1 dominates our lower bound, but the improved bound comes at a significant additional computational cost and with no obvious way of obtaining feasible solutions to the original problem.

6.1 Timetabling

In general, the optimal schedules of E/T problems involve unforced idle time due to the *non-regularity* of their objective functions (see section 1), which implies that given a job sequence it is not straightforward to determine the optimal job start times. Traditionally, optimizing over E/T objectives requires a two-phase procedure. First, a job processing sequence is determined, and then a *timetabling* procedure is applied to the given sequence to compute the optimal job start times.

As discussed in sections 3 and 5, any feasible schedule for P1 is a sequence of *blocks*, separated by idle time. It is a well known fact that the cost of a block is a piecewise linear convex function of the start time

of the block (see Property 3.1 and Yano and Kim (1991)), and optimal timetabling algorithms of low order polynomial complexity have been developed for several special cases of P1 using this property (Garey et al. (1988), Davis and Kanet (1993), Szwarc and Mukhopadhyay (1995)). Also see Kanet and Sridharan (2000) for an overview of different timetabling algorithms.

For our problem, a given job sequence can also be timetabled via a linear program. In the LP below, assume that jobs are numbered in sequence order. We note that since the earliness variables, E_j, and the tardiness variables, T_j, are linearly dependent, there exists an extreme point solution to the linear program for which one of them is zero for each j. The following LP finds the optimal start times for a given job sequence:

(TT-P1)

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} [\pi_j * T_j + \epsilon_j * E_j] \\ \text{s.t.} \quad & \\ & s_j \geq r_j && \forall j \\ & s_j + p_j + E_j - T_j = d_j && \forall j \\ & s_j + p_j \leq s_{j+1} && \forall j \neq n \\ & E_j, T_j \geq 0 && \forall j \end{aligned}$$

Although there exists an $O(n^2)$ algorithm for timetabling P1 (see Nandkeolyar et al. (1993)), we use the LP above to determine the final schedule in each of our heuristics in the actual implementation. As demonstrated by our computational results, the timetabling LP's are small and take very little time to solve even if several of them are solved for a single problem.

6.2 Another Lower Bound

For those instances that we could not solve to optimality, in addition to our lower bound $C(S^*_{TR})$, we employed another lower bound, the LP relaxation of the time-indexed formulation of P1, which we call TIP1. It is well known that the LP relaxations of time indexed formulations of single machine scheduling problems, first introduced by Dyer and Wolsey (1990), provide very tight bounds. In this section, we argue that the lower bound obtained from the LP relaxation of TIP1 dominates the lower bound from TR. However, the tightness of the LP relaxation comes at a significant additional computational cost as we demonstrate in the computational experiments. In addition, it is less obvious how feasible solutions to P1 can be constructed from fractional solutions to this time-indexed formulation.

From sections 3 and 4, recall that $t_{min} = \min_j \ r_j + 1$, $t^j_{min} = r_j + p_j$, $t_{max} = \max_j \ d_j + P$, and the planning horizon is defined as $H = \{k | k \in \mathbb{Z}, \ t \in [t_{min}, t_{max}]\}$. Also, define z_{jk} as a binary variable that takes the value 1 if job j finishes processing at time k, and zero otherwise. Then the time-indexed formulation

of P1 is:

$$\min \quad \sum_{j} \sum_{\substack{k \in H \\ k \geq t^j_{min}}} z_{jk}(\epsilon_j \max(d_j - k)^+ + \pi_j (k - d_j)^+) \tag{6.1}$$

s.t.

(TIP1)

$$\sum_{j} \sum_{t=\max(k,\ t^j_{min})}^{\min(t_{max},\ k+p_j-1)} z_{jt} \leq 1 \qquad \forall k \in H \tag{6.2}$$

$$\sum_{\substack{k \in H \\ k \geq t^j_{min}}} z_{jk} = 1 \qquad \forall j \tag{6.3}$$

$$z_{jk} \in \{0, 1\} \qquad \forall j,\ k \in H,\ k \geq t^j_{min} \tag{6.4}$$

The constraints (6.2) and (6.4) together prescribe that only one job is being processed by the machine at any point in time and that processing is non-preemptive. Constraints (6.3) ensure that all jobs are processed.

Now, consider the linear programming relaxation of TIP1, LP(TIP1). A fractional value α for z_{jk} implies that job j will be processed on the machine in *each* of the p_j unit-length time periods $[k-1,\ k],\ \ldots,\ [k+p_j-2,\ k+p_j-1]$, however only for a duration of α in any such period. This implies that LP(TIP1) is a more constrained version of TR1 that imposes contiguity constraints on X_{jk}'s that belong to the same job. By this observation and Theorem 3.2 one can show that for each feasible solution of LP(TIP1) there exists a feasible solution to TR1 with no greater objective function value. Thus, the bound obtained from LP(TIP1) is tighter than the one obtained from TR1. A formal proof of Theorem 6.1 below is available from the authors upon request.

Theorem 6.1 $C(S^*_{TR2}) = C(S^*_{TR1}) \leq C(S^*_{LP(TIP1)}) \leq C(S^*_{P1})$.

6.3 Data Generation

To generate the processing times and the due dates of jobs for our computational study, we follow the popular scheme of Potts and Van Wassenhove (1982). This method is motivated by the observation that the level of difficulty of problems with due dates depends on the mean and range of the due date distribution.

Processing times are generated from a discrete uniform distribution $U[p_{min},\ p_{max}]$. The due dates are then generated from a discrete uniform distribution $U[\lceil(1 - TF - \frac{RDD}{2}) * P\rceil, \lceil(1 - TF + \frac{RDD}{2}) * P\rceil]$, where P is the sum of processing times. The tardiness factor, TF, is a coarse measure of the proportion of jobs that might be expected to be tardy in an arbitrary sequence (Srinivasan (1971)), and the due date range factor, RDD, specifies the width of the interval centered around the average due date from which the due dates are generated. If $1 - TF - \frac{RDD}{2} < 0$, then the lower bound of the due date range is set to zero.

The ready times are generated from a discrete uniform distribution $U[0,\ P]$. The earliness and tardiness costs are generated from uniform distributions $U[\epsilon_{min}, \epsilon_{max}]$ and $U[\pi_{min}, \pi_{max}]$, respectively.

6.4 Summary of Results

The solution procedure for TR2 and the heuristics were implemented in C using `CPLEX 7.0` callable libraries. When possible, TIP1 and/or its linear programming relaxation were solved via `ILOG AMPL 7.0`. All computations were performed on a Sun UltraSPARC-IIi 300MHz computer with 128 MB of memory.

In the following sections, we discuss the designs and results of the various experiments we conducted to explore the solution quality and/or time for the transportation problem and the heuristics. We examine the effects of the number of jobs, processing times and the relationship between the earliness and tardiness costs. We also benchmark our algorithms against results published in the literature.

6.4.1 Number of Jobs

In order to test whether increasing the number of jobs affects the performance of the lower bound and the heuristics, we design experiments with the parameters given in Table 1. For each combination of parameters, 5 instances are generated, resulting in a total of 100 problems for each value of n. When possible, the problems are solved optimally using the time-indexed formulation TIP1. The maximum CPU time is restricted to half an hour.

n	p_j	r_j	TF	RDD	ϵ_j	π_j
{20, 40, 60, 80}	U[1, 10]	U[0, P]	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	U[0, 100]	U[0, 100]

Table 1: Effect of the Number of Jobs: Problem Parameters.

In Table 2, we report the performance of our lower bound and heuristics as the number of jobs increases, both in terms of solution quality and solution time. BH ("Best Heuristic") indicates the best of the solutions obtained from the four different heuristic sequences, and the columns TL ("Time Limit") and # Opt. ("Number Optimal") indicate the number of times the B&B was terminated due to the time limit and the number of times the best integer solution from the B&B was matched by the heuristics and/or the LP relaxation of TIP1, respectively. The ratio LP/BH compares the solution time of the LP relaxation of TIP1 to the total time to solve the transportation problem and timetable all four heuristic sequences. For each value of n, the first row indicates the average and the second row indicates the worst case performance. The Percentage Gap for any heuristic H refers to

$$\frac{C(H) - C(\text{opt sol'n})}{C(\text{opt sol'n})}$$

when we were able to find the optimal solution. For those instances where we were not able to find the optimal solution, we used $C(S^*_{LP(TIP1)})$ in place of the optimal objective value. The Percentage Gap columns LP and TR indicate how tight the two lower bounds are. Note that these numbers are calculated using only the instances we were able to solve to optimality.

	CPU Time(sec.)					# Opt.		Percentage Gap(%)						
n	**B&B**	**TL**	**LP**	**BH**	**LP/BH**	**BH**	**LP**	**LP**	**TR**	**LCT**	**ACT**	**MCT**	**SW**	**BH**
20	2.05	0	0.28	0.03	8.79	6	10	-0.86	-5.08	12.27	8.64	5.22	4.53	2.45
	8.60		0.72	0.05	14.4			-4.74	-14.75	65.36	37.89	25.77	18.13	12.76
40	62.61	0	3.3	0.13	26.22	0	1	-0.69	-2.86	8.87	7.64	4.19	3.47	2.36
	410		12.82	0.18	71.22			-2.15	-7.90	32.11	28.35	18.59	15.25	10.15
60	626.08	11	14.39	0.38	37.81	0	0	-0.55*	-2.05*	7.89	7.31	3.85	2.99	2.35
	1800		42.24	0.57	74.11			-1.65*	-6.72*	37.63	34.16	20.40	17.30	9.39
80	1325.97	54	38.72	0.83	46.81	1	0	-0.33*	-1.51*	7.97	7.11	3.81	2.82	2.39
	1800		94.4	1.35	69.93			-0.80*	-4.43*	48.16	34.82	17.62	8.68	8.68

(*) Includes only instances solved to optimality.

Table 2: Effect of the Number of Jobs: CPU Times, Average and Worst Case Gaps of Heuristics and Bounds.

The results indicate that the quality of both our lower bound and the heuristics improves as the number of jobs increases. One intuitive explanation is that as the number of jobs increases, more jobs clash, and there are fewer jobs that finish soon after their due dates. Recall that the time interval $[d_j + 1,\ d_j + p_j - 1]$ is the only interval in which the contribution of job j to the objective function of TR, when scheduled non-preemptively, is strictly less than that in the original problem. As the gap between the lower bound and optimal solution diminishes, the sequences extracted from the lower bound improve.

The switch heuristic sequence is the best, in general, followed by the median completion time, average completion time and the last completion time sequences. Overall in these 400 problems, BH has very good average and worst case performances of 2.39% and 12.76%, respectively.

Observe that the solution time of the LP relaxation of TIP1 grows very rapidly, and that TR performs almost as well, is solved much more quickly and is less sensitive to increases in the number of jobs.

6.4.2 Processing Times

We also investigate whether our lower bound and the heuristics are sensitive to increases in processing times, which may imply a potential increase in the number of preemptions. The parameters used are given in Table 3. For each combination of parameters, 5 problem instances are generated, resulting in a total of 300 problems for $n = 20,\ 40,\ 60$ and 200 problems for $n = 80$. The problems with $p_j \sim U[1,\ 10]$ are the same problems as in the previous section.

In this section, we evaluate our algorithms only against the LP relaxation of TIP1 in order to reduce the total computation time required. However, for $n = 60$ and $n = 80$, it is not even possible to set up the LP relaxations within the memory available, so we choose smaller processing times for larger n.

n	p_j	r_j	TF	RDD	ϵ_j	π_j
{20, 40}	U[1, 10], U[1, 30], U[1, 50]	U[0, P]	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	U[0, 100]	U[0, 100]
60	U[1, 10],U[1, 20],U[1, 30]	U[0, P]	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	U[0, 100]	U[0, 100]
80	U[1, 10],U[1, 20]	U[0, P]	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	U[0, 100]	U[0, 100]

Table 3: Effect of Processing Times: Problem Parameters.

		CPU Time(sec.)			**Percentage Gap(%)**					
n	p_j	**LP**	**BH**	**LP/BH**	**TR**	**LCT**	**ACT**	**MCT**	**SW**	**BH**
20	U[1,10]	0.28	0.03	8.79	-4.27	13.25	9.61	6.14	5.44	3.35
		0.72	0.05	14.40	-13.95	65.85	40.43	25.77	18.59	12.94
	U[1,30]	2.77	0.11	25.26	-4.52	16.48	10.99	5.73	7.02	3.99
		8.47	0.22	38.50	-15.39	153.31	50.04	22.85	37.65	15.87
	U[1,50]	8.33	0.25	33.48	-4.79	16.36	12.22	6.15	8.15	4.24
		30.49	0.59	51.68	-20.90	56.42	73.81	35.52	47.78	35.52
40	U[1,10]	3.30	0.13	26.22	-2.19	9.65	8.40	4.93	4.20	3.08
		12.82	0.18	71.22	-6.70	33.21	29.43	20.65	15.97	12.06
	U[1,30]	31.67	0.83	38.30	-2.37	11.11	9.87	4.84	4.97	3.47
		61.00	1.44	42.36	-6.55	49.57	61.00	14.72	15.64	14.72
	U[1,50]	79.81	2.59	30.86	-2.40	12.10	9.21	4.96	4.87	3.46
		140.00	4.93	28.40	-6.54	80.88	38.84	26.20	22.84	15.52
60	U[1,10]	14.39	0.38	37.81	-1.52	8.43	7.85	4.36	3.51	2.85
		42.24	0.57	74.11	-5.15	39.07	35.17	20.40	17.95	10.21
	U[1,20]	49.78	1.28	38.99	-1.62	9.25	7.07	4.72	3.86	3.01
		77.00	1.92	40.10	-4.88	48.37	21.27	19.26	15.97	8.22
	U[1,30]	97.38	3.07	31.73	-1.71	9.43	8.62	5.00	3.77	3.32
		190.00	4.71	40.34	-5.73	31.22	51.34	17.95	11.23	10.38
80	U[1,10]	38.72	0.83	46.81	-1.20	8.14	7.27	3.96	2.98	2.55
		94.40	1.35	69.93	-4.44	49.36	34.82	17.62	9.05	8.68
	U[1,20]	105.70	3.42	30.88	-1.22	8.37	7.26	4.02	3.06	2.66
		200.00	5.56	35.97	-3.90	28.12	25.54	17.88	9.49	9.49

Table 4: Effect of Processing Times: CPU Times, Average and Worst Case Gaps of Heuristics and Bounds.

Note that by comparing our heuristics to a lower bound, we give an *upper* bound on the optimality gap. Nevertheless, the results in Table 4 are excellent. For these 1100 problems, the average gap of the best heuristic with respect to the LP relaxation of TIP1 is just 3.79%, and except for $n = 20$ and $p_j \sim U[1,\ 50]$, the worst case is within 16% of the LP lower bound. Note that the column TR indicates the gap between the optimal solutions of TR and the LP relaxation.

There is a very slight degradation in the performance measures as the processing times increase, although this effect is less pronounced than the effect of an increase in the number of jobs. The combined effect of increasing the number of jobs and the processing times is a decrease in the percentage gap. In general, the quality of our lower bound does not seem to be jeopardized by the larger number of preemptions that occurs when processing times are longer.

In section 3, we argued that generating TR from P1 is carried out in pseudo-polynomial time because the number of nodes and arcs in the transportation network depend on the sum of the processing times. In Table 5, we divide the average solution times given in Table 4 into their components. The columns S, TR, TT and TO indicate the average CPU times, spent for setting up the transportation problem, solving the transportation problem, timetabling all four sequences, and the whole algorithm, respectively, for 100 instances.

	n=20				**n=40**				**n=60**				**n=80**			
p_j	**S**	**TR**	**TT**	**TO**	**S**	**TR**	**TT**	**TO**	**S**	**TR**	**TT**	**TO**	**S**	**TR**	**TT**	**TO**
U[1,10]	0.00	0.01	0.01	0.03	0.01	0.08	0.03	0.13	0.03	0.30	0.05	0.38	0.05	0.70	0.07	0.83
U[1,20]									0.05	1.17	0.06	1.28	0.08	3.25	0.09	3.42
U[1,30]	0.01	0.09	0.02	0.11	0.03	0.75	0.04	0.83	0.07	2.93	0.07	3.07				
U[1,50]	0.01	0.22	0.02	0.25	0.05	2.49	0.05	2.59								

Table 5: Detailed CPU time (sec.) analysis of the heuristics.

It is clear that compared to the solution time of the transportation problem, setting up the problem and timetabling all four sequences takes an insignificant amount of time. Generating additional sequences from the optimal transportation solution improves solution quality enormously, but does not slow down the overall procedure significantly.

6.4.3 Cost Structure

In this section, we explore the sensitivity of our algorithms to the relationship between the earliness and the tardiness costs. We use the data sets from Mazzini and Armentano (2001), who present both optimal and heuristic solutions for these problems. The values of the parameters are given in Table 6. For each combination of parameters, there are 5 problem instances, resulting in a total of 180 problems for each n.

n	p_j	r_j	TF	RDD	ϵ_j	π_j
{8, 10, 12, 20, 40, 60, 80}	U[1, 100]	U[0, P]	{0.2, 0.5, 0.8}	{0.4, 0.7, 1.0}	U[0, 50]	$\pi_j = \epsilon_j * 2$
{8, 10, 12, 20, 40, 60, 80}	U[1, 100]	U[0, P]	{0.2, 0.5, 0.8}	{0.4, 0.7, 1.0}	U[0, 100]	$\pi_j = \epsilon_j,\ \pi_j = \frac{\epsilon_j}{2},\ U[0,\ 100]$

Table 6: Parameters for the data sets from Mazzini and Armentano (2001).

For $n = 8,\ 10,\ 12$, we compute the performance measures with respect to the optimal solution when it is available or with respect to $C(S^*_{LP(TIP1)})$ when we were not able to find the optimal solution within the time limit of half an hour (only for 4 instances with n=12). We did not use the solutions of Mazzini and Armentano (2001), as in several cases we discovered that they erroneously report optimal solutions. Hence, we also re-evaluated the performance of the heuristics proposed by Mazzini and Armentano (2001) based on the new best solutions obtained.

	Costs		**Percentage Gap(%)**							**CPU Time**
n	ϵ_j	π_j	**LB**	**LCT**	**ACT**	**MCT**	**SW**	**BH**	**# Opt.**	**(sec.)**
8	U[0, 50]	ϵ_j*2	-2.44	7.22	1.90	0.95	3.05	0.15	33	0.09
			-7.08	40.22	24.69	7.94	43.48	3.87		0.23
	U[0, 100]	ϵ_j*1	-3.82	7.21	3.30	0.89	3.62	0.15	37	0.11
			-20.01	35.51	21.87	10.41	27.65	3.04		0.39
	U[0, 100]	ϵ_j*0.5	-5.01	9.52	3.44	1.83	2.62	0.39	34	0.09
			-48.51	136.36	44.17	25.86	24.60	5.23		0.21
	U[0, 100]	100	-4.20	9.53	3.32	2.22	4.58	0.71	31	0.09
			-30.25	50.27	29.59	18.96	31.65	12.52		0.21
10	U[0, 50]	ϵ_j*2	-7.40	16.40	10.30	5.88	14.65	1.90	17	0.17
			-21.85	77.14	80.87	80.87	115.13	15.09		0.45
	U[0, 100]	ϵ_j*1	-8.27	18.39	8.81	4.56	9.62	2.00	20	0.14
			-32.01	87.02	31.42	20.40	58.90	13.92		0.41
	U[0, 100]	ϵ_j*0.5	-8.31	13.57	8.26	6.34	8.95	1.99	19	0.14
			-27.96	50.56	75.32	75.32	61.48	9.45		0.26
	U[0, 100]	100	-8.33	20.10	15.43	9.95	8.66	2.97	18	0.15
			-31.28	95.59	187.62	149.17	54.46	29.50		0.37
12	U[0, 50]	ϵ_j*2	-7.89*	19.03	11.66	5.75	7.06	1.88	12	0.21
			-31.16*	98.81	80.69	39.95	22.67	13.47		0.37
	U[0, 100]	ϵ_j*1	-7.74*	16.78	11.47	7.33	11.33	2.86	13	0.25
			-33.51*	68.29	76.26	62.94	98.53	14.74		0.43
	U[0, 100]	ϵ_j*0.5	-7.36	19.61	8.29	4.04	6.34	2.06	14	0.22
			-35.67	105.39	43.12	25.06	28.52	15.64		0.37
	U[0, 100]	100	-6.83*	16.76	9.39	7.20	7.73	3.00	8	0.24
			-20.09*	91.09	41.44	88.99	44.57	19.02		0.43

(*) Includes only instances solved to optimality.

Table 7: Effect of the Cost Structure: CPU Times, Average and Worst Case Gaps of Heuristics and Bounds.

For $n = 20$, we could not solve the LP relaxation of TIP1 for 14 problems, and for n greater than 20, none of the LP relaxations was solvable due to memory limitations. Therefore, for the larger problems ($n \geq 20$),

	Costs		Percentage Gap(%)					CPU Time
n	ϵ_j	π_j	**LCT**	**ACT**	**MCT**	**SW**	**BH**	**(sec.)**
20	U[0, 50]	ϵ_j*2	6.91	5.57	3.84	4.22	2.57	1.04
			28.60	38.25	12.16	24.28	9.22	1.81
	U[0, 100]	ϵ_j*1	6.40	4.67	3.95	3.81	2.82	1.16
			17.26	23.02	13.85	11.49	7.70	2.59
	U[0, 100]	ϵ_j*0.5	9.06	6.38	4.39	5.17	3.85	1.07
			34.73	27.11	21.93	25.23	21.93	2.02
	U[0, 100]	100	8.69	6.59	4.49	5.35	3.54	1.14
			39.94	26.00	17.79	17.32	11.70	2.15
40	U[0, 50]	ϵ_j*2	6.35	4.66	2.71	2.85	2.03	7.87
			23.89	43.07	9.36	11.40	9.36	12.32
	U[0, 100]	ϵ_j*1	4.63	3.46	2.10	2.29	1.71	7.63
			15.17	11.81	7.53	7.78	3.76	11.13
	U[0, 100]	ϵ_j*0.5	6.18	4.81	2.92	3.13	2.49	8.06
			27.87	20.58	14.18	10.53	10.53	17.74
	U[0, 100]	100	6.03	4.67	3.12	3.03	2.42	9.49
			19.48	16.35	11.91	13.12	6.55	14.98
60	U[0, 50]	ϵ_j*2	3.98	2.89	1.90	1.94	1.58	21.71
			16.60	9.60	5.81	5.02	4.20	32.50
	U[0, 100]	ϵ_j*1	4.59	2.97	2.08	1.97	1.57	20.15
			16.78	11.39	9.13	6.11	4.46	33.57
	U[0, 100]	ϵ_j*0.5	4.38	3.18	2.01	2.13	1.65	20.48
			13.01	17.11	7.31	10.31	6.09	35.30
	U[0, 100]	100	4.27	3.37	2.21	2.15	1.82	24.82
			15.11	9.07	6.65	8.59	6.65	41.72
80	U[0, 50]	ϵ_j*2	3.21	2.94	1.57	1.48	1.27	48.65
			12.21	13.33	6.97	5.91	4.63	92.12
	U[0, 100]	ϵ_j*1	3.30	3.09	1.95	1.68	1.45	49.79
			13.79	13.00	6.10	6.44	4.74	93.36
	U[0, 100]	ϵ_j*0.5	3.87	3.20	1.91	1.85	1.62	47.83
			14.90	12.45	8.12	5.65	5.42	101.15
	U[0, 100]	100	4.38	3.38	2.28	1.77	1.67	66.51
			11.03	15.06	8.91	4.27	4.27	154.99

Table 8: Effect of the Cost Structure: CPU Times, Average and Worst Case Gaps of Heuristics.

we compute the performance measures for all instances with respect to our lower bound, which yields an upper bound on the performance measures as in the previous section.

In Tables 7 and 8, for each combination of the number of jobs and the cost parameters, the first row indicates the average and the second row indicates the worst case performance. From section 4, recall that one can create single-job instances with $\epsilon > \pi$ for which the gap between the optimal solution to P1 and the lower bound is quite large. However, Tables 7 and 8 indicate that the performance of our algorithms is not sensitive to the cost structure. In particular, for the case when the earliness costs are twice as large as the tardiness costs, they perform as well as in the other cases.

We again note that as the problem size increases the performance measures improve significantly and the solution time increases rapidly. The last completion time sequence is the worst in most instances, and for small n the median completion time sequence is superior to all others. As n increases, the median completion time and switch heuristic sequences dominate the other two.

Overall, our algorithms demonstrate outstanding performance. For $n = 8,\ 10,\ 12$, we were able to find the optimal solution in 135, 74 and 47 out of 180 problems in each case, respectively. For these 540 problems, the average and worst case gap of BH are 1.67% and 29.50%, respectively. For $n = 20,\ 40,\ 60,\ 80$, for which we give an upper bound on the performance measures for a total of 720 problems, the average and worst case gaps are 2.13% and 21.93%, respectively.

	Gap(%)		**# Opt.**		**Number of Times**		
n	**MA**	**BH**	**MA**	**BH**	**BH < MA**	**MA < BH**	**BH=MA**
8	0.89	0.35	127	135	44	28	108
	30.03	12.52					
10	5.47	2.22	64	74	82	60	38
	85.12	29.50					
12	7.11	2.45	41	47	104	58	18
	59.04	19.02					
20	4.88	3.19			80	90	10
	45.47	21.93					
40	4.35	2.16			116	64	0
	38.52	10.53					
60	3.79	1.65			115	65	0
	23.64	6.65					
80	3.50	1.50			125	55	0
	44.30	5.42					

Table 9: Comparison with Mazzini and Armentano (2001).

Finally, Table 9 presents a comparison of the results of Mazzini and Armentano (2001) to our results. For each n, the first row indicates the average and the second row indicates the worst case performance over 180 problems. The label MA denotes the heuristic presented in Mazzini and Armentano (2001). In almost

all cases, our average gap is less than half of the average gap of MA, and the worst cases indicate that our algorithms are much more robust. Additionally, the difference between the two approaches becomes more significant as n increases, and for $n = 80$, we find better results in more than two-thirds of the problems.

7 Concluding Remarks

We developed a methodology to solve a difficult single machine scheduling problem, the non-preemptive earliness/tardiness problem, in its most general form, with only mild assumptions about the structure of the problem parameters. This research was motivated by the appearance of this model as a sub-problem in more complex scheduling environments. We investigated a relatively unexplored path by considering a preemptive structure that differs from the preemption implicit in the LP relaxation of the integer programming representation of the original non-preemptive model. Our computational experiments demonstrate that this approach is both fast and effective.

Our model is flexible and can be used to solve a variety of problems by making appropriate adjustments to the problem parameters. For instance, the celebrated weighted tardiness problem can be solved by setting all earliness costs equal to zero. We also note that our model is able to handle raw material inventory holding costs for jobs that are ready, but waiting for processing. This extension requires an update of costs, $\epsilon_j^{'} = \epsilon_j - h_j$ and $\pi_j^{'} = \pi_j + h_j$ for all jobs, where h_j is the raw material inventory holding cost of job j, and $\epsilon_j^{'}$ and $\pi_j^{'}$ are the new weights. In the future, we hope to use these properties to solve flow and job shop scheduling problems with intermediate inventory holding costs.

We recognize that our model has several limitations. We assume complete knowledge of all parameters of all jobs to arrive in the future, but this is rarely true in real life settings. However, it is customary to schedule the available jobs using available information, so the problem still needs to be solved. Also, we do not take into account that processing times are often stochastic, and the machine might not be reliable. Finally, we note that our approach cannot accommodate precedence constraints among jobs, as these would destroy the structure of the preemptive relaxation. Nevertheless, we are encouraged by the performance of our preemption-based heuristics for this model, and we hope to extend this approach to more complex environments that better reflect real world issues.

Acknowledgment

We would like to thank to Professor Vinicius Armentano and Dr. Renata Mazzini for providing the data in section 6.4.3. This research was partially supported by NSF Grant DMI-0092854.

A Appendix

A.1 Proof of Theorem 3.2

Proof. We show that for each optimal solution S^*_{P1} to P1, there exists a corresponding feasible schedule S_{TR1} for TR1 such that $C(S_{TR1}) \leq C(S^*_{P1})$. In particular, we consider solution S_{TR1} to TR1 constructed by converting S^*_{P1} into a feasible solution to TR1. This is accomplished by dividing each job in S^*_{P1} into contiguous unit-duration segments. We demonstrate that for a schedule S_{TR1} constructed in this manner, $C(S_{TR1}) \leq C(S^*_{P1})$. Clearly, an optimal solution S^*_{P1} to P1 exists in which all job completion times belong to $H = \{k | k \in \mathbb{Z},\ k \in [t_{min}, t_{max}]\}$, the same time horizon considered in problem TR1.

Our strategy is to consider each job in S^*_{P1} separately. Since each job can either be early, tardy, or on time, we consider the following cases:

1. If job j completes **on time** in S^*_{P1}, it incurs a cost of zero. In S_{TR1}, the same job j incurs a cost of:

$$\begin{aligned}
\sum_{k=d_j-p_j+1}^{d_j} c_{jk} &= \frac{\epsilon_j}{p_j} \sum_{k=d_j-p_j+1}^{d_j} \left[(d_j - \frac{p_j}{2}) - (k - \frac{1}{2}) \right] \\
&= \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (d_j - p_j + 1 - \frac{1}{2}) \right] + ... + \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (d_j - p_j + p_j - \frac{1}{2}) \right] \\
&= \frac{\epsilon_j}{p_j} \left[\frac{p_j^2}{2} - \frac{p_j(p_j+1)}{2} + \frac{p_j}{2} \right] \\
&= 0
\end{aligned}$$

2. If job j completes (strictly) **early** at time C_j in S^*_{P1}, it incurs a cost of $\epsilon_j * (d_j - C_j)$. In S_{TR1}, the same job j incurs a cost of:

$$\begin{aligned}
\sum_{k=C_j-p_j+1}^{C_j} c_{jk} &= \frac{\epsilon_j}{p_j} \sum_{k=C_j-p_j+1}^{C_j} \left[(d_j - \frac{p_j}{2}) - (k - \frac{1}{2}) \right] \\
&= \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (C_j - p_j + 1 - \frac{1}{2}) \right] + ... + \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (C_j - p_j + p_j - \frac{1}{2}) \right] \\
&= \frac{\epsilon_j}{p_j} \left[p_j(d_j - C_j) + \frac{p_j^2}{2} - \frac{p_j(p_j+1)}{2} + \frac{p_j}{2} \right] \\
&= \epsilon_j(d_j - C_j)
\end{aligned}$$

3. A **tardy** job j that completes at time C_j in S^*_{P1} incurs a cost of $\pi_j * (C_j - d_j)$. To determine the cost in S_{TR1}, we distinguish between two cases.

If $C_j \geq d_j + p_j$, then in S_{TR1}, all unit jobs of job j incur a tardiness cost of:

$$\begin{aligned}
\sum_{k=C_j-p_j+1}^{C_j} c_{jk} &= \frac{\pi_j}{p_j} \sum_{k=C_j-p_j+1}^{C_j} \left[(k - \frac{1}{2}) - (d_j - \frac{p_j}{2}) \right] \\
&= \frac{\pi_j}{p_j} \left[(C_j - p_j + 1 - \frac{1}{2}) - (d_j - \frac{p_j}{2}) \right] + ... + \frac{\pi_j}{p_j} \left[(C_j - p_j + p_j - \frac{1}{2}) - (d_j - \frac{p_j}{2}) \right] \\
&= \frac{\pi_j}{p_j} \left[p_j(C_j - d_j) - \frac{p_j^2}{2} + \frac{p_j(p_j+1)}{2} - \frac{p_j}{2} \right] \\
&= \pi_j(C_j - d_j)
\end{aligned}$$

If $d_j + 1 \leq C_j \leq d_j + p_j - 1$, then in S_{TR1}, x unit jobs of job j incur a tardiness cost, and the remaining $(p_j - x)$ unit jobs incur an earliness cost. Clearly, this case is only relevant if $p_j \geq 2$. We have $x = C_j - d_j$ where $1 \leq x \leq p_j - 1$, and in S_{TR1} job j incurs a cost of:

$$\sum_{k=C_j-p_j+1}^{C_j} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j+x} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j} c_{jk} + \sum_{k=d_j+1}^{d_j+x} c_{jk} \tag{A.1}$$

We first examine the costs incurred by the unit jobs completed at or before d_j:

$$\begin{aligned}
\sum_{k=d_j+x-p_j+1}^{d_j} c_{jk} &= \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (d_j + x - p_j + 1 - \frac{1}{2}) \right] + ... \\
&+ \frac{\epsilon_j}{p_j} \left[(d_j - \frac{p_j}{2}) - (d_j + x - p_j + p_j - x - \frac{1}{2}) \right] \\
&= \frac{\epsilon_j}{p_j} \left[\frac{p_j}{2}(p_j - x) - x(p_j - x) - \frac{(p_j - x)(p_j - x + 1)}{2} + \frac{p_j - x}{2} \right] \\
&= \frac{\epsilon_j}{p_j} \left[(p_j - x) \left(\frac{p_j}{2} - x - \frac{p_j}{2} + \frac{x}{2} - \frac{1}{2} \right) + \frac{p_j}{2} - \frac{x}{2} \right] \\
&= \frac{\epsilon_j}{p_j} \left[\frac{-xp_j}{2} - \frac{p_j}{2} + \frac{x^2}{2} + \frac{x}{2} + \frac{p_j}{2} - \frac{x}{2} \right] \\
&= \frac{\epsilon_j}{p_j} \left[\frac{-xp_j}{2} + \frac{x^2}{2} \right] \quad \text{(see below)} \\
&\leq 0
\end{aligned}$$

The final step is justified by maximizing the term in square brackets for $1 \leq x \leq p_j - 1$. Note that $f(x) = \frac{-xp_j}{2} + \frac{x^2}{2}$ is a convex function that is strictly decreasing (increasing) for $x \leq \frac{p_j}{2}$ ($x \geq \frac{p_j}{2}$). Hence, on the interval $[1,\ p_j - 1]$, $f(x)$ must attain its maximum value at one of the end points of the interval. Since $p_j \geq 2$ by assumption, $f(1) = \frac{-p_j}{2} + \frac{1}{2} \leq 0$. Also, for $p_j \geq 2$, we have $f(p_j - 1) < f(p_j) = \frac{-p_j^2}{2} + \frac{p_j^2}{2} = 0$.

Next, we examine the costs incurred by the unit jobs completed after d_j.

$$\begin{aligned}
\sum_{k=d_j+1}^{d_j+x} c_{jk} &= \frac{\pi_j}{p_j}\left[(d_j+1-\frac{1}{2})-(d_j-\frac{p_j}{2})\right]+...+\left[(d_j+x-\frac{1}{2})-(d_j-\frac{p_j}{2})\right] \\
&= \frac{\pi_j}{p_j}\left[\frac{x(x+1)}{2}-\frac{x}{2}+\frac{xp_j}{2}\right] \\
&= \frac{\pi_j}{p_j}\left[\frac{x^2}{2}+\frac{xp_j}{2}\right] \\
&= \pi_j x\left[\frac{x}{2p_j}+\frac{1}{2}\right] \\
&< \pi_j x \qquad \text{(because } x < p_j\text{)}
\end{aligned}$$

Therefore,

$$\sum_{k=C_j-p_j+1}^{C_j} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j+x} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j} c_{jk} + \sum_{k=d_j+1}^{d_j+x} c_{jk} < \pi_j x = \pi_j(C_j-d_j)$$

by the definition of x.

Thus, summing over all jobs,

$$C(S^*_{TR1}) \le C(S_{TR1}) \le C(S^*_{P1}). \quad \blacksquare$$

A.2 Proof of Lemma 3.3

Proof. In the solution to TR1, if X_{jk} is equal to 1, then job j is processed in period k. Now, assume that S^*_{TR1} is an optimal solution to TR1, where $X_{jm} = 1$ and $m \notin H_j$, i.e., there is a time period k assigned to job j such that k is more than P time units away from d_j if permitted by the ready time r_j.

1. If $m \le d_j$, then $r_j + 1 \le m \le d_j - P$ must hold because no unit job of job j can start before r_j. Note that $H_j = \{k | k \in \mathbb{Z},\ k \in [d_j - P + 1,\ d_j + P]\}$ in this case, and since there are P total units of processing, there exists an idle period k such that $d_j - P + 1 \le k \le d_j$. But $c_{jm} - c_{jk} = \frac{\epsilon_j}{p_j}\left[(d_j - \frac{p_j}{2}) - (m - \frac{1}{2}) - (d_j - \frac{p_j}{2}) + (k - \frac{1}{2})\right] = \frac{\epsilon_j}{p_j}(k - m) > 0$, i.e., we can decrease the objective function value by setting $X_{jm} = 0$ and $X_{jk} = 1$ which contradicts the optimality of S^*_{TR1}.

2. If $m > d_j$, then $m > d_j + P$ must be true by assumption. However, by the same reasoning as above, there exists an idle period k such that $d_j + 1 \le k \le d_j + P$.
   Then, $c_{jm} - c_{jk} = \frac{\pi_j}{p_j}\left[(m - \frac{1}{2}) - (d_j - \frac{p_j}{2}) - (k - \frac{1}{2}) + (d_j - \frac{p_j}{2})\right] = \frac{\pi_j}{p_j}(m - k) > 0$, i.e., we can decrease the objective function value by setting $X_{jm} = 0$ and $X_{jk} = 1$ which contradicts the optimality of S^*_{TR1}.
   $\blacksquare$

A.3 Proof of Lemma 4.1

Proof. Recall that the cost parameters in TR are:

$$c_{jk} = \begin{cases} \frac{\epsilon_j}{p_j}\left[(d_j - \frac{p_j}{2}) - (k - \frac{1}{2})\right] & \text{if } k \le d_j \\ \frac{\pi_j}{p_j}\left[(k - \frac{1}{2}) - (d_j - \frac{p_j}{2})\right] & \text{if } k > d_j \end{cases}$$

Clearly, in an optimal transportation solution S^*_{TR} there will be no idle time between the unit jobs of job j. Also, because $c_{jk} \le 0$ in the interval $d_j - \frac{p_j}{2} + \frac{1}{2} \le k \le d_j$, in the optimal solution S^*_{TR}, unit jobs will be scheduled throughout the interval $[d_j - \frac{p_j}{2} + \frac{1}{2},\ d_j]$, and the remaining unit jobs will be distributed around this interval so as to minimize the cost.

Hence, the cost to be minimized can be expressed as a function of the tardiness of the job, $x = \max(0,\ C_j - d_j)$ (see Theorem 3.2):

$$f(x) = \sum_{k=C_j-p_j+1}^{C_j} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j+x} c_{jk} = \frac{1}{2}\left(\frac{\epsilon_j}{p_j} + \frac{\pi_j}{p_j}\right)x^2 + \frac{1}{2}\left(\pi_j - \epsilon_j\right)x \tag{A.2}$$

where $x \in \mathbb{Z}$ and $0 \le x \le p_j$.

Observe that $f(x)$ is convex, and for $x \in \mathbb{R}$ its unconstrained minimizer $x^*_{\mathbb{R}}$ can be obtained from the first order condition $f'(x) = 0$. Then

$$x^*_{\mathbb{R}} = \frac{\epsilon_j - \pi_j}{2 * (\epsilon_j + \pi_j)} * p_j = \frac{\beta * \pi_j - \pi_j}{2 * (\beta * \pi_j + \pi_j)} * p_j.$$

Now, we consider two cases:

i. If $\beta \le 1$, then $x^*_{\mathbb{R}} \le 0$, and by the convexity of f, $x^* = 0$ minimizes (A.2). This means that the optimal transportation schedule S^*_{TR}, described by $X_{j,d_j-p_j+1} = \ldots = X_{j,d_j} = 1$, is the same as the optimal schedule S^*_{P1} of the original problem, and $C(S^*_{TR}) = C(S^*_{P1}) = 0$.

ii. If $\beta > 1$, then $x^*_{\mathbb{R}} > 0$, and by convexity of f,

$$x^* = \begin{cases} \lfloor \frac{\beta*\pi_j - \pi_j}{2*(\beta*\pi_j + \pi_j)} * p_j \rfloor & \text{if } f(\lfloor \frac{\beta*\pi_j - \pi_j}{2*(\beta*\pi_j + \pi_j)} * p_j \rfloor) \le f(\lceil \frac{\beta*\pi_j - \pi_j}{2*(\beta*\pi_j + \pi_j)} * p_j \rceil) \\ \lceil \frac{\beta*\pi_j - \pi_j}{2*(\beta*\pi_j + \pi_j)} * p_j \rceil & \text{otherwise} \end{cases} \tag{A.3}$$

Thus, if $\frac{\beta*\pi_j - \pi_j}{2*(\beta*\pi_j + \pi_j)} * p_j \ge 1$, then $x^* \ge 1$, so S^*_{TR} is different than S^*_{P1} and $C(S^*_{TR}) < 0$. Note that this is a sufficient condition. ∎

A.4 Proof of Corollary 4.2

Proof. We noted before that $f(x)$, defined above, is convex and its unconstrained minimizer for $x \in \mathbb{R}$ is $x^*_{\mathbb{R}} = \frac{\epsilon_j - \pi_j}{2*(\epsilon_j + \pi_j)} * p_j = \frac{\beta - 1}{2(\beta+1)} p_j$ with $\epsilon_j = \beta * \pi_j$. Therefore, for $x \in \mathbb{R}$ we have $\lim_{\beta \to \infty} x^*_{\mathbb{R}} = \frac{p_j}{2}$. This implies that there are two cases for our problem:

i. If p_j is even then $\lim_{\beta\to\infty} x^* = \frac{p_j}{2}$ and we can find the limiting value of $C(S^*_{TR})$ by computing $f(\frac{p_j}{2})$ which yields $\lim_{\beta\to\infty} C(S^*_{TR}) = \frac{\pi_j p_j}{8} * (3-\beta)$.

ii. If p_j is odd then $\lim_{\beta\to\infty} x^* = \frac{p_j}{2} - \frac{1}{2}$ because $f(\frac{p_j}{2}+\frac{1}{2}) - f(\frac{p_j}{2}-\frac{1}{2}) = \pi_j > 0$. Then, $\lim_{\beta\to\infty} C(S^*_{TR}) = f(\frac{p_j}{2}-\frac{1}{2}) = \frac{\pi_j}{8p_j}\beta(1-p_j^2) + \frac{\pi_j}{8p_j} * (1+3p_j^2-4p_j)$.

∎

A.5 Proof of Lemma 4.4

Proof. We have observed that (see Theorem 3.2), $C_j(S_{P1}) = C_j(S_{TR})$ if $C_j \le d_j$ or $C_j \ge d_j + p_j$. If $p_j = 1$ then these are the only possibilities, and in the remaining case with $p_j \ge 2$, the only pertinent values of C_j are $d_j < C_j < d_j + p_j$. Let x be the tardiness of the job. Then,

$$\begin{aligned} C_j(S_{P1}) - C_j(S_{TR}) &= g(x) = \pi_j x - \frac{1}{2}\left(\frac{\pi_j+\epsilon_j}{p_j}\right)x^2 - \frac{1}{2}(\pi_j-\epsilon_j)\,x \\ &= \frac{1}{2}(\pi_j+\epsilon_j)\,x - \frac{1}{2}\left(\frac{\pi_j+\epsilon_j}{p_j}\right)x^2 \quad \text{where } x\in\mathbb{Z} \text{ and } 1\le x\le p_j-1 \end{aligned}$$

The function $g(x)$ is concave since $\frac{\partial^2 g}{\partial x^2} = -\frac{\pi_j+\epsilon_j}{p_j} < 0$ and its maximizer for $x\in\mathbb{R}$ is given by

$$x^*_{\mathbb{R}} = \frac{p_j}{2}$$

This implies two cases when x is integer.

i. If p_j is even then $x^* = x^*_{\mathbb{R}} = \frac{p_j}{2}$ and $g(x^*) = \frac{1}{8}(\pi_j+\epsilon_j)\,p_j$.

ii. If p_j is odd then $x^* = \frac{p_j}{2} - \frac{1}{2} = \frac{p_j}{2} + \frac{1}{2}$ and $g(x^*) = \frac{1}{8}(\pi_j+\epsilon_j)\,p_j - \frac{1}{8p_j}(\pi_j+\epsilon_j) \le \frac{1}{8}(\pi_j+\epsilon_j)\,p_j$.

Thus,

$$\begin{cases} C_j(S_{P1}) - C_j(S_{TR}) \le \frac{1}{8}(\pi_j+\epsilon_j)\,p_j, & d_j < C_j < d_j + p_j \\ C_j(S_{P1}) - C_j(S_{TR}) = 0, & \text{otherwise} \end{cases}$$

Finally,

$$\begin{aligned} C(S_{P1}) - C(S_{TR}) &= \sum_j [C_j(S_{P1}) - C_j(S_{TR})] \\ &\le 0 + \sum_{j:d_j<C_j<d_j+p_j} [C_j(S_{P1}) - C_j(S_{TR})] + 0 \\ &\le \frac{1}{8}\sum_{j=1}^{n}(\pi_j+\epsilon_j)\,p_j. \end{aligned}$$

. ∎

A.6 Proof of Lemma 4.5

Proof. $0 \leq C(S_{TR}) \leq C(S_{P1})$ by Lemma 4.1, Assumption 4.3 and Theorem 3.2. Let $C_j(S_{P1})$ and $C_j(S_{TR})$ be the total cost associated with job j in schedules S_{P1} and S_{TR}, respectively. As in Lemma 4.4, $C_j(S_{P1}) = C_j(S_{TR})$ if $C_j \leq d_j$ or $C_j \geq d_j + p_j$. If $p_j = 1$ then these are the only possibilities.

We now consider the remaining case with $p_j \geq 2$ and $d_j < C_j < d_j + p_j$. Let $x = C_j - d_j$, so $1 \leq x \leq p_j - 1$. Then, the costs incurred by job j in S_{P1} and S_{TR} are

$$C_j(S_{P1}) = \pi_j * x$$

and

$$C_j(S_{TR}) = f(x) = \frac{1}{2}\left(\frac{\epsilon_j}{p_j} + \frac{\pi_j}{p_j}\right)x^2 + \frac{1}{2}(\pi_j - \epsilon_j)\,x \qquad \text{(see A.2)},$$

respectively.

For this case we will show that $C_j(S_{P1}) \leq p_j C_j(S_{TR})$ by defining the ratio of individual job costs,

$$R_j(x,\ \alpha) = \frac{C_j(S_{TR})}{C_j(S_{P1})} = \frac{\frac{1}{2}\left(\frac{\epsilon_j}{p_j} + \frac{\pi_j}{p_j}\right)x^2 + \frac{1}{2}(\pi_j - \epsilon_j)\,x}{\pi_j x} = \frac{(\alpha+1)}{2p_j\alpha}x + \frac{(\alpha-1)}{2\alpha} > 0$$

where $\pi_j = \alpha * \epsilon_j$, and $\alpha \geq 1$ by Assumption 4.3. The ratio $\frac{C_j(S_{TR})}{C_j(S_{P1})}$ is easier to analyze than its inverse, which is the expression we seek to maximize.

Thus, we need to find

$$\min_{\alpha \geq 1,\ 1 \leq x \leq p_j - 1} R_j(x,\ \alpha).$$

Observe that

$$\min_{\alpha \geq 1,\ 1 \leq x \leq p_j - 1} R_j(x,\ \alpha) = R_j(1,\ \alpha)$$

because the coefficient of x, i.e., $\frac{(\alpha+1)}{2p_j\alpha}$, is strictly positive and therefore $R_j(x,\ \alpha)$ is minimized for all fixed α by setting $x = 1$.

Now, $\frac{\partial R_j(1,\ \alpha)}{\partial \alpha} = \frac{p_j - 1}{2p_j\alpha^2} > 0$ for all values of α which implies that $R_j(1,\ \alpha)$ is a strictly increasing function for all α. Therefore, we have

$$\min_{\alpha \geq 1,\ 1 \leq x \leq p_j - 1} R_j(x,\ \alpha) = R_j(1,\ 1) = \frac{1}{p_j}.$$

Thus,

$$\begin{cases} C_j(S_{P1}) \leq p_j C_j(S_{TR}) & \text{if } d_j < C_j < d_j + p_j \\ C_j(S_{P1}) = C_j(S_{TR}) & \text{otherwise} \end{cases}$$

Finally,

$$\begin{aligned}
C(S_{P1}) = \sum_{j=1}^{n} C_j(S_{P1}) &= \sum_{j:C_j \le d_j} C_j(S_{P1}) + \sum_{j:d_j < C_j < d_j + p_j} C_j(S_{P1}) + \sum_{j:C_j \ge d_j + p_j} C_j(S_{P1}) \\
&= \sum_{j:C_j \le d_j} C_j(S_{TR}) + \sum_{j:d_j < C_j < d_j + p_j} C_j(S_{P1}) + \sum_{j:C_j \ge d_j + p_j} C_j(S_{TR}) \\
&\le \sum_{j:C_j \le d_j} C_j(S_{TR}) + \sum_{j:d_j < C_j < d_j + p_j} p_j C_j(S_{TR}) + \sum_{j:C_j \ge d_j + p_j} C_j(S_{TR}) \\
&\le \sum_{j:C_j \le d_j} p_{max} C_j(S_{TR}) + \sum_{j:d_j < C_j < d_j + p_j} p_{max} C_j(S_{TR}) + \sum_{j:C_j \ge d_j + p_j} p_{max} C_j(S_{TR}) \\
&= p_{max} C(S_{TR})
\end{aligned}$$

∎

Note that the fact that $R_j(1,\ \alpha)$ is a strictly increasing function suggests that the lower bound becomes tighter when the unit tardiness cost increases relative to the unit earliness cost.

A.7 Proof of Lemma 4.7

Proof. To demonstrate that this dual solution is feasible, we consider each of the constraints and make the appropriate substitutions.

(4.11): $\sum_{t=t_{min}}^{d} p\gamma_t + \sum_{t=1}^{p-1}(p-t)\delta_{d+t} = p(d - t_{min} + 1 - 2 + \gamma_{t_{min}} + \gamma_d) + \frac{p(p-1)}{2} = 0$ by (4.15)-(4.18).

(4.12): $\sum_{t=1}^{p-1} t\delta_{d+t} + \sum_{t=d+p}^{t_{max}} p\mu_t = \frac{p(p-1)}{2} + p(t_{max} - d - p + 1 - 2 + \mu_{d+p} + \mu_{t_{max}}) = 0$ by (4.18)-(4.21).

(4.13): $\sum_{t=t_{min}}^{d} \sum_{k=t-p+1}^{t} (d-k)\gamma_t + \sum_{t=d+1}^{d+p-1} \sum_{k=t-p+1}^{d} (d-k)\delta_t \overset{?}{\le} 0$

We first analyze the first term:

$$\begin{aligned}
\sum_{t=t_{min}}^{d} \sum_{k=t-p+1}^{t} (d-k)\gamma_t &= \sum_{y=1}^{d-t_{min}+1} \left[(y-1)p + \frac{p(p-1)}{2}\right] \gamma_{d+1-y} \\
&= \frac{p(p-1)}{2}\gamma_d + \sum_{y=2}^{d-t_{min}} \left[(y-1)p + \frac{p(p-1)}{2}\right] * 1 + [(d - t_{min})p \\
&\quad + \frac{p(p-1)}{2}]\gamma_{t_{min}} \\
&= \frac{p(p-1)}{2}(\gamma_d + \gamma_{t_{min}}) + p(d - t_{min})\gamma_{t_{min}} + \sum_{y=2}^{d-t_{min}} (y-1)p \\
&\quad + \sum_{y=2}^{d-t_{min}} \frac{p(p-1)}{2} \\
&= \frac{-p(p-1)(p-2)}{6} \qquad \text{by (4.15)-(4.17)}
\end{aligned}$$

Next, we analyze the second term:

$$\begin{aligned}\sum_{t=d+1}^{d+p-1}\sum_{k=t-p+1}^{d}(d-k)\delta_t &= [(p-2)+(p-3)+\ldots+1]\,\delta_{d+1}\\ &\quad+[(p-3)+(p-4)+\ldots+1]\,\delta_{d+2}+\ldots+1\delta_{d+p-2}+0\delta_{d+p-1}\\ &= \frac{p(p-1)(p-2)}{6} \qquad \text{by (4.18)}\end{aligned}$$

Summarizing,

$$\sum_{t=t_{min}}^{d}\sum_{k=t-p+1}^{t}(d-k)\gamma_t+\sum_{t=d+1}^{d+p-1}\sum_{k=t-p+1}^{d}(d-k)\delta_t=\frac{-p(p-1)(p-2)}{6}+\frac{p(p-1)(p-2)}{6}=0.$$

(4.14): $\sum_{t=d+1}^{d+p-1}\sum_{k=d+1}^{t}(k-d-1)\delta_t+\sum_{t=d+p}^{t_{max}}\sum_{k=t-p+1}^{t}(k-d-1)\mu_t \overset{?}{\leq} 0$

By (4.18), the first term can be shown to equal $\frac{p(p-1)(p-2)}{6}$. The derivation is similar to that of the second term of (4.13).

We next analyze the second term:

$$\begin{aligned}\sum_{t=d+p}^{t_{max}}\sum_{k=t-p+1}^{t}(k-d-1)\mu_t &= \sum_{y=1}^{t_{max}-d-p+1}\left[(y-1)p+\frac{p(p-1)}{2}\right]\mu_{d+p-1+y}\\ &= \frac{p(p-1)}{2}\mu_{d+p}+\sum_{y=2}^{t_{max}-d-p}\left[(y-1)p+\frac{p(p-1)}{2}\right]*1\\ &\quad+\left[p(t_{max}-d-p)+\frac{p(p-1)}{2}\right]\mu_{t_{max}}\\ &= \frac{-p(p-1)(p-2)}{6} \qquad \text{by (4.19)-(4.21)}\end{aligned}$$

Summarizing,

$$\sum_{t=d+1}^{d+p-1}\sum_{k=d+1}^{t}(k-d-1)\delta_t+\sum_{t=d+p}^{t_{max}}\sum_{k=t-p+1}^{t}(k-d-1)\mu_t=\frac{p(p-1)(p-2)}{6}-\frac{p(p-1)(p-2)}{6}=0.$$

(4.8): $\gamma_t \overset{?}{\leq} 1,\ t=t_{min},\ldots,d$. In (4.16), $\gamma_t=1$ for $t_{min}+1\leq t\leq d-1$, therefore we only need to show that $\gamma_{t_{min}}\leq 1$, and $\gamma_d\leq 1$.

$\gamma_d=\frac{-(p+1)(p-1)}{12K}-\frac{K+p}{2}+1=\frac{-(p^2-1)}{12K}-\frac{K+p}{2}+1<1$. Note that the first term is nonpositive, and the second term is strictly greater than zero because $K\geq p\geq 1$.

$\gamma_{t_{min}}=-\gamma_d-\frac{(p-1)}{2}-K+1\overset{?}{\leq}1$ which by substituting γ_d becomes $\frac{(p+1)(p-1)}{12K}+\frac{K+p}{2}\overset{?}{\leq}\frac{p-1}{2}+K+1$. The first term on the left hand side is less than or equal to $\frac{p-1}{2}$ because $\frac{p+1}{6K}\leq\frac{p+1}{6p}\leq\frac{1}{2}$ when $K\geq p\geq 1$. Similarly, $\frac{K+p}{2}\leq\frac{K+K}{2}\leq K$ because $K\geq p$. Thus $\gamma_{t_{min}}<1$.

(4.9): By construction. See (4.18).

(4.10): $\mu_t \overset{?}{\leq} 1,\ t = d+p, \ldots, t_{max}$. In (4.20), $\mu_t = 1$ for $d+p+1 \leq t \leq t_{max}-1$, therefore we only need to show that $\mu_{d+p} \leq 1$, and $\mu_{t_{max}} \leq 1$.

$\mu_{d+p} = \frac{\frac{N^2}{2}+N(\frac{-p}{2}-1)+1+(p-1)\left(\frac{p+13}{12}\right)}{-(N-p)} \leq 0 < 1$. In the numerator, the function $g(N) = \frac{N^2}{2}+N(\frac{-p}{2}-1)+1$ is non-decreasing for $N \geq \frac{p}{2}+1$. Thus, $g(N) \geq g(2p) = (p-1)^2 \geq 0$ because $N \geq 2p \geq \frac{p}{2}+1$ for $p \geq 1$. The remaining term in the numerator is nonnegative because $p \geq 1$. The denominator is strictly negative because $N \geq 2p$ and $p \geq 1$. Thus $\mu_{d+p} < 1$.

$\mu_{t_{max}} = -\mu_{d+p} - \frac{p-1}{2} - N + p + 1 \overset{?}{\leq} 1$ which by substituting μ_{d+p} is equivalent to

$-\frac{N^2}{2}+N\left(p-\frac{1}{2}\right)+p+\frac{(p-1)(p+1)}{12} \overset{?}{\leq} \frac{p(p+1)}{2}$. The function $h(N) = -\frac{N^2}{2}+N\left(p-\frac{1}{2}\right)+p$ is non-increasing for $N \geq p-\frac{1}{2}$. Thus, for $N \geq 2p$ and $p \geq 1$, $h(N) \leq h(2p) = 0$, which together with $\frac{(p-1)(p+1)}{12} \leq \frac{p(p+1)}{2}$ when $p \geq 1$ yields $\mu_{t_{max}} \leq 1$.

Thus, the solution given in (4.15)-(4.21) is a feasible solution to the dual of CP. ∎

Note that the conditions $t_{min} \leq d-p$ and $t_{max} \geq d+2p$ are sufficient, but not necessary for the coefficients in Theorem 4.6 to be optimal. For instance, assume $d = 10$, $p = 4$, $\epsilon = 1$, $\pi = 1$. Choose $t_{min} = 8 > d-p$ and $t_{max} = 14 < d+2p$. Then, the optimal solution to CP is $m_E = 0.25,\ m_T = 0,\ a_E = -0.375$, and $a_T = 1$. These values are equal to $m_E = \frac{\epsilon}{p}$ and $a_E = \frac{\epsilon}{p}(\frac{-p}{2}+\frac{1}{2})$, but different than $m_T = \frac{\pi}{p}$ and $a_T = \frac{\pi}{p}(\frac{p}{2}+\frac{1}{2})$, i.e., the cost coefficients characterized in Theorem 4.6 are not optimal. The planning horizon extends for such a short duration after the due date that the optimal cost coefficients are constant in the interval $[11,\ 14]$. However, if t_{max} is increased to $d+2p = 18$, then the cost coefficients in Theorem 4.6 are optimal even if $d - t_{min} < p$.

References

Baker, K. R. and Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 38(1):22–36.

Bülbül, K., Kaminsky, P., and Yano, C. (2001). Earliness/tardiness scheduling in a job shop with intermediate inventory holding costs. Working paper. Department of IEOR, University of California at Berkeley.

Davis, J. and Kanet, J. (1993). Single-machine scheduling with early and tardy completion costs. *Naval Research Logistics*, 40(1):85–101.

Dessouky, M., Kijowski, B., and Verma, S. (1999). Simultaneous batching and scheduling for chemical processing with earliness and tardiness penalties. *Production and Operations Management*, 8(4):433–444.

Dyer, M. and Wolsey, L. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26(2–3):255–270.

Garey, M., Tarjan, R., and Wilfong, G. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13(2):330–348.

Gelders, L. and Kleindorfer, P. (1974). Coordinating aggregate and detailed scheduling decisions in the one-machine job shop. i. theory. *Operations Research*, 22(1):46–60.

Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

Kanet, J. and Sridharan, V. (2000). Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, 48(1):99–110.

Labetoulle, J., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1984). Preemptive scheduling of uniform machines subject to release dates. In *Progress in Combinatorial Optimization*, pages 245–261. Academic Press.

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

Mazzini, R. and Armentano, V. (2001). A heuristic for single machine scheduling with early and tardy costs. *European Journal of Operations Research*, 128(1):129–146.

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12.

Mosheiov, G. (1996). A note on pre-emptive scheduling with earliness and tardiness costs. *Production Planning and Control*, 7(4):401–406.

Nandkeolyar, U., Ahmed, M., and Sundararaghavan, P. (1993). Dynamic single-machine-weighted absolute deviation problem: predictive heuristics and evaluation. *International Journal of Production Research*, 31(6):1453–1466.

Ovacik, I. M. and Uzsoy, R. (1997). *Decomposition methods for complex factory scheduling problems*. Kluwer Academic Publishers, Boston, Massachusetts.

Potts, C. and Van Wassenhove, L. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1(5):177–181.

Srinivasan, V. (1971). A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly*, 18(3):317–327.

Szwarc, W. and Mukhopadhyay, S. (1995). Optimal timing schedules in earliness-tardiness single machine sequencing. *Naval Research Logistics*, 42(7):1109–1114.

Verma, S. and Dessouky, M. (1998). Single-machine scheduling of unit-time jobs with earliness and tardiness penalties. *Mathematics of Operations Research*, 23(4):930–943.

Yano, C. and Kim, Y. (1991). Algorithms for a class of single-machine weighted tardiness and earliness problems. *European Journal of Operations Research*, 52(2):167–178.