Flow Shop Scheduling with Earliness, Tardiness and Intermediate Inventory Holding Costs

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

May 2003

Abstract

We consider the problem of scheduling customer orders in a flow shop with the objective of minimizing the sum of tardiness, earliness (finished goods inventory holding) and intermediate (work-in-process) inventory holding costs. We formulate this problem as an integer program, and based on approximate solutions to two different, but closely related, Dantzig-Wolfe reformulations, we develop heuristics to minimize the total cost. We exploit the duality between Dantzig-Wolfe reformulation and Lagrangian relaxation to enhance our heuristics. This combined approach enables us to develop two different lower bounds on the optimal integer solution, together with intuitive approaches for obtaining near-optimal feasible integer solutions. To the best of our knowledge, this is the first paper that applies column generation to a scheduling problem with different types of strongly $\mathcal{NP}$-hard pricing problems which are solved heuristically. The computational study demonstrates that our algorithms have a significant speed advantage over alternate methods, yield good lower bounds, and generate near-optimal feasible integer solutions for problem instances with many machines and a realistically large number of jobs.

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

1 Introduction

We address the problem of scheduling flow shops in settings with penalties (explicit or implicit) for tardiness in delivering customer orders, as well as costs for holding both finished goods and work-in-process inventory. Our work was motivated, in part, by applications in the biotechnology industry. The vast majority of biotechnology products require a well-defined series of processing steps, and processing usually occurs in batches, at least until the final packaging steps. Thus, in many instances, the manufacturing arrangement may be accurately characterized as a flow shop, with batches moving from one process to the next.

The scheduling of the various process steps is complicated by the relative fragility of these products between successive processing stages. Delays between stages increase the chances of contamination or deterioration. Some products require refrigeration and storage in other types of specially-controlled facilities that are expensive to build and maintain. For instance, the authors are aware of an example in which live blood cells are used to generate certain blood components that are extracted and further processed in the manufacture of a treatment for a particular blood disorder. Sterile conditions are essential throughout the process, including during any waiting between stages. At various work-in-process stages, the product requires very precise temperature control, physical separation of batches, etc. Thus, the cost of holding work-in-process inventory can be substantial, and at some stages, may greatly exceed the cost of holding finished goods inventory.

There are also "low-tech" examples in which intermediate inventory holding costs are important and quite different in value from finished goods holding costs. For example, the steel used in automobile body parts is coated (to delay corrosion) before the steel is blanked (cut) and stamped into the proper shapes. Until the blanking stage, the steel is held in the form of coils and is relatively compact. After coating, the product is more stable (less subject to corrosion) than before coating, so the inventory holding costs may be lower after coating. On the other hand, after the parts are stamped, they must be stored in containers or racks, and require significantly more space than the equivalent amount of steel in coil form. This is an instance in which total earliness (at all stages) is not a reliable indicator of the cost of holding inventory, and distinguishing among inventory at different stages is important in making good scheduling decisions.

Problems with earliness (finished goods holding) costs and tardiness costs have been studied extensively in single-machine contexts. (See Kanet and Sridharan (2000) and references therein). Also, many results exist for parallel machine earliness/tardiness scheduling problems, especially when all jobs have the same due date. (See Sundararaghavan and Ahmed (1984), Kubiak et al. (1990), Federgruen and Mosheiov (1996), Chen and Powell (1999a) and Chen and Powell (1999b).) However, relatively little research has been done on flow- and job shop environments when earliness and tardiness costs are present in the objective function. Some examples include Sarper (1995) and Sung and Min (2001) who consider 2-machine flow shop common due date problems, and Chen and Luh (1999) who develop heuristics for a job shop weighted earliness and

tardiness problem. Work-in-process inventory holding costs have been explicitly incorporated in Park and Kim (2000), Kaskavelis and Caramanis (1998), and Chang and Liao (1994) for various flow- and job shop scheduling problems, although these papers consider different objective functions and approaches than ours.

For complex scheduling problems, a common approach involves decomposing the original problem into a set of independent subproblems by relaxing some of the constraints. The aim is to achieve a decomposition in which the subproblems are relatively easy to solve, and to use the collective solution from these subproblems as the basis for constructing an effective solution for the original problem. These decomposition methods include traditional mathematical programming decomposition methods, including Lagrangian relaxation (LR) and Dantzig-Wolfe (DW) decomposition, as well as decomposition methods that were designed specifically for scheduling problems, such as shifting bottleneck algorithms. In this paper, we develop two different Dantzig-Wolfe reformulations of the problem described above, and solve them using column generation enhanced by Lagrangian relaxation and use of artificial variables in the linear programming (LP) master problems. We also construct high quality feasible solutions for the original problem from the information in the solution of the Dantzig-Wolfe reformulations. In Section 3, we first describe our initial approach and then discuss in detail each of the modifications to the column generation algorithm.

Column generation has recently been applied successfully to a variety of scheduling problems. Van den Akker et al. (2000) develop a framework for the Dantzig-Wolfe reformulation of time-indexed formulations of machine scheduling problems and apply it to the single-machine weighted completion time problem with unequal ready times. Van den Akker et al. (2002) report that the LP relaxation of their set covering formulation of the single-machine unrestrictive common due date problem with asymmetric weights yields the optimal integer solution in all of their randomly generated instances. Chen and Powell (1999a), Chen and Powell (1999b), Van den Akker et al. (1999a), Lee and Chen (2000) and Chen and Lee (2002), among others, have developed column generation methods for parallel machine scheduling problems. In particular, Chen and Powell (1999a) and Chen and Lee (2002) consider two parallel machine scheduling problems with the objective of minimizing the total weighted earliness and tardiness. In both cases, a set partitioning formulation is solved by column generation. In the first paper, there is a common unrestrictive due date for all jobs, and in the second, the authors consider a common due window which may have a restrictively small starting point.

When decomposition approaches are employed for parallel machine scheduling, the subproblems are typically identical, or of similar form. In our flow shop scheduling problem, however, the subproblems may be quite different from each other; nevertheless, we develop an algorithm that can solve all of them effectively. In all of the parallel machine scheduling problems mentioned above, the pricing problems are pseudo-polynomial and solved optimally by a dynamic programming algorithm. To the best of our knowledge, this paper is the first attempt to apply column generation to a machine scheduling problem where some subproblems are strongly $\mathcal{NP}$-hard. (See Section 3.) In Section 5, we demonstrate computationally that approximate

solutions for the subproblems do not compromise the overall solution quality. Furthermore, we are able to derive insights from the complementary properties of Dantzig-Wolfe reformulation and Lagrangian relaxation with little additional effort. Recently, Van den Akker et al. (2002) combined Lagrangian relaxation with column generation in order to solve a single-machine common due date problem. The authors enhance their column generation algorithm in several ways by obtaining an alternate lower bound from Lagrangian relaxation, and by tuning their LP master problem formulation based on insights from the Lagrangian model. The algorithm of Cattrysse et al. (1993), who improve the values of the dual variables from the optimal solution of the restricted LP master problem by performing Lagrangian iterations before solving the pricing problems, is in some sense similar to our approach in Section 3.3, although we do not pass dual variables from Lagrangian relaxation to column generation. In addition, Dixon and Poh (1990) and Drexl and Kimms (2001) consider Dantzig-Wolfe reformulation and Lagrangian relaxation as alternate methods, but do not use both approaches together.

The key to developing effective decomposition approaches is finding a "good" set of coupling constraints to relax. On one hand, the relaxed problem needs to be relatively easy to solve, but on the other, it should provide a reasonable approximation of the original problem. Typically, there is a trade-off between these two goals. Relatively easy subproblems frequently imply looser bounds, and thus less useful information for solving the original problem. In this paper, we relax the operation precedence constraints, as opposed to the machine capacity constraints that are most frequently relaxed in the scheduling literature. Although this approach results in more difficult subproblems, we solve these approximately and obtain effective solutions for the original problem. See Fisher (1971) for a general discussion of Lagrangian approaches to resource-constrained scheduling problems in which capacity constraints are relaxed. More recent examples of Lagrangian algorithms can be found in Chang and Liao (1994) and Kaskavelis and Caramanis (1998). Note that Van de Velde (1990) and Hoogeveen and van de Velde (1995) apply Lagrangian relaxation to the 2-machine flow shop total completion time problem with equal ready times by relaxing the operation precedence constraints, and solve this problem effectively by developing polynomial algorithms for their subproblems. Also, Chen and Luh (1999) solve a job shop total weighted earliness/tardiness problem by Lagrangian relaxation where the operation precedence constraints are relaxed. In their model, there may be several machines of the same type, and thus their subproblems are parallel machine total weighted completion time problems, which are solved approximately. They compare their approach to an alternate Lagrangian relaxation algorithm where machine capacity constraints are relaxed. Their results indicate that relaxing operation precedence constraints yields superior results in memory and computation time, especially when the due dates are not tight.

In this paper, we develop heuristics for the m-machine flow shop earliness/tardiness scheduling problem with intermediate inventory holding costs based on the LP relaxation of its Dantzig-Wolfe reformulation solved approximately by column generation. We compare two different reformulations in which the column

generation scheme is enhanced by the solution of an equivalent Lagrangian relaxation formulation and use of artificial variables in the LP master problem. The subproblems are solved heuristically. From tight lower bounds for the subproblems, we develop two different lower bounds for the overall problem. Additionally, from the approximate solution of the LP relaxation of the Dantzig-Wolfe reformulation, we develop a technique to construct good feasible solutions for the original problem.

In Section 2, we present a formal description of the problem, and discuss complexity and modeling issues. In Section 3, we develop the mathematical models, explain how we combine column generation and Lagrangian relaxation, present our lower bounds, and the heuristics to obtain feasible integer solutions from the (not necessarily optimal) solution of the Dantzig-Wolfe decomposition. In Section 4, we discuss the various steps of our column generation algorithm, including an effective approach for the subproblems. Finally, in Section 5 we present computational results, and in Section 6, we conclude and discuss future research directions.

2 Problem Description

Consider a non-preemptive flow shop with m machines in series and n jobs. Each job visits the machines in the sequence $i = 1, \ldots, m$ and is processed without preemption on each machine. We do not require a permutation sequence, i.e., the sequence of jobs may differ from one machine to another. Associated with each job j, $j = 1, \ldots, n$, are several parameters: p_{ij}, the processing time for job j on machine i; r_j, the ready time of job j; d_j, the due date for job j; h_{ij}, the holding cost per unit time for job j while it is waiting in the queue before machine i; ϵ_j, the earliness cost per unit time if job j completes processing on the final machine before time d_j; and π_j, the tardiness cost per unit time if job j completes processing on the final machine after time d_j. All ready times, processing times and due dates are assumed to be integer.

The objective is to determine a feasible schedule that minimizes the sum of costs across all jobs. For a given schedule S, let C_{ij} be the time at which job j finishes processing on machine i, and w_{ij} be the time job j spends in the queue before machine i. The sum of costs for all jobs in schedule S, $C(S)$, can be expressed as follows:

$$C(S) = \sum_{j=1}^{n} h_{1j}(C_{1j} - p_{1j} - r_j) + \sum_{i=2}^{m}\sum_{j=1}^{n} h_{ij}w_{ij} + \sum_{j=1}^{n} \epsilon_j E_j + \pi_j T_j, \tag{2.1}$$

where $E_j = \max(0, d_j - C_{mj})$ and $T_j = \max(0, C_{mj} - d_j)$. Thus, the m-machine flow shop scheduling problem with earliness, tardiness and intermediate inventory holding costs can be formulated as:

$$z_{Fm} = \min_{S} \quad C(S) \tag{2.2}$$

s.t.

(Fm)

$$C_{1j} \geq r_j + p_{1j} \qquad \forall j \tag{2.3}$$

$$C_{i-1j} - C_{ij} + w_{ij} = -p_{ij} \qquad i = 2, \ldots, m, \ \forall j \tag{2.4}$$

$$C_{ik} - C_{ij} \geq p_{ik} \textbf{ or } C_{ij} - C_{ik} \geq p_{ij} \qquad \forall i, j, k \tag{2.5}$$

$$C_{mj} + E_j - T_j = d_j \qquad \forall j \tag{2.6}$$

$$w_{ij} \geq 0 \qquad i = 2, \ldots, m, \ \forall j \tag{2.7}$$

$$E_j, T_j \geq 0 \qquad \forall j \tag{2.8}$$

The constraints (2.3) prevent processing of jobs before their respective ready times on machine one. Constraints (2.4), referred to as operation precedence constraints, ensure that jobs follow the processing sequence machine 1, machine 2,..., machine m. Machine capacity constraints (2.5) ensure that a machine processes only one job at a time, and a job is finished once started. Constraints (2.6) relate the completion times on the final machine to earliness and tardiness values. In the rest of the paper, z_M denotes the optimal objective function value of a model M, where M refers to any of the models we consider.

In classifying scheduling problems, we follow the three-field notation of Graham et al. (1979). Problem Fm is represented as $Fm/r_j/\sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij}w_{ij} + \sum_{j=1}^{n}(\epsilon_j E_j + \pi_j T_j)$, where in the first field, Fm denotes a flow shop with m machines, and the entry r_j in the second field indicates that the ready times may be unequal. Fm is strongly $\mathcal{NP}$-hard because a single-machine special case with all earliness costs equal to zero, the single-machine weighted tardiness problem $1/r_j/\sum \pi_j T_j$, is known to be strongly $\mathcal{NP}$-hard (Lenstra et al. (1977)).

Although the single-machine earliness/tardiness problem has attracted considerable attention in the literature, to the best of our knowledge, this paper is the first attempt to develop an effective approach for the m-machine flow shop earliness/tardiness problem. A survey of the early research on single-machine problems appears in Baker and Scudder (1990), and more recent results are cited in Kanet and Sridharan (2000) and the references therein.

Observe that Fm would be a linear program if we knew the sequence of jobs on each machine. This is a property of earliness/tardiness problems that is often used to develop a two-phase heuristic: first, a good job processing sequence is determined, and then idle time is inserted either by solving a linear program, or by using a specialized algorithm that exploits the structure of the optimal solution with fixed job processing sequences. This second step is commonly referred to as *timetabling*. (See Kanet and Sridharan (2000) for a more detailed discussion of timetabling.) For our problem, once jobs are sequenced, and assuming jobs are renumbered in sequence order on each machine, the optimal *schedule* is found by solving the linear program

TTFm below.

(TTFm)

$$z_{TTFm} = \min_{C_{ij}} \quad C(S) \tag{2.9}$$

$$\text{s.t.}$$

$$(2.3) - (2.4) \text{ and } (2.6) - (2.8)$$

$$C_{ij} - C_{ij-1} \geq p_{ij} \qquad \forall i, j = 2, \ldots, n \tag{2.10}$$

In our approach, we reformulate Fm, solve its linear programming relaxation approximately, construct job processing sequences on each machine from this solution, and find the optimal schedule given these sequences by solving TTFm. The strength of our algorithm lies in its ability to identify near-optimal job processing sequences.

In Fm, the machine capacity constraints are modeled as a set of disjunctive constraints (2.5), and we have to specify a mechanism such that exactly one of each pair of constraints is imposed depending on the job sequence. One approach is to introduce binary variables δ_{ijk}, such that $\delta_{ijk} = 1$ if job j precedes job k on machine i, and zero otherwise. Then, we can add the following constraints to Fm and drop (2.5) from the formulation, where M is a large enough constant.

$$C_{ij} - C_{ik} \geq p_{ij} - M\delta_{ijk} \quad \forall i, j, k \tag{2.11}$$

$$\delta_{ijk} \in \{0, 1\} \quad \forall i, j, k \tag{2.12}$$

Unfortunately, in the linear programming relaxation of Fm, the constraints (2.11) are not binding and the bound obtained is loose. Recall that we are interested in near-optimal feasible integer solutions, and for this purpose, we require a relatively tight LP relaxation.

Dyer and Wolsey (1990) introduced time-indexed formulations of scheduling problems. These formulations have attracted much attention in recent years because they often have strong LP relaxations, and they frequently play an important role in approximation algorithms for certain classes of scheduling problems. For instance, Goemans et al. (1999) and Schulz and Skutella (1997) use time-indexed formulations to develop approximation algorithms for $1/r_j/\sum \beta_j C_j$ and $R/r_j/\sum \beta_j C_j$, respectively, where R in the first field stands for unrelated parallel machines. In time-indexed formulations, job finish times are represented by binary variables, and constraints such as (2.5) are replaced by a large set of generalized upper bound constraints which tighten the formulation. (For details, see Dyer and Wolsey (1990), Sousa and Wolsey (1992) and Van den Akker et al. (1999b).) In general, the major disadvantage of time-indexed formulations is their size which grows rapidly with the number of jobs and the length of the processing times. (See the survey by Queyranne and Schulz (1994).)

The concept of time-indexed formulations is relevant to our approach for several reasons. First, in Proposition 3.11 we show that the Dantzig-Wolfe reformulation of the time-indexed formulation for our

problem (see TIFm below) is the same as that of Fm. This implies that the optimal objective function value of the LP relaxation of our Dantzig-Wolfe reformulation is at least as large as the optimal objective function value of the LP relaxation of TIFm. Also, we use the LP relaxation of TIFm as a benchmark in our computational study to demonstrate that our heuristic approach has a significant speed advantage, with little compromise in solution quality.

In order to present the time-indexed formulation for our problem, let $\underline{t}_{ij} = r_{ij} + p_{ij}$ represent the earliest possible time job j can finish on machine i, where the *operational* ready time r_{ij} of job j on machine i is defined as $r_{ij} = r_j + \sum_{l=1}^{i-1} p_{lj}$, and $r_{1j} = r_j$. Note that $t_{max} = \max_j \max(r_j, d_j) + P$, where $P = \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}$ is the sum of processing times on all machines, is the latest time that any job could conceivably be finished in an optimal schedule, accounting for potential idle time. Thus, let the latest time job j can finish on machine i be $\bar{t}_{ij} = t_{max} - \sum_{l=i+1}^{m} p_{lj}$, and define the processing interval for job j on machine i as $H_{ij} = \{k \in \mathbb{Z} | \underline{t}_{ij} \leq k \leq \bar{t}_{ij}\}$. Similarly, the processing interval for machine i is defined as $H_i = \{k \in \mathbb{Z} | \min_j \underline{t}_{ij} \leq k \leq \max_j \bar{t}_{ij}\}$. Finally, let x_{ijk} equal one if job j finishes processing on machine i at time k, and zero otherwise. The time-indexed formulation is:

(TIFm)

$$z_{TIFm} = \min_{\substack{x_{ijk} \\ w_{ij}}} \quad \sum_{j=1}^{n} \sum_{k \in H_{1j}} h_{1j}(k - p_{1j} - r_j) x_{1jk} + \sum_{i=2}^{m} \sum_{j=1}^{n} h_{ij} w_{ij} + \sum_{j=1}^{n} \left[\sum_{\substack{k \in H_{mj} \\ k \leq d_j}} \epsilon_j (d_j - k) + \sum_{\substack{k \in H_{mj} \\ k > d_j}} \pi_j (k - d_j) \right] x_{mjk} \tag{2.13}$$

s.t.

$$\sum_{k \in H_{ij}} x_{ijk} = 1 \qquad \forall i, j \tag{2.14}$$

$$\sum_{k \in H_{i-1j}} k x_{i-1jk} - \sum_{k \in H_{ij}} k x_{ijk} + w_{ij} = -p_{ij} \qquad i = 2, \ldots, m, \forall j \tag{2.15}$$

$$\sum_{j=1}^{n} \sum_{\substack{t \in H_{ij} \\ k \leq t \leq k + p_{ij} - 1}} x_{ijt} \leq 1 \qquad \forall i,\ k \in H_i \tag{2.16}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j,\ k \in H_{ij} \tag{2.17}$$

$$w_{ij} \geq 0 \qquad j = 2, \ldots, m, \forall j \tag{2.18}$$

Constraints (2.14) ensure that each job is processed exactly once on each machine. The generalized upper bound constraints (2.16) ensure that each machine processes at most one job at any time. Observe that the operation precedence constraints (2.15) are equivalent to the operation precedence constraints (2.4) in Fm because there is exactly one $k \in H_{ij}$ for which $x_{ijk} = 1$, and thus $\sum_{k \in H_{ij}} k x_{ijk} = C_{ij}$. Note that these constraints introduce coefficients with value $O(t_{max})$ into the constraint matrix. One can also consider an alternate, potentially tighter formulation as follows: drop w_{ij} from the formulation, introduce binary variables w_{ijk} that take the value one if job j is waiting in queue before machine i at time k and zero

otherwise, make the appropriate changes to the objective function, and replace constraints (2.15) with the following:

$$\sum_{\substack{t\in H_{i-1j}\\ t\le k}} x_{i-1jt} - w_{ij(k+1)} = \sum_{\substack{t\in H_{ij}\\ t\le k+p_{ij}}} x_{ijt} \quad i=2,\dots,m,\forall j,\ k\in H_{i-1j} \tag{2.19}$$

$$w_{ijk} \ge 0 \quad \forall i,\ j,\ k\in H_{i-1j}\setminus\{\underline{t}_{i-1j}\}\cup\{\bar{t}_{i-1j}+1\} \tag{2.20}$$

As we mentioned previously, time-indexed formulations can grow very rapidly with the number of jobs. Indeed, in TIFm there are $O(2mn+mt_{max})$ constraints, but if (2.15) is replaced by (2.19), then there are $O(mn+m(n+1)t_{max})$ constraints, which simply becomes unmanageable. For this reason, we have elected to utilize a formulation that includes a single operation precedence constraint per job per machine, and in the next section we explain our solution approach for this formulation.

3 Solution Approach

In this section, first we present the Dantzig-Wolfe reformulation of Fm, as well as the corresponding Lagrangian relaxation model, and then we discuss how we can exploit the duality between these two approaches. In fact, our observations concerning the subproblem and the Lagrangian objective functions lead us to an alternate DW reformulation and the corresponding LR model.

Both of the alternate DW reformulations can be used to solve the m-machine flow shop earliness/tardiness scheduling problem with intermediate inventory holding costs, but the convergence is slow when we solve these DW reformulations with column generation. We note that this "tailing-off" effect is rooted in the relative magnitudes of the dual variables and the inventory holding costs, and we explain this phenomenon by using the "shadow price" interpretation of the dual variables. Furthermore, we develop a two-phase algorithm that alleviates this "tailing-off" effect to a great extent. In the first phase, artificial variables are introduced into the DW reformulation to impose a structure on the dual variables which speeds up the convergence. The artificial variables are eventually driven to zero in the second phase.

3.1 Initial Models

First DW Reformulation

We formulate the m-machine flow shop earliness/tardiness scheduling problem with intermediate inventory holding costs as an integer programming problem with an exponential number of variables that represent *capacity-feasible* schedules for individual machines. Although there is an infinite number of capacity-feasible schedules on each machine if the completion times are continuous variables, the last job on the last machine will finish at or before t_{max} in any optimal solution to our problem. Furthermore, by the integrality of ready

times, due dates and processing times, there exists an optimal solution in which all completion times are integral. Thus, without loss of generality, we can assume that the number of possible schedules on each machine is finite but exponential. Additionally, we require that these *capacity-feasible* schedules satisfy the *operational* ready time constraints, $C_{ij} \geq r_{ij} + p_{ij}$. Let the set of all schedules on machine i be denoted by S_i. Each of these schedules $S_i^k,\ k = 1, \ldots, K_i$, where $K_i = |S_i|$, is defined by a set of completion times $\{C_{ij}^k\}$, i.e., $S_i^k = \{C_{ij}^k \in \mathbb{Z},\ j = 1, \ldots, n |\ \underline{t}_{ij} \leq C_{ij}^k \leq \overline{t}_{ij}\ \forall j$, and $C_{ij}^k,\ j = 1 \ldots, n$, satisfy capacity constraints on machine $i\}$. Let x_i^k be a binary variable that takes the value one if we choose S_i^k on machine i, zero otherwise. The earliness and tardiness of job j in schedule S_m^k are computed as $E_j^k = \max(0,\ d_j - C_{mj}^k)$, and $T_j^k = \max(C_{mj}^k - d_j,\ 0)$, respectively. Then, the *integer programming master problem* IM1 below is equivalent to Fm.

$$z_{IM1} = \min_{x_i^k,\ w_{ij}} \quad \sum_{k=1}^{K_1} \left[\sum_{j=1}^{n} h_{1j}(C_{1j}^k - r_j - p_{1j}) \right] x_1^k \tag{3.1}$$

$$+ \sum_{i=2}^{m} \sum_{j=1}^{n} h_{ij} w_{ij} \tag{3.2}$$

$$+ \sum_{k=1}^{K_m} \left[\sum_{j=1}^{n} \left(\epsilon_j E_j^k + \pi_j T_j^k \right) \right] x_m^k \tag{3.3}$$

(IM1) s.t.

$$(\mu_{ij}) \quad \sum_{k=1}^{K_{i-1}} C_{i-1,j}^k x_{i-1}^k - \sum_{k=1}^{K_i} C_{ij}^k x_i^k + w_{ij} = -p_{ij} \quad i = 2, \ldots, m,\ \forall j \tag{3.4}$$

$$(\gamma_i) \quad \sum_{k=1}^{K_i} x_i^k = 1 \quad i = 1, \ldots, m \tag{3.5}$$

$$x_i^k \in \{0,\ 1\} \quad i = 1, \ldots, m,\ k = 1, \ldots, K_i \tag{3.6}$$

$$w_{ij} \geq 0 \quad i = 2, \ldots, m,\ \forall j \tag{3.7}$$

The constraints (3.4) are the operation precedence constraints that correspond to constraints (2.4) in Fm and (2.15) in TIFm, and the convexity constraints (3.5) ensure that we select exactly one feasible schedule on each machine. The Greek letters in the parentheses on the left are the dual variables associated with the constraints.

IM1 is an integer program with an exponential number of binary variables. Below, we show that solving the LP relaxation of IM1 by column generation is strongly $\mathcal{NP}$-hard. Therefore, rather than devising an optimal algorithm, we develop an approach for obtaining a near-optimal solution for IM1 *quickly*, along with *tight* lower bounds on its optimal objective value. Our strategy is based on solving the linear programming relaxation of IM1, which we call LM1, *approximately*, and then using this approximate LP solution to construct near-optimal feasible schedules for the original problem. In the linear programming master problem LM1, the constraints (3.6) are replaced by:

$$x_i^k \geq 0,\ i = 1, \ldots, m,\ k = 1, \ldots, K_i \tag{3.8}$$

Since there are exponentially many capacity-feasible schedules on each machine, it is not possible to include

them all explicitly in the formulation. Therefore, we use column generation to solve LM1 by adding new schedules with negative reduced costs to the model as needed. The linear programming master problem, which includes only a subset of all possible schedules, is called the *restricted linear programming master problem* RLM1. To facilitate the presentation of the *pricing problems*[1] that we use to identify which negative reduced cost columns to add, we next present the dual of LM1. From linear programming theory, recall that the reduced cost of a variable is the infeasibility in the corresponding dual constraint.

(DLM1)

$$z_{DLM1} = \max_{\mu_{ij},\ \gamma_i} \quad \sum_{i=2}^{m}\sum_{j=1}^{n} -p_{ij}\mu_{ij} + \sum_{i=1}^{m}\gamma_i \tag{3.9}$$

s.t.

$$\sum_{j=1}^{n} \mu_{2j}C_{1j}^k + \gamma_1 \le \sum_{j=1}^{n} h_{1j}(C_{1j}^k - r_j - p_{1j}) \quad k = 1,\dots,K_1 \tag{3.10}$$

$$\sum_{j=1}^{n} (-\mu_{ij}C_{ij}^k + \mu_{i+1j}C_{ij}^k) + \gamma_i \le 0 \quad i = 2,\dots,m-1$$

$$k = 1,\dots,K_i \tag{3.11}$$

$$\sum_{j=1}^{n} -\mu_{mj}C_{mj}^k + \gamma_m \le \epsilon_j E_j^k + \pi_j T_j^k \quad k = 1,\dots,K_m \tag{3.12}$$

$$\mu_{ij} \le h_{ij} \quad i = 2,\dots,m, \forall j \tag{3.13}$$

$$\mu_{ij} \text{ unrestricted} \quad i = 2,\dots,m, \forall j$$

$$\gamma_i \text{ unrestricted} \quad \forall i$$

In order to find the schedule with the smallest reduced cost on machine 1, we need to solve the following pricing problem:

(LM1-PP1)

$$z_{LM1}^{PP1} = \min_{C_{1j}} \quad \sum_{j=1}^{n}(h_{1j} - \mu_{2j})C_{1j} - \sum_{j=1}^{n} h_{1j}(r_j + p_{1j}) - \gamma_1 \tag{3.14}$$

s.t.

machine capacity constraints on machine 1

$$\underline{t}_{1j} \le C_{1j} \le \bar{t}_{1j} \quad \forall j$$

The last two terms in (3.14) are constants, and hence the pricing problem for machine 1 looks very similar to the single-machine weighted completion time problem with unequal ready times, i.e., $1/r_j/\sum \beta_j C_j$, which is strongly $\mathcal{NP}$-hard (Lenstra et al. (1977)). Note that $1/r_j/\sum \beta_j C_j$ is a special case of Fm in which $m = 1,\ d_j = 0\ \forall j,\ h_{1j} = \beta_j\ \forall j$, and $\epsilon_j = \pi_j = 0\ \forall j$. Using this observation, we prove that LM1-PP1 is indeed strongly $\mathcal{NP}$-hard.

Proposition 3.1 *The pricing problem LM1-PP1 is strongly $\mathcal{NP}$-hard.*

Proof. See Appendix.

[1] Throughout the rest of the paper we use *pricing problem* and *subproblem* interchangeably.

Similarly, all pricing problems on machines 2 through $m-1$ are weighted completion time problems with unequal ready times; the objective function coefficients have a different structure than (3.14), however, as shown below.

(LM1-PPi)

$$z_{LM1}^{PPi} = \min_{C_{ij}} \quad \sum_{j=1}^{n} (\mu_{ij} - \mu_{i+1j}) C_{ij} - \gamma_i \tag{3.15}$$

s.t.

machine capacity constraints on machine i

$$\underline{t}_{ij} \leq C_{ij} \leq \bar{t}_{ij} \qquad \forall j$$

The pricing problem on the last machine, presented below, has a similar structure to the single-machine weighted earliness/tardiness problem with unequal ready times, i.e., $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$. The latter is strongly $\mathcal{NP}$-hard which follows from the complexity of $1/r_j/\sum \pi_j T_j$ (Lenstra et al. (1977)) which is a special case of $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$.

(LM1-PPm)

$$z_{LM1}^{PPm} = \min_{C_{mj}} \quad \sum_{j=1}^{n} [(\epsilon_j - \mu_{mj}) E_j + (\pi_j + \mu_{mj}) T_j] + \sum_{j=1}^{n} \mu_{mj} d_j - \gamma_m \tag{3.16}$$

s.t.

machine capacity constraints on machine m

$$\underline{t}_{mj} \leq C_{mj} \leq \bar{t}_{mj} \qquad \forall j$$

By recognizing that $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$ is a special case of Fm in which $m = 1$, and $h_{1j} = 0\ \forall j$, it can be proved that LM1-PPm is strongly $\mathcal{NP}$-hard. We omit the proof which is very similar to that of Proposition 3.1.

Proposition 3.2 *The pricing problem LM1-PPm is strongly $\mathcal{NP}$-hard.*

Finally, observe that $1/r_j/\sum \beta_j C_j$ is also a special case of $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$ obtained by setting all due dates equal to zero ($d_j = 0,\ \forall j$), and all tardiness costs equal to the completion time costs ($\pi_j = \beta_j,\ \forall j$). This allows us, by setting the problem parameters appropriately, to use the heuristics developed by Bülbül et al. (2001) for $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$ to solve all pricing problems approximately. In Section 4.1, we explain in greater detail how we solve these single-machine subproblems.

Corresponding Lagrangian Problem

An alternate decomposition approach is Lagrangian relaxation. For a given set of dual variables $\{\mu_{ij}\}$, the Lagrangian function LR1(μ_{ij}) is obtained from Fm by dualizing the operation precedence constraints (2.4). In fact, LM1 is equivalent to the Lagrangian dual problem LD1 which maximizes LR1(μ_{ij}) presented below over the dual variables (Lagrange multipliers). See Wolsey (1998) for a discussion of the duality between DW reformulation and Lagrangian relaxation.

(LR1(μ_{ij}))

$$z_{LR1(\mu_{ij})} = \min_{C_{ij},\, w_{ij}} \quad \sum_{j=1}^{n} (h_{1j} - \mu_{2j}) C_{1j} - \sum_{j=1}^{n} h_{1j}(r_j + p_{1j}) \tag{3.17}$$

$$+ \sum_{i=2}^{m-1} \sum_{j=1}^{n} (\mu_{ij} - \mu_{i+1j}) C_{ij} \tag{3.18}$$

$$+ \sum_{j=1}^{n} [(\epsilon_j - \mu_{mj}) E_j + (\pi_j + \mu_{mj}) T_j] + \sum_{j=1}^{n} \mu_{mj} d_j \tag{3.19}$$

$$+ \sum_{i=2}^{m} \sum_{j=1}^{n} (h_{ij} - \mu_{ij}) w_{ij} \tag{3.20}$$

$$- \sum_{i=2}^{m} \sum_{j=1}^{n} p_{ij} \mu_{ij} \tag{3.21}$$

s.t.

$$(2.3),\ (2.5)\text{-}(2.8)$$

$$\mu_{ij} \text{ unrestricted} \quad i = 2, \ldots, m,\ \forall j \tag{3.22}$$

By comparing (3.14) with (3.17), (3.15) with (3.18), and (3.16) with (3.19), we note that the Lagrangian subproblems are the same as the pricing problems of LM1 except for the dual variables associated with the convexity constraints (3.5) which are constants in the pricing problems. Therefore, if there exists a procedure to update the Lagrange multipliers that is computationally less expensive than solving a linear program, then we can potentially solve LD1 more quickly than LM1. Unfortunately, we cannot apply subgradient optimization to update the dual variables because we are not guaranteed to find a subgradient in each iteration. Recall that some of the subproblems are strongly $\mathcal{NP}$-hard and we solve all subproblems approximately. Hence, we do not attempt to solve LD1, but use LR1(μ_{ij}) to compute an alternate lower bound on the objective function value of the optimal integer solution for a given set of dual variables $\{\mu_{ij}\}$. As we discuss in Sections 3.4 and 4.1, in certain cases this improves the overall performance of our algorithm significantly.

Second DW Reformulation

Our objective is to find a near-optimal feasible solution for the original integer programming problem Fm by using information from the solution of a relaxation of the problem. Thus, it is important not only that the lower bound be tight, but also that the solution itself be easily modified into a near-optimal feasible solution. However, optimal solutions for LM1 have a property that does not generally hold for optimal integer solutions:

Lemma 3.3 *There exists an optimal solution to LM1 such that* $w_{ij}^* = 0,\ i = 2, \ldots, m,\ \forall j$.

Proof. See Appendix.

In the lemma above and throughout the rest of the paper, an asterisk denotes the optimal value of a variable. Furthermore, in Lemma 3.8 we show that for an important special inventory cost structure, *all*

optimal solutions of LM1 have this property. We also note that in LM1, the waiting time variables do not appear in the pricing problems. Thus, in an optimal solution for LM1, the inventory holding costs are not necessarily reflected either by the waiting time variables or by the schedules we generate. Consequently, we expect that LM1 will produce solutions that are poor starting points for constructing near-optimal feasible solutions; for this reason we consider an alternate DW reformulation. However, we include LM1 here because the computational experiments do not always reflect our expectations.

In the new formulation, we incorporate the effect of the waiting times implicitly in the objective function coefficients of the schedules identified by the column generation procedure, and consequently in the pricing problems as well. By substituting $w_{ij} = C_{ij} - C_{i-1j} - p_{ij}$ we obtain an alternate DW reformulation:

(IM2)

$$z_{IM2} = \min_{x_i^k} \quad \sum_{k=1}^{K_1} \left[\sum_{j=1}^{n} (h_{1j} - h_{2j}) C_{1j}^k - h_{1j}(r_j + p_{1j}) \right] x_1^k \tag{3.23}$$

$$+ \sum_{i=2}^{m-1} \sum_{k=1}^{K_i} \left[\sum_{j=1}^{n} (h_{ij} - h_{i+1j}) C_{ij}^k \right] x_i^k \tag{3.24}$$

$$+ \sum_{k=1}^{K_m} \left[\sum_{j=1}^{n} \left((\epsilon_j - h_{mj}) E_j^k + (\pi_j + h_{mj}) T_j^k \right) \right] x_m^k \tag{3.25}$$

$$+ \sum_{j=1}^{n} h_{mj} d_j - \sum_{i=2}^{m} \sum_{j=1}^{n} h_{ij} p_{ij} \tag{3.26}$$

s.t.

$$(\mu_{ij}) \qquad \sum_{k=1}^{K_{i-1}} C_{i-1j}^k x_{i-1}^k - \sum_{k=1}^{K_i} C_{ij}^k x_i^k \le -p_{ij} \qquad i = 2, \ldots, m, \ \forall j \tag{3.27}$$

$$(\gamma_i) \qquad \sum_{k=1}^{K_i} x_i^k = 1 \qquad i = 1, \ldots, m \tag{3.28}$$

$$x_i^k \in \{0, \ 1\} \qquad i = 1, \ldots, m$$

$$k = 1, \ldots, K_i \tag{3.29}$$

Relaxing the integrality constraints (3.29) yields the (restricted) linear programming master problem LM2 (RLM2). The pricing problems of LM2 differ from those of LM1 only in their objective functions. (See Table 1 below.)

Corresponding Lagrangian Problem

The corresponding Lagrangian dual problem of LM2, which we call LD2, maximizes LR2(μ_{ij}) over the nonnegative dual variables μ_{ij}. We omit the formulation of LR2(μ_{ij}) because it is structurally very similar to that of LR1(μ_{ij}). However, note that in IM2 the operation precedence constraints (3.27) must be modeled as inequalities because the waiting time variables are dropped from the formulation. Therefore, the dual variables in LR2 are not unrestricted. (Compare to (3.22) in LR1.)

Table 1 summarizes the subproblems for the models discussed in this section. For each subproblem type, the first row is the objective function to be minimized on a single machine over the appropriate set of feasible

schedules, and the second row is an associated additive constant, given the problem parameters and a set of dual variables.

	With Waiting Time Vars. (LM1/LR1)		**Without Waiting Time Vars. (LM2/LR2)**	
	DW	**LR**	**DW**	**LR**
PP1	$\sum_{j=1}^{n}(h_{1j}-\mu_{2j})C_{1j}$	Same	$\sum_{j=1}^{n}(h_{1j}-h_{2j}-\mu_{2j})C_{1j}$	Same
	$-\sum_{j=1}^{n}h_{1j}(r_j+p_{1j})-\gamma_1$	$-\sum_{j=1}^{n}h_{1j}(r_j+p_{1j})$	*	*
PPi†	$\sum_{j=1}^{n}(\mu_{ij}-\mu_{i+1j})C_{ij}$	Same	$\sum_{j=1}^{n}(h_{ij}-h_{i+1j}+\mu_{ij}-\mu_{i+1j})C_{ij}$	Same
	$-\gamma_i$	0	*	*
PPm	$\sum_{j=1}^{n}[(\epsilon_j-\mu_{mj})E_j+(\pi_j+\mu_{mj})T_j]$	Same	$\sum_{j=1}^{n}[(\epsilon_j-h_{mj}-\mu_{mj})E_j+(\pi_j+h_{mj}+\mu_{mj})T_j]$	Same
	$\sum_{j=1}^{n}\mu_{mj}d_j-\gamma_m$	$\sum_{j=1}^{n}\mu_{mj}d_j$	*	*

(†)For $i=2,\ldots,m-1$.

(*)Same as their counterparts in models with waiting time variables.

Table 1: Summary of subproblems.

Pricing Problem Objectives

The key to solving LM1 and LM2 effectively is to understand the nature of the pricing problems. Note that there is no explicit reason why the objective function coefficients of the completion time or the tardiness variables in the pricing problems must be nonnegative. Below, we give an interpretation of the pricing problem objectives and discuss the implications if there exist completion time or tardiness variables with negative objective function coefficients in the pricing problems. First, in order to simplify the discussion, we introduce the term "$\bar{t}$-boundedness":

Definition 3.4 *In a pricing problem, if at least one tardiness variable T_j or a completion time variable C_{ij} has a negative objective function coefficient, then this pricing problem is called $\bar{t}$-bounded. Otherwise, it is called bounded.*

Observe that a completion time/tardiness variable with a negative objective function coefficient in the pricing problem of machine i would imply that we could identify schedules with arbitrarily small reduced costs if the minimization were not carried out over a polytope of feasible schedules, i.e., S_i. However, $C_{ij} \leq \bar{t}_{ij}$ holds in all pricing problems, which motivates the definition above.

Now, we explain the interplay between the inventory holding costs and the dual variables for job j on machine i ($2 \leq i \leq m-1$) using (3.30)-(3.31):

$$\sum_{k=1}^{K_{i-1}} C_{i-1j}^k x_{i-1}^k - \sum_{k=1}^{K_i} C_{ij}^k x_i^k = -p_{ij} \tag{3.30}$$

$$\sum_{k=1}^{K_i} C_{ij}^k x_i^k - \sum_{k=1}^{K_{i+1}} C_{i+1j}^k x_{i+1}^k = -p_{i+1j} \tag{3.31}$$

where we assume that strict complementary slackness holds, i.e., $\mu_{ij} < 0$ and $\mu_{i+1j} < 0$. Note that (3.30)-(3.31) belong to the set of constraints (3.27) with the slack variables equal to zero. A similar analysis can be carried out for the other pricing problems as well. Assume that we increase C_{ij} by a small amount $\delta > 0$ while keeping C_{i-1j} and C_{i+1j} constant. Then, the marginal increase in cost for job j is $(h_{ij} - h_{i+1j})\delta$. In order to measure the effect induced on other jobs we use the "shadow price" interpretation of dual variables and view this change in C_{ij} as an equivalent decrease/increase in the right hand side of (3.30)/(3.31). The resulting marginal increase in the objective function is given by $\mu_{ij}(-\delta) + \mu_{i+1j}\delta = (-\mu_{ij} + \mu_{i+1j})\delta$, for sufficiently small δ. Thus, we observe that the objective function coefficient of C_{ij} in LM2-PPi, $i = 2, \ldots, m-1$, is the difference between these two effects, i.e., $(h_{ij} - h_{i+1j}) - (-\mu_{ij} + \mu_{i+1j})$, and the sign of this difference is not necessarily nonnegative.

This potential "$\bar{t}$-boundedness" in the pricing problems has very significant practical implications. Although we can solve the $\bar{t}$-bounded pricing problems as effectively as we can the bounded pricing problems (see Section 4.1), $\bar{t}$-boundedness has several negative consequences. First, we observe that when some pricing problems are $\bar{t}$-bounded, other bounded pricing problems generally do not yield negatively priced schedules, and the column generation algorithm converges very slowly. Second, $\bar{t}$-boundedness may cause the column generation algorithm to terminate prematurely with a large optimality gap as we demonstrate in our computational study. Third, the performance of the simplex algorithm suffers while solving the LP master problems, and each iteration of the column generation algorithm takes longer due to $\bar{t}$-boundedness. Note that the columns from the $\bar{t}$-bounded pricing problems that are added to the LP master problems may have coefficients that are much larger than those of other columns. Consequently, the LP master problem becomes poorly scaled, and numerical difficulties appear in the simplex algorithm.

3.2 Models With Artificial Variables

In this section, we develop conditions for boundedness of all pricing problem objectives and introduce a two-phase approach for solving LM1 and LM2 using these conditions and LP duality theory. As we demonstrate in our computational study, this approach speeds up the convergence of the column generation algorithm significantly.

Lemma 3.5 *In LM1, all pricing problems are bounded if the dual variables* μ_{ij} $(i = 2, \ldots, m, \ \forall j)$ *satisfy*

the following relationships:

$$\mu_{2j} \leq h_{1j} \qquad \forall j \tag{3.32}$$

$$-\mu_{ij} + \mu_{i+1j} \leq 0 \quad i = 2, \ldots, m-1, \ \forall j \tag{3.33}$$

$$-\mu_{mj} \leq \pi_j \qquad \forall j \tag{3.34}$$

In LM2, all pricing problems are bounded if the dual variables μ_{ij} $(i = 2, \ldots, m, \ \forall j)$ satisfy the following relationships:

$$\mu_{2j} \leq h_{1j} - h_{2j} \qquad \forall j \tag{3.35}$$

$$-\mu_{ij} + \mu_{i+1j} \leq h_{ij} - h_{i+1j} \quad i = 2, \ldots, m-1, \ \forall j \tag{3.36}$$

$$-\mu_{mj} \leq \pi_j + h_{mj} \qquad \forall j \tag{3.37}$$

Proof. These conditions follow from nonnegativity of the objective function coefficients of the completion time and tardiness variables in the pricing problems. ∎

Furthermore, by constraint (3.13) of DLM1, all dual variables of LM1 satisfy:

$$\mu_{ij} \leq h_{ij} \quad i = 2, \ldots, m, \ \forall j, \tag{3.38}$$

and from (3.27) the following holds for the dual variables of LM2:

$$\mu_{ij} \leq 0 \quad i = 2, \ldots, m, \ \forall j. \tag{3.39}$$

Together, these observations lead us to the lemma below:

Lemma 3.6 *For all optimal solutions of LM1:*

$$\mu^*_{ij} \leq \min_{1 \leq k \leq i} h_{kj} \quad i = 2, \ldots, m, \ \forall j \tag{3.40}$$

For all optimal solutions of LM2:

$$\mu^*_{ij} \leq \min(0, \ h_{1j} - h_{ij}) \quad i = 2, \ldots, m, \ \forall j \tag{3.41}$$

Proof. See Appendix.

Lemma 3.6 provides some economic insight into our problem as well. Suppose that we obtain an optimal solution to LM1 and would like to know how the objective function would change if we reduce the processing time of job j on machine i by one time unit. Clearly, we can keep the current schedule and incur h_{ij} for the additional delay between machine $i-1$ and machine i. However, this lemma tells us that there may

be better alternatives. The reduction in p_{ij} can also be viewed as an increase in the right hand side of the corresponding operation precedence constraint. Hence, the marginal change in the optimal cost is μ_{ij}^* and bounded by $\min_{1 \leq k \leq i} h_{kj}$, which implies that the slack introduced between the operations of job j on machines $i-1$ and i can be absorbed anywhere upstream from machine i.

Below, we prove a stronger version of Lemma 3.3 after defining an important economic concept for our problem. Note that each operation on a job consumes resources (e.g., materials, labor), adding value to the product. Hence, one can often assume that the inventory holding cost of a job is increasing as it proceeds through its processing stages, which motivates the following definition:

Definition 3.7 *The m-machine flow shop is said to be a value-added chain if the inventory holding costs are strictly increasing over the processing stages, i.e.,* $h_{1j} < h_{2j} < \cdots < h_{mj} < \epsilon_j, \ \forall j.$

Our algorithms do not make any specific assumption about the cost structure; however, part of our computational study will be devoted to exploring the effect of the assumption of a "value-added chain" on the performance of our algorithms because it reflects a commonly observed relationship among holding costs. Furthermore, the lemma below provides additional motivation to compare LM1 and LM2.

Lemma 3.8 *If the m-machine flow shop is a value-added chain, then in all optimal solutions of LM1,* $w_{ij}^* = 0, \ i = 2, \ldots, m, \ \forall j.$

Proof. See Appendix.

Now, we are ready to introduce our *two-phase* approach to solve LM1 and LM2 in which we impose a structure on the dual variables according to Lemmas 3.5 and 3.6. Observe that this structure is automatically satisfied by the optimal dual variables of LM1 and LM2, but not necessarily by the optimal dual variables of the restricted LP master problems RLM1 and RLM2. Our approach is based on the correspondence of *dual constraints* and *primal variables* which implies that in order to impose a constraint on the dual variables we need to introduce primal variables.

In the first phase, we add *artificial* primal variables into the LP master problems that constrain the dual variables in such a way that the pricing problems are always bounded. The intent is to generate schedules that drive the solutions of the restricted LP master problems RLM1/RLM2 towards the optimal solutions of LM1/LM2, and we stop when we cannot identify any negatively priced schedules. However, when we terminate the column generation algorithm in this phase, there is no guarantee that all artificial variables will be equal to zero because in the first phase we solve a related but a slightly different problem.

In the second phase, we drive the artificial variables to zero using complementary slackness. We *gradually* relax the constraints that we impose in the first phase by increasing the objective function coefficients of

the artificial variables. Note that during this phase it is quite possible that some of the pricing problems become $\bar{t}$-bounded. We terminate the column generation when all artificial variables are equal to zero and we cannot find any negatively priced columns. Overall, this approach enables us to avoid $\bar{t}$-bounded pricing problems initially, speeds up the convergence of the column generation, and reduces the optimality gap.

Let v_{ij} $(i = 2, \ldots, m,\ \forall j)$, y_{ij} $(i = 2, \ldots, m-1,\ \forall j)$ and z_j $(j = 1, \ldots, n)$ be the artificial primal variables associated with (3.40) in LM1 and (3.41) in LM2, (3.33) in LM1 and (3.36) in LM2, and (3.34) in LM1 and (3.37) in LM2, respectively. Then, the LP master problem for phase one of solving LM1 is presented below. Here "LHS" stands for "left hand side," and the corresponding restricted LP master problem is called RALM1.

(ALM1)

$$
\begin{aligned}
z_{ALM1} = \min_{\substack{x_i^k,\ w_{ij},\\ v_{ij},\ y_{ij},\ z_j}} \quad & (3.1) + (3.2) + (3.3) \\
& + \sum_{i=2}^{m} \sum_{j=1}^{n} \left(\min_{1 \le k \le i} h_{kj} \right) v_{ij} && (3.42) \\
& + \sum_{i=2}^{m-1} \sum_{j=1}^{n} 0 * y_{ij} && (3.43) \\
& + \sum_{j=1}^{n} \pi_j z_j && (3.44)
\end{aligned}
$$

s.t.

$$
\begin{aligned}
(\mu_{2j}) \quad & \text{LHS of } (3.4) + v_{2j} - y_{2j} = -p_{ij} && i = 2,\ \forall j && (3.45) \\
(\mu_{ij}) \quad & \text{LHS of } (3.4) + v_{ij} + y_{i-1j} - y_{ij} = -p_{ij} && 3 \le i \le m-1,\ \forall j && (3.46) \\
(\mu_{mj}) \quad & \text{LHS of } (3.4) + v_{mj} + y_{m-1j} - z_j = -p_{ij} && i = m,\ \forall j && (3.47) \\
(\gamma_i) \quad & \sum_{k=1}^{K_i} x_i^k = 1 && \forall i && (3.48) \\
& x_i^k \ge 0 && \forall i,\ k = 1, \ldots, K_i && (3.49) \\
& w_{ij} \ge 0 && i \ge 2,\ \forall j && (3.50) \\
& v_{ij} \ge 0 && i \ge 2,\ \forall j && (3.51) \\
& y_{ij} \ge 0 && 2 \le i \le m-1,\ \forall j && (3.52) \\
& z_j \ge 0 && \forall j && (3.53)
\end{aligned}
$$

Similarly, in order to obtain the LP master problem ALM2 for phase one of solving LM2, we would need to add the appropriate terms to the objective function of LM2 based on Lemmas 3.5 and 3.6, and introduce artificial variables to the left hand side of constraints (3.27) exactly as in (3.45)-(3.47). The corresponding restricted LP master problem is called RALM2.

When phase one terminates, we start phase two by increasing the objective function coefficients of positive artificial variables by a small amount δ and continue with column generation. In general, we may need to repeat this process several times until all artificial variables become zero.

The advantage of this two-phase approach derives from the fact that although LM1 differs from ALM1, and LM2 differs from ALM2, the corresponding pricing problems have *exactly the same structure*. Thus, we are able to impose a structure on the dual variables without the need for new algorithms for the pricing problems. Some stronger results can be obtained from the Lagrangian relaxations ALR1/ALR2 corresponding to ALM1/ALM2:

(ALR1(μ_{ij}))

$$z_{ALR1(\mu_{ij})} = \min_{\substack{C_{ij},\ w_{ij} \\ v_{ij},\ y_{ij},\ z_j}} \quad (3.17) + (3.18) + (3.19) + (3.20) + (3.21)$$

$$+ \sum_{i=2}^{m} \sum_{j=1}^{n} \left(\min_{1 \le k \le i} h_{kj} - \mu_{ij} \right) v_{ij} \tag{3.54}$$

$$+ \sum_{i=2}^{m-1} \sum_{j=1}^{n} (\mu_{ij} - \mu_{i+1j}) y_{ij} \tag{3.55}$$

$$+ \sum_{j=1}^{n} (\pi_j + \mu_{mj}) z_j \tag{3.56}$$

s.t.

$$(2.3),\ (2.5)\text{-}(2.8)$$

$$(3.22)$$

$$(3.51) - (3.53)$$

Observe that ALR1(μ_{ij}) and LR1(μ_{ij}) are identical, except for the terms (3.54)-(3.56) in the objective function of ALR1(μ_{ij}) that are based on Lemmas 3.5 and 3.6. ALR2(μ_{ij}) can be obtained from LR2(μ_{ij}) in a similar way. Then, the Lagrangian dual problems ALD1 and ALD2 maximize ALR1(μ_{ij}) and ALR2(μ_{ij}) over the dual variables, respectively. The Lagrangian functions with artificial variables have an additional advantage over their corresponding DW reformulations; not only are their subproblems the same as those of their counterparts with no artificial variables, but there is also no need for a "phase two" in the Lagrangian case as we prove next.

Lemma 3.9 *There exist optimal solutions to ALD1 and ALD2 such that* $v_{ij}^* = 0$ $(i = 2, \dots, m,\ \forall j)$, $y_{ij}^* = 0$ $(i = 2, \dots, m-1,\ \forall j)$ *and* $z_j^* = 0$ $(j = 1, \dots, n)$.

Proof. The proof is similar to that of Lemma 3.3. See Appendix.

Now, we are ready to present our main result in this section:

Proposition 3.10 *All linear programming master problems and Lagrangian dual problems presented in Section 3 have the same optimal objective function value, i.e.,*

$$z_{LM1} = z_{LM2} = z_{LD1} = z_{LD2} = z_{ALM1} = z_{ALM2} = z_{ALD1} = z_{ALD2} \tag{3.57}$$

Proof. Clearly, $z_{LM1} = z_{LM2}$. Then, $z_{LM1} = z_{LD1}$, $z_{LM2} = z_{LD2}$, $z_{ALM1} = z_{ALD1}$ and $z_{ALM2} = z_{ALD2}$ follow from the equivalence of corresponding DW reformulations and Lagrangian dual problems. By Lemma 3.9, we have $z_{LD1} = z_{ALD1}$ and $z_{LD2} = z_{ALD2}$ which establishes the desired result. ∎

In summary, all models we consider have the same optimal objective function value; however, their optimal solutions have different properties, which has very significant practical consequences, as we demonstrate in our computational study. Also, although the objective function coefficients may have a different structure from pricing problem to pricing problem, we always solve weighted completion time problems on machines 1 through $m-1$ and an earliness/tardiness problem on the last machine.

Finally, in the proposition below we establish the relationship between the optimal objective function values of our models and the optimal objective function value of the LP relaxation of TIFm.

Proposition 3.11 *The optimal objective function value of LM1 is greater than or equal to that of the LP relaxation of TIFm, i.e.,* $z_{LPTIFm} \leq z_{LM1}$.

Proof. See Appendix.

In other words, our models can provide tighter bounds on the optimal integer solution than the LP relaxation of TIFm. In addition, our models have shorter solution times than that of the LP relaxation of TIFm, especially for large problem instances. (See Section 5.)

3.3 Switching From Column Generation to Lagrangian Relaxation

Although we do not solve any Lagrangian dual problem explicitly, we use the Lagrangian functions to compute alternate lower bounds, both on the optimal objective value of the LP master problems and on the optimal integer objective value.

The link between the LP master problems and the Lagrangian functions is established through dual variables, i.e., we can pass the optimal dual variables of the *restricted* LP master problems to the corresponding Lagrangian functions and compute alternate lower bounds. Recall that the conditions in Lemmas 3.5 and 3.6 are not necessarily satisfied by the optimal dual variables of a restricted LP master problem. However, here Lagrangian relaxation has one advantage over column generation: in column generation, the values of the dual variables are dictated by a linear program whereas we are able to manipulate the dual variables before optimizing the Lagrangian function. Thus, in any iteration of the column generation algorithm, we can take the optimal dual vector of the *restricted* LP master problem, change the values of the dual variables according to Algorithm 1 for RLM1/RALM1 and Algorithm 2 for RLM2/RALM2 (see below), and then minimize the corresponding Lagrangian function with this new dual vector. Algorithms 1 and 2 ensure that the dual variables passed to Lagrangian relaxation satisfy the conditions in Lemmas 3.5 and 3.6. In addition,

in the computational experiments, we observed that after running these algorithms, the dual variables differ very little from their original values.

Lemma 3.12 *For any optimal dual vector $\{\mu_{ij}\}$ from RLM1/RALM1 (RLM2/RALM2), the dual variables satisfy the conditions in Lemmas 3.5 and 3.6 when Algorithm 1 (Algorithm 2) terminates.*

Proof. See Appendix.

Algorithm 1 Adjusting Dual Variables For RLM1/RALM1

Require: An optimal dual vector $\{\mu_{ij}\}$ for the restricted LP master problem RLM1 or RALM1 is given.

$\mu_{ij} = \max(-\pi_j,\ \mu_{ij}),\ i = 2, \ldots, m,\ \forall j$

$\mu_{ij} = \min(h_{ij},\ \mu_{ij}),\ i = 2, \ldots, m,\ \forall j$

$\mu_{2j} = \min(h_{1j},\ \mu_{2j}),\ \forall j$

for $j = 1$ **to** n **do**

  for $i = 2$ **to** $m - 1$ **do**

    $\mu_{i+1j} = \min(\mu_{ij},\ \mu_{i+1j})$

Algorithm 2 Adjusting Dual Variables For RLM2/RALM2

Require: An optimal dual vector $\{\mu_{ij}\}$ for the restricted LP master problem RLM2 or RALM2 is given.

$\mu_{ij} = \max(-\pi_j - h_{ij},\ \mu_{ij}),\ i = 2, \ldots, m,\ \forall j$

$\mu_{2j} = \min(0,\ h_{1j} - h_{2j},\ \mu_{2j}),\ \forall j$

for $j = 1$ **to** n **do**

  for $i = 2$ **to** $m - 1$ **do**

    $\mu_{i+1j} = \min(0,\ \mu_{ij} + h_{ij} - h_{i+1j},\ \mu_{i+1j})$

3.4 Lower Bounds and Feasible Integer Solutions

Lower Bounds

A well known drawback of column generation algorithms is that although a good solution is usually obtained quickly it may take a very long time to prove LP optimality. See Barnhart et al. (1998) for a discussion of this so-called "tailing-off effect."

This problem is particularly relevant to our case because we are not even guaranteed to find the LP optimal solution, as some of our pricing problems are strongly $\mathcal{NP}$-hard, and all pricing problems are solved heuristically. Therefore, in order to terminate the column generation procedure when a good, but not necessarily optimal solution is found, we need a tight lower bound on the optimal objective value of the LP master problem. Of course, such a lower bound can also be used to quantify the optimality gap of the

feasible *integer* solutions we construct. Bounds based on the optimal objective value of the current restricted LP master problem and the objective values of the pricing problems appear in Lasdon (1970), Farley (1990) and Vanderbeck and Wolsey (1996). We follow a similar approach.

From LP duality theory, we know that the objective value of any dual feasible solution provides a lower bound on the optimal objective value of the primal problem, and that the reduced cost of a variable is the infeasibility in the corresponding dual constraint. These two facts together imply that we can construct a dual feasible solution using the *optimal* objective values of the pricing problems (reduced costs), and thus compute a lower bound on the optimal objective value of the LP master problem. Below, we develop a lower bound on the optimal objective value of LM1, and analogous lower bounds can be computed for the other LP master problems. In the following discussion, a bar under an optimal objective value denotes a corresponding lower bound. Let $\{\mu_{ij},\ \gamma_i\}$ represent the optimal dual variables of RLM1 in the current iteration. Then, $\{\mu_{ij},\ \gamma_i'\}$ is a feasible solution for DLM1, where

$$\gamma_i' = \gamma_i + z_{LM1}^{PPi},\ \forall i. \tag{3.58}$$

To see why this holds, observe that adding the *optimal* objective values from the pricing problems to the dual variables associated with the convexity constraints decreases the left hand sides of constraints (3.10)-(3.12) in DLM1, *exactly* by the amount of infeasibility. Thus, $\{\mu_{ij},\ \gamma_i'\}$ is feasible for DLM1, and

$$\underline{z}_{LM1} = \sum_{i=2}^{m}\sum_{j=1}^{n} -p_{ij}\mu_{ij} + \sum_{i=1}^{m}\gamma_i' \tag{3.59}$$

is a lower bound on z_{LM1} and the optimal integer solution. The only modification necessary for our problem is to replace z_{LM1}^{PPi} by $\underline{z}_{LM1}^{PPi}$ in (3.58) because we solve the pricing problems approximately. In order to compute the lower bounds $\underline{z}_{LM1}^{PPi}\ \forall i$, we use the algorithms developed by Bülbül et al. (2001) who provide a tight lower bound for the single-machine earliness/tardiness scheduling problem.

A similar lower bound can be obtained from Lagrangian relaxation as well. Note that $z_{LR1(\mu_{ij})} = \sum_{i=1}^{m} z_{LM1}^{PPi} + \sum_{i=1}^{m}\gamma_i - \sum_{i=2}^{m}\sum_{j=1}^{n} p_{ij}\mu_{ij}$ where the last two terms are constant for a given set of dual variables. Thus, similar to above,

$$\underline{z}_{LR1(\mu_{ij})} = \sum_{i=1}^{m} \underline{z}_{LM1}^{PPi} + \sum_{i=1}^{m}\gamma_i - \sum_{i=2}^{m}\sum_{j=1}^{n} p_{ij}\mu_{ij} \tag{3.60}$$

is a lower bound on z_{LM1} and z_{IM1}. Together with the algorithms in Section 3.3 to pass dual vectors from column generation to Lagrangian relaxation, (3.60) provides a very useful alternate lower bound for our problem.

Feasible Integer Solutions

In order to construct feasible solutions for the original integer programming problem Fm, we need a job processing sequence on each machine. Then, *given* these sequences we can solve TTFm (see Section 2), and obtain a feasible integer solution for the original problem. In general, TTFm is a small LP that is solved optimally very quickly. Hence, the major concern here is to find good job processing sequences, which we obtain in two different ways.

First, for all jobs and all machines, we compute the completion time of job j on machine i in the optimal solution of the current restricted LP master problem as:

$$C'_{ij} = \sum_{k=1}^{K'_i} C_{ij}^k x_i^k, \tag{3.61}$$

assuming that there are K'_i columns for machine i in the current restricted LP master problem. Then, we generate job processing *sequences* on each machine by sequencing jobs in non-decreasing order of these completion times. Finally, we find the optimal *schedule* given these sequences by solving TTFm which yields a feasible solution for Fm.

Second, for each machine i, we identify the schedule S_i^k with the largest x_i^k value in the current solution, and use the sequence of that schedule in solving TTFm. Similarly, we obtain job processing sequences from the heuristic solutions of Lagrangian subproblems, and construct feasible integer solutions.

4 Column Generation

In this section, we briefly discuss some details of the column generation algorithm and then summarize our approach to solve the single-machine subproblems.

Creating An Initial Feasible Restricted LP Master Problem

Creating a good feasible initial restricted LP master problem is important to ensure that proper dual information is passed to the pricing problems. In order to start the column generation procedure with a good set of columns, we initially assign values to the dual variables such that the conditions in Lemmas 3.5 and 3.6 are satisfied (see below). Then, we solve the pricing problems with these dual variables, obtain job processing sequences from their solutions, and timetable these sequences with TTFm to obtain a feasible integer solution. Clearly, adding the individual machine schedules from this feasible integer solution to the restricted LP master problem guarantees a feasible LP solution. The dual variables are initialized as:

$$\mu_{ij} = \min_{1 \le k \le i} h_{kj} \qquad i = 2, \ldots, m, \ \forall j. \text{ (For RLM1/RALM1)} \tag{4.1}$$

$$\mu_{ij} = \min_{1 \le k \le i} h_{kj} - h_{ij} \quad i = 2, \ldots, m, \ \forall j. \text{ (For RLM2/RALM2)} \tag{4.2}$$

Lemma 4.1 *The dual variables in* (4.1) *and* (4.2) *satisfy the conditions in Lemmas 3.5 and 3.6.*

Proof. See Appendix.

Adding Columns

Our heuristics to solve the single-machine subproblems may identify several different negatively priced columns in a single iteration. (See Section 4.1.) However, our preliminary computational results indicated that although this approach may allow the column generation procedure to terminate in fewer iterations, the total CPU time is generally much longer because the size of the restricted LP master problem grows very rapidly and each iteration consumes substantially more time. Therefore, in each iteration we add at most one column from each subproblem.

In Lemma 3.3, we established that there exists an optimal solution to LM1 such that all waiting times are equal to zero. Motivated by this lemma, whenever we identify a negatively priced schedule on machine i to be added to RLM1/RALM1, we generate schedules on *all* machines using Algorithm 3 such that no job waits between its operations, and add these schedules to the restricted LP master problem as well. These no-wait schedules do not necessarily have negative reduced costs, and we may delete them later. However, our initial results indicated that this approach speeds up the convergence and decreases the total CPU time of the column generation procedure for solving LM1/ALM1 considerably despite the rapid growth of the restricted LP master problems.

Algorithm 3 Generating No-Wait Schedules For RLM1/RALM1

Require: A negatively priced schedule S_i^k on machine i is given.

for $q = i$ **to** $m - 1$ **do**
  $C^k_{q+1j} = C^k_{qj} + p_{q+1j}, \ \forall j$ {No job waits between its operations}
for $q = i$ **downto** 2 **do**
  $C^k_{q-1j} = C^k_{qj} - p_{qj}, \ \forall j$ {No job waits between its operations}
for $j = 1$ **to** $n - 1$ **do** {Assume jobs are in sequence order}
  for $i = 1$ **to** m **do**
    if $C^k_{ij} > C^k_{ij+1} - p_{ij+1}$ **then** {Machine capacity is violated}
      Shift all operations of job $j + 1$ forward by $C^k_{ij} - C^k_{ij+1} + p_{ij+1}$
Add schedules $S_1^k, \ldots, S_m^k$ to RLM1/RALM1.

In order to speed up the column generation procedure further, we apply a local improvement algorithm to existing columns in order to identify new schedules with negative reduced costs. (See Algorithm 4 below.) One major advantage of the local improvement algorithm is that it is much faster than solving the pricing problems and potentially decreases the total number of iterations. For each machine i, a good starting schedule for the local improvement algorithm is the column with the largest x_i^k value in the current solution. If we can improve on such a schedule, we may intuitively expect that it would enter the solution with a larger x_i^k value, increasing the quality of our feasible integer solutions. Note that large x_i^k values imply that the optimal solution of the current restricted LP master problem is close to integral (see (3.61)), and we may expect that the feasible integer solutions obtained from the optimal solution of the current restricted LP master problem have small optimality gaps. (See Section 3.4.) Our computational study demonstrates that the local improvement scheme is extremely useful, in accordance with the observations of Savelsbergh and Sol (1998) for a different problem.

Algorithm 4 Local Improvement On Existing Columns

Require: An initial schedule S_i^k on machine i.

1: **for** $j = 1$ **to** $n - 1$ **do** {Assume jobs are in sequence order}
2: Reverse the sequence of jobs j and $j + 1$.
3: Schedule $j + 1$ and j optimally in the interval $[C_{ij}^k - p_{ij},\ C_{ij+1}^k]$.
4: **if** the sequence $j + 1$, j does not decrease the reduced cost of S_i^k **then**
5: Recover the original schedule before the interchange
6: STOP if no improvement during an entire pass through S_i^k, otherwise go back to Step 1.

Removing Columns

Column deletion is an essential part of our column management strategy. We observed that the columns that are added at the initial stages of the column generation procedure are usually not present in the final solution. This is particularly true for LM1/LM2, and if we never remove any columns, then solving the restricted LP master problem becomes eventually very time consuming. Therefore, we delete columns by using a rule similar to that in Van den Akker et al. (2002). Let $\underline{z}_{Fm}$ and $\overline{z}_{Fm}$ represent the best lower bound and the best integer solution available for the original scheduling problem, respectively. Then, we delete columns with a reduced cost strictly larger than $\overline{z}_{Fm} - \underline{z}_{Fm}$ because they cannot appear in an optimal *integer* solution. Clearly, these columns may be necessary for solving the LP master problems, in which case they will be generated again.

Here, we need to re-emphasize the significance of Lagrangian relaxation in obtaining an alternate lower bound. It potentially improves the best lower bound available, and thus increases the number of columns deleted and speeds up the column generation algorithm. In fact, we always perform one Lagrangian iteration before any column deletion step.

Termination Conditions

For large problems, generating columns until we cannot identify any negatively priced column may take a very long time, with no significant improvement in the objective function value of the restricted LP master problem or the feasible integer solution. Therefore, we compute various quantities and terminate column generation when one of them is sufficiently small.

Ideally, we would like to stop when the *integer* optimality gap $(\overline{z}_{Fm} - \underline{z}_{Fm})/\underline{z}_{Fm}$ is small enough, but unfortunately, this gap is usually quite large and does not provide useful information. As an alternative, we terminate column generation when $(z_{RLM} - \underline{z}_{LM})/\underline{z}_{LM}$ or $(z_{RLM} - z_{LR(\mu_{ij})})/z_{LR(\mu_{ij})}$ is sufficiently small, where z_{RLM} is the optimal objective function value of the current restricted LP master problem, and $z_{LR(\mu_{ij})}$ is the objective function value of the (not necessarily optimal) solution of the corresponding Lagrangian relaxation LR(μ_{ij}). The latter ratio does not have any theoretical meaning; however, it is a useful stopping condition for hard problems in which the other gaps are not within specified limits. Note that the optimal objective of LR(μ_{ij}) is a lower bound on the optimal objective of the LP master problem, and the gap between z_{RLM} and $z_{LR(\mu_{ij})}$ still provides a good stopping condition when the others fail. Occasionally, we continue generating columns with no decrease in the objective function value of the restricted LP master problem, indicating degeneracy in the simplex algorithm. In those cases, we stop if the objective does not improve in n consecutive iterations.

4.1 Subproblems

We use the heuristics developed by Bülbül et al. (2001) for the single-machine earliness/tardiness scheduling problem to solve all subproblems. Note that a weighted completion time problem is a special case of an earliness/tardiness problem with all due dates set equal to zero, and the tardiness costs set equal to the completion time costs. Thus, we are able to solve all subproblems within a unified framework. Below, we explain the fundamental ideas behind our single-machine heuristics and then discuss briefly the modifications that arise from negative earliness or tardiness costs in the subproblem objectives.

To solve the single-machine earliness/tardiness (E/T) problem P1, we follow a two-phase approach, where in the first phase we obtain a good job processing sequence, and in the second phase we schedule jobs optimally given the job sequence. Our approach incorporates computing a fast and tight **preemptive** lower bound for P1, and using information from the lower bound solution to construct near-optimal feasible schedules for P1. To compute the preemptive lower bound, we divide the jobs into unit time intervals and construct a transportation problem TR1 by assigning a cost to each of these unit jobs and considering an appropriate planning horizon which depends on the ready times, due dates and the sum of the processing times. In general, TR1 corresponding to a weighted completion time problem is solved more quickly than TR1 corresponding to an E/T problem because weighted completion time problems have shorter planning

horizons than E/T problems. The cost structure of the unit jobs is fundamental to the success of our algorithms, and when all earliness and tardiness costs are *positive*, all unit jobs of a job are within a single *block* in the optimal solution of TR1, where a block is a sequence of unit jobs processed without idle time. (See Kanet and Sridharan (2000) for a discussion of the well known block structure in the E/T problems.) We observe that in these blocks, less expensive jobs are split more, and more expensive jobs are scheduled almost contiguously and close to their positions in the optimal non-preemptive schedule. Based on this observation, we construct various job processing sequences that yield excellent non-preemptive feasible schedules for P1.

From Section 3.1, we know that depending on the relative values of inventory holding and tardiness costs and dual variables, some jobs may have negative earliness or tardiness costs in the subproblem objectives. In this case, we need two types of modifications to our single-machine algorithms. First, the planning horizon needs to be expanded to account for negative costs if we want to compute a lower bound from TR1. However, in our column generation algorithm, we compute the planning horizon only once at the beginning under the assumption of positive costs because we do not want to lose speed by increasing the size of the subproblems. Thus, we give up our ability to compute lower bounds from column generation in favor of solution speed, if there are negative costs. Note that we can still obtain lower bounds from Lagrangian relaxation, if there are no negative earliness costs, by using the approach in Section 3.3 that removes all negative completion time and tardiness costs in Lagrangian subproblem objectives. This is particularly important when solving LM1 and LM2, in which the pricing problems are $\bar{t}$-bounded until the solution gets close to optimality.

Second, we need to ensure that jobs whose tardiness costs are negative appear at the end of the job processing sequence for single-machine timetabling. To achieve this, we solve TR1 with a planning horizon determined under the assumption of positive earliness and tardiness costs, generate job processing sequences, and then transfer jobs with negative tardiness costs to the end of the sequence. Afterwards, when we solve the single-machine timetabling LP for machine i, which is obtained from TTFm by setting $m = 1$, the simplex algorithm identifies a feasible schedule S_i^{k-1} and an *unboundedness* ray λ_i if there are negative tardiness costs. Thus, the completion times in the final single-machine schedule S_i^k are computed as $C_{ij}^k = C_{ij}^{k-1} + \alpha\lambda_{ij}$ where λ_{ij} is the j^{th} component of λ_i and $\alpha = \min_{\pi_j<0} \frac{\bar{t}_{ij} - C_{ij}^{k-1}}{\lambda_{ij}}$.

Finally, we need to mention that if there are $\bar{t}$-bounded subproblems, we *only* solve these for a few iterations of the column generation algorithm and skip the bounded subproblems. This enhancement is based on the observation that if there exist $\bar{t}$-bounded subproblems, usually no negatively priced columns are identified by solving the bounded subproblems.

5 Computational Experiments

We performed a variety of computational experiments in order to evaluate the performance of both the lower bounds and the feasible integer solutions. The experimental design has two main objectives. First, we would like to demonstrate that our algorithms find good solutions and find them rapidly. There are two important factors that affect the speed of our algorithms: the size of the restricted LP master problem, which depends on the number of columns added, and the sizes of the subproblems, where the sum of the processing times on machine i determines the size of the subproblem corresponding to machine i.

Our other major concern is the robustness of our algorithms. In particular, we are interested in the performance of our algorithms as the number of machines, the number of jobs and the length of the processing times increase. The importance of the latter derives from the fact that we use a preemptive relaxation when solving the single-machine subproblems (see Section 4.1), and the number of preemptions increases as the processing times increase. The cost structure is also potentially important because $\bar{t}$-boundedness is related to the relative magnitudes of the costs and the dual variables.

Finally, we compare the performance of our algorithms to that of the LP relaxation of TIFm. Recall that in Proposition 3.11 we proved that z_{LM1} is a tighter bound on the optimal integer solution than is z_{LPTIFm}. Here, we demonstrate that our column generation procedure is also faster than solving the LP relaxation of TIFm, especially for large problem instances. Additionally, it provides a good starting point for constructing near-optimal feasible integer solutions while there is no obvious way of converting the solution of the LP relaxation of TIFm into a feasible integer solution.

5.1 Data Generation

On each machine i, the processing times are generated from a discrete uniform distribution $U[\underline{p}_i,\ \bar{p}_i]$. Then, the ready times are generated from a discrete uniform distribution $U[0,\ \lfloor 0.5P_1 \rfloor]$, where $P_1 = \sum_{j=1}^{n} p_{1j}$.

In scheduling problems with due dates, it is widely observed that the tightness and range of the due dates play an important role in determining the level of difficulty. (See Potts and van Wassenhove (1982) and Hall and Posner (2001).) Therefore, in setting the due dates, we follow a scheme similar to those of Pinedo and Singer (1999) and Eilon and Chowdhury (1976), where we change the due date tightness systematically. The due date of job j is set equal to $\lfloor r_j + f_j^d \sum_{i=1}^{m} p_{ij} \rfloor$, where f_j^d is the *due date slack factor* of job j and is generated from a continuous uniform distribution $U[TF - \frac{RDD}{2},\ TF + \frac{RDD}{2}]$. TF is the *mean* due date slack factor, and RDD is the *range* of the due date slack factors.

The inventory holding costs $\{h_{1j}\}$ are generated from a uniform distribution $U[1,\ 50]$. Then, $h_{ij} = f_{ij}^c h_{i-1j}$ $(i = 2, \ldots, m)$, $\epsilon_j = f_{m+1}^c h_{mj}$ and $\pi_j = f_{m+2}^c \epsilon_j$, where f_{ij}^c is a percentage generated from a

continuous uniform distribution $U[\underline{f}_{ij}^c,\ \overline{f}_{ij}^c]$. This method enables us to model different cost structures easily.

5.2 Summary of Results

Our algorithms were implemented in C using the `CPLEX 7.0` callable libraries. For benchmarking, TIFm and/or its linear programming relaxation were solved via `ILOG AMPL/CPLEX 7.0`. All computations were performed on a Sun UltraSPARC-IIi 440MHz computer with 256 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 of our algorithms. Specifically, we examine the effects of the number of machines and jobs, the length of the processing times and the cost structure.

5.2.1 Number of Machines and Jobs

In order to test how increasing the number of machines and/or the number of jobs affects the CPU time and the solution quality of our algorithms, we design experiments with the parameters given in Table 2. The due date slack factor, TF, has three different levels for each m, corresponding to tight, moderately tight and loose due dates. In order to maintain approximately the same percentage of tardy jobs across different m for a given TF level, we need to increase the numerical value of TF as m increases due to increasing queueing effects when there are more machines. For each combination of parameters, 5 instances are generated, resulting in a total of 15 problems for each combination of m and n. When possible, the problems are solved optimally using the time-indexed formulation TIFm. However, for instances in which `AMPL/CPLEX` identifies at least one feasible integer solution but cannot verify optimality within a maximum CPU time of half an hour, we report the best integer solution obtained. The CPU time of the branch-and-bound algorithm for solving TIFm optimally is limited to half an hour because our goal here is to demonstrate that our algorithms consistently yield high quality feasible schedules to the original problem in a short period of time compared to an alternate method. As explained below, for large problem instances, it is often difficult to even find a single feasible solution for TIFm in half an hour of CPU time.

m	n	$p_{ij},\ \forall i$	TF	RDD	$f_{ij}^c/100,\ i=2,\ldots,m+1$	$f_{m+2j}^c/100$
2	{10, 20, 30}	U[1, 10]	{1.5, 2.5, 3.5}	0.4	U[1.0, 1.5]	U[1.5, 2.0]
3	{10, 20, 30}	U[1, 10]	{2.0, 3.0, 4.0}	0.4	U[1.0, 1.5]	U[1.5, 2.0]
4	{10, 20, 30}	U[1, 10]	{2.5, 3.5, 4.5}	0.4	U[1.0, 1.5]	U[1.5, 2.0]

Table 2: Effect of the Number of Machines and Jobs: Problem Parameters.

In Tables 5 and 6, we report the performance of our lower bounds and heuristics as the number of

machines and jobs increase, both in terms of solution quality and solution time. The columns #OP ("Number Optimal") and #BI ("Number Best Integer") indicate the number of times AMPL/CPLEX terminated with the optimal integer and a feasible integer solution without verifying optimality within the time limit, respectively. For example, for $m = 4$ and $n = 10$, AMPL/CPLEX found the optimal integer solution for TIFm in 1 instance, terminated with a feasible integer solution in 7 instances, and could not identify any integer solution for the remaining 7 instances in half an hour of CPU time. Our preliminary computational tests for $m \geq 3$ and $n \geq 20$ indicated that AMPL/CPLEX could generally not find any feasible integer solution in half an hour of CPU time. Therefore, we do not report any integer results from AMPL/CPLEX for problems with $n \geq 20$, except for $m = 2$. This is indicated by N/A ("Not Available") in columns #OP and #BI. For any LP master problem LM, the columns LB ("Lower Bound") and MA-LP ("Master-Linear Programming") indicate the optimality gap of the restricted LP master problem, when column generation terminates, with respect to our lower bound (LB) and the lower bound from the LP relaxation of TIFm (MA-LP), respectively. Specifically, $LB = \frac{z_{RLM} - z_{LM}}{z_{LM}} * 100\%$, and $MA - LP = \frac{z_{RLM} - z_{LPTIFm}}{z_{LPTIFm}} * 100\%$. In order to quantify the quality of our feasible integer solutions, we report $\frac{\overline{z}_{Fm} - z_{LPTIFm}}{z_{LPTIFm}} * 100\%$ in the column IF-LP ("Integer Feasible-Linear Programming") which is an upper bound on the optimality gap of our feasible integer solutions. In addition, the column IF-AP ("Integer Feasible-AMPL") compares our feasible integer solutions to the optimal or best integer solutions obtained from AMPL/CPLEX. A minus sign in column IF-AP indicates that our integer solutions are superior to those obtained from AMPL/CPLEX. Under the CPU times, the column MA ("Master") is the total CPU time of the column generation algorithm in seconds, and MA/LP ("Master/Linear Programming") is the ratio of the solution time of our algorithm to that of the LP relaxation of TIFm. In each cell, the first number is the performance measure when no local improvement in column generation is invoked, and the number in parentheses is the corresponding measure when we perform local improvement on existing columns to identify negatively priced columns. For each value of n, the first row indicates the average and the second row indicates the worst case performance.

From Tables 5 and 6, it is clear that generating columns by local improvement makes a substantial difference, both in terms of percentage gaps and solution time. In particular, note that LM1 performs poorly in some instances, e.g., for $m = 2$ and $n = 30$, if no local improvement is carried out in column generation.

The average and worst case optimality gaps of our lower bound decrease as the number of machines increases. For a given m, these gaps may increase or decrease as the number of jobs increases; however, they are relatively close to each other. Overall, as the problem size increases, the quality of our lower bound improves. Note that for ALM2, $m = 4$, and $n = 30$, the average and worst case optimality gaps of our lower bound are 4.57% and 9.99%, respectively, when local improvement is invoked. We note that LM1 has the worst lower bound performance; the other three are similar.

Similar qualitative observations hold for the optimality gap of the LP master problems. The gap MA-LP decreases as the number of machines increases, but the number of jobs does not have an important impact

on the optimality gap of the LP master problems for a given m. We note that although ALM1, LM2 and ALM2 yield similar results, the performance of LM1 is poor when we do not perform local improvement. In this case, local improvement or the introduction of artificial variables (ALM1) has a tremendous positive effect. For ALM1, LM2, and ALM2, the average gaps are within 2% and the worst case gaps are within 5%, with or without local improvement, when $m = 4$ and $n = 30$. Note that by Proposition 3.11, these are upper bounds on the gaps, and we re-emphasize that approximate solutions for the subproblems have a significant speed advantage, and compromise very little in solution quality.

For very small problems, `AMPL/CPLEX` outperforms our algorithms in obtaining feasible integer solutions. Note that for the 14 instances solved to optimality by `AMPL/CPLEX` where $m = 2$ and $n = 10$, the average gap between the objective of the feasible integer solutions obtained by LM1/LM2 (with local improvement) and that from `AMPL/CPLEX` (IF-AP) is approximately 6-7%. For $m \geq 3$, the feasible integer solutions identified by our algorithms are significantly superior to those of `AMPL/CPLEX`. For 8 instances with $m = 4$ and $n = 10$ for which `AMPL/CPLEX` was able to return an integer solution after half an hour of CPU time, LM1 and LM2 find feasible integer solutions within 1 second that are on average 43% better. We also note that the optimality gaps of our feasible integer solutions with respect to the LP relaxation of TIFm (IF-LP) increase as the problem size grows. This may be because the LP relaxation of TIFm becomes looser as the problem size increases.

There is a trade-off between solution speed and quality of the feasible integer solutions. Comparing LM1 with ALM1 and LM2 with ALM2 shows that the solution speed gained from the use of artificial variables comes at the expense of the quality of feasible integer solutions. When solving ALM1 and ALM2, many fewer columns are added, which implies that there is less information available to construct job processing sequences for feasible integer solutions, and thus the sequences are more likely to contain errors. In Section 3.1, we argue that LM1 may not be a good starting point for constructing feasible integer solutions because of Lemma 3.3; however, the results indicate the contrary. ALM2 has in general the shortest CPU times, and the worst integer solutions.

Even if we were only interested in finding good lower bounds to the original flow shop scheduling problem Fm, our algorithms, with their short CPU times and small optimality gaps, would be superior to solving the LP relaxation of TIFm. Note that ALM2 with local improvement is almost 5 times faster than the LP relaxation of TIFm when $m = 4$ and provides solutions that are on average within 1.16% of optimality across 45 instances. In general, our strategy to avoid $\bar{t}$-boundedness initially and then drive artificial variables to zero serves very well in obtaining excellent LP solutions. It provides stability, i.e., smaller worst case bounds, and speed.

Finally, although we do not report detailed results here, we observe that the due date tightness has an important impact on problem difficulty. Under tight due dates, the gaps LB, MA-LP and the solution

times increase, however the quality of the feasible integer solutions appears not to be sensitive to due date tightness.

5.2.2 Processing Times

We also investigate whether our algorithms are sensitive to increases in processing times, which may imply a potential increase in the number of preemptions when solving the subproblems, and thus adversely affect the solution quality. The problem parameters for our experiments are given in Table 3. For each combination of parameters, 5 instances are generated, resulting in a total of 15 problems for each combination of m and n. Note that the problems with $p_{ij} \sim U[1,\ 10]$ are the same problems as in the previous section. In this section, we do not report the column IF-AP because `AMPL/CPLEX` is unable to find feasible integer solutions when the processing times increase. In all problems reported in Table 7, local improvement is invoked during column generation.

m	n	$p_{ij},\ \forall i$	TF	RDD	$f^c_{ij}/100,\ i = 2, \ldots, m+1$	$f^c_{m+2j}/100$
2	{10, 20}	U[1, 10], U[1, 20], U[1, 30]	{1.5, 2.5, 3.5}	0.4	U[1.0, 1.5]	U[1.5, 2.0]
3	{10, 20}	U[1, 10], U[1, 20], U[1, 30]	{2.0, 3.0, 4.0}	0.4	U[1.0, 1.5]	U[1.5, 2.0]

Table 3: Effect of Processing Times: Problem Parameters.

The robustness of our approach is evident from the results in Table 7. As the processing times increase, there is sometimes a slight degradation in the performance measures LB and MA-LP, although this effect is less pronounced than the effect of an increase in the number of machines and jobs. The combined effect of increasing the number of machines and jobs and the length of the processing times is a decrease in the percentage gaps of LB and MA-LP. 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. For all models, when $m = 3$, $n = 20$, and $p_{ij} \sim U[1,\ 30]$, the average gap of our lower bound (LB) ranges from 8.7 to 10.91% and the average gap of our LP master problems (MA-LP) ranges from 2.40 to 2.56%, respectively. The effect of the processing times on the optimality gap of the feasible integer solutions is not consistent; in some cases it is positive, in others, negative. However, this effect is clearly not as significant as the effect of the number of machines and jobs.

The size of our subproblems depends on the sum of the processing times(Bülbül et al. (2001)). Therefore, we expect the solution time for the subproblems to increase with the processing times, and Table 7 verifies this trend. However, we observe that the CPU time of our algorithms is much less sensitive to the planning horizon than the LP relaxation of TIFm. When $m = 3$, $n = 20$, and $p_{ij} \sim U[1,\ 30]$, our algorithms are 5.5-12.5 times faster than the LP relaxation of TIFm.

5.2.3 Cost Structure

In order to explore the sensitivity of our algorithms to the inventory holding cost structure, we compare the performance of our algorithms for problems with non-decreasing inventory holding costs in Section 5.2.1 to the performance when $f_{ij}^c/100 \sim U[0.8,\ 1.2]$ $(i = 2, \ldots, m+1)$, i.e., the inventory holding costs are similar. All other parameters are the same as in Section 5.2.1.

We do not report detailed results here, but make the following observations: whether the inventory holding costs satisfy the value-added assumption or are similar to each other, we obtain good lower bounds and high quality feasible integer solutions. We note that we have slightly better feasible integer solutions when costs are similar, and attribute this to the fact that "errors" in the job processing sequences are potentially more costly when inventory costs are non-decreasing. Also, despite Lemma 3.8, the integer solutions obtained from LM1 are in general better than those from LM2 for problems with non-decreasing inventory holding costs. The solution times are insensitive to the cost structure as well, except for the largest problems. For those instances ($m = 3$, $n = 30$), problems with similar inventory holding costs appear to be easier to solve. We conclude that in general, the cost structure does not affect the performance of our algorithms significantly.

5.2.4 Permutation Sequences

Finally, we report some observations regarding the structure of the job processing sequences that can be of great value for future research. Recall that we do not require a permutation sequence, i.e., the sequence of jobs may differ from one machine to another, and our algorithms yield high quality feasible integer solutions. Therefore, if we can show that the job processing sequences we generate do not deviate significantly from a permutation sequence, there is motivation to devise algorithms that construct permutation sequences exclusively in search for a good solution. This would reduce the number of possible job processing sequences to consider from $O((n!)^m)$ to $O(n!)$, possibly allowing us to design more effective algorithms. Now, let $q_j = \frac{1}{m}\sum_{i=1}^{m} |\overline{s}_j - s_{ij}|$ represent the *permutation index* of job j, where s_{ij} is the sequence number of job j in the job processing sequence of machine i, and $\overline{s}_j$ is the average sequence number of job j across all machine sequences. This is therefore a measure of deviation from a permutation sequence for job j. Then, we define the *permutation index* of our problem to be: $q = \frac{1}{n}\sum_{j=1}^{n} q_j$. We would expect q to be equal to zero if we have a permutation sequence, hence the lower is q, the closer are our job processing sequences to a permutation sequence. The average and the standard deviation of the permutation indices for a subset of problems from the previous section are given in Table 4. Each statistic is computed across 15 instances.

From Table 4, we note that when the inventory holding costs are non-decreasing over the processing stages, our job processing sequences are very close to permutation sequences. However, when inventory holding costs are similar, the job processing sequences deviate more from a permutation sequence because the trade-offs among operations of different jobs are less significant. The results in Table 4 indicate that

there may be advantages to solution methods that focus on permutation sequences.

	$m = 2,\ n = 10$		$m = 3,\ n = 10$	
	Non-decreasing Costs	**Similar Costs**	**Non-decreasing Costs**	**Similar Costs**
Average	0.16	0.21	0.08	0.56
Standard Deviation	0.21	0.23	0.09	0.34

Table 4: Permutation Indices.

6 Concluding Remarks

We developed a method to solve the m-machine flow shop scheduling problem with intermediate inventory holding, earliness and tardiness costs. This model is one of the first to incorporate work-in-process inventory holding costs in scheduling, and was partly motivated by recent examples from the biotechnology industry where work-in-process is expensive and needs special treatment while waiting for processing at the next stage.

We use mathematical programming methods to solve our problem. We reformulate the problem with an exponential number of variables, solve its LP relaxation approximately by column generation, and develop simple and effective ways of obtaining feasible solutions to the original problem. We exploit the duality between Dantzig-Wolfe reformulation and Lagrangian relaxation extensively to develop structural properties and tune our models accordingly. To the best of our knowledge, this paper is the first attempt to apply column generation to a machine scheduling problem in which some of the subproblems are strongly $\mathcal{NP}$-hard. Our computational study demonstrates that heuristic solutions for the subproblems provide a significant speed advantage over alternate methods and do not compromise solution quality.

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. We also note that we cannot accommodate precedence constraints among jobs on the same machine as it would destroy the structure of the subproblems. Nevertheless, we are encouraged by the performance of our algorithms, and as a more general setting, we hope to develop approaches for the job shop scheduling problem with intermediate inventory holding, earliness and tardiness costs.

Acknowledgment

We would like to thank Professor Alper Atamtürk for his helpful insights. We also would like to thank the associate editor and the anonymous referees for their valuable comments. This research was partially supported by NSF Grant DMI-0092854.

		AMPL/CPLEX		Percentage Gap (†)								CPU Time (sec.)(†)			
m	**n**	**#OP**	**#BI**	**LM1**				**ALM1**				**LM1**		**ALM1**	
				LB	**MA-LP**	**IF-LP**	**IF-AP**	**LB**	**MA-LP**	**IF-LP**	**IF-AP**	**MA**	**MA/LP**	**MA**	**MA/LP**
2	10	14	1	**21.15(17.27)**	**3.89(2.62)**	**26.36(18.93)**	**12.05(5.77)**	**18.14(20.04)**	**3.55(2.52)**	**27.61(24.59)**	**12.89(10.58)**	**0.37(0.25)**	**1.19(0.80)**	**0.24(0.23)**	**0.76(0.71)**
				107.82(57.68)	26.49(11.56)	68.82(55.04)	33.59(14.91)	61.44(74.82)	16.60(14.56)	87.81(71.60)	33.36(48.04)	0.60(0.39)	2.00(1.39)	0.41(0.68)	1.14(1.36)
	20	0	9(**)	**18.05(16.09)**	**5.46(3.31)**	**40.91(37.27)**	**-47.73(-47.53)**	**15.78(13.71)**	**4.00(3.15)**	**49.87(41.55)**	**-46.44(-47.40)**	**3.67(2.25)**	**1.40(0.87)**	**2.31(1.29)**	**0.89(0.49)**
				52.68(45.38)	11.14(7.27)	71.75(86.35)	-0.88(6.40)	51.14(38.64)	10.53(8.31)	127.99(116.95)	-1.19(-1.19)	5.49(3.33)	2.30(1.39)	4.41(2.21)	1.68(0.97)
	30	N/A	N/A	**1302.25(11.99)(*)**	**44.74(2.93)**	**40.57(30.50)**	**N/A**	**13.34(12.32)**	**4.26(3.20)**	**40.87(42.37)**	**N/A**	**14.67(10.99)**	**1.17(0.88)**	**11.20(6.80)**	**0.90(0.54)**
				15457.10(20.10)	163.49(5.54)	80.54(47.72)	N/A	21.08(20.06)	7.14(5.45)	83.14(97.32)	N/A	24.59(20.79)	2.19(1.89)	35.50(18.70)	2.96(1.42)
3	10	3	12	**5.40(6.40)**	**1.66(1.29)**	**16.72(18.05)**	**-19.17(-18.20)**	**5.96(5.78)**	**1.77(1.25)**	**28.35(22.92)**	**-10.31(-14.04)**	**0.45(0.34)**	**0.64(0.50)**	**0.23(0.22)**	**0.33(0.33)**
				23.83(23.56)	9.08(8.65)	55.42(62.17)	2.78(11.47)	26.98(26.19)	10.62(8.28)	67.39(58.56)	26.88(22.62)	1.12(0.91)	1.23(1.05)	0.63(0.43)	0.64(0.66)
	20	N/A	N/A	**7.54(8.31)(*)**	**37.67(1.61)**	**57.56(51.16)**	**N/A**	**8.54(7.76)**	**1.99(1.43)**	**69.11(63.66)**	**N/A**	**8.34(5.65)**	**1.19(0.80)**	**5.01(2.80)**	**0.61(0.36)**
				18.11(21.20)	351.25(3.49)	242.81(115.36)	N/A	23.34(18.04)	5.49(4.00)	174.49(142.54)	N/A	19.59(10.41)	2.30(1.55)	16.70(12.13)	1.96(1.21)
	30	N/A	N/A	**8.03(8.79)(*)**	**127.34(26.24)**	**86.37(66.09)**	**N/A**	**11.27(9.36)**	**3.05(2.38)**	**70.68(79.33)**	**N/A**	**59.79(43.01)**	**1.60(1.07)**	**51.14(19.87)**	**1.33(0.50)**
				13.49(19.86)	407.16(361.63)	224.37(156.53)	N/A	25.63(20.79)	5.60(4.40)	133.08(147.88)	N/A	142.22(105.64)	4.18(2.46)	110.19(44.54)	3.24(1.04)
4	10	1	7(**)	**2.16(2.09)**	**0.33(0.27)**	**14.56(13.90)**	**-43.26(-43.15)**	**1.96(2.01)**	**0.29(0.31)**	**32.47(30.54)**	**-29.88(-29.72)**	**0.38(0.33)**	**0.37(0.32)**	**0.23(0.24)**	**0.23(0.23)**
				5.65(5.38)	1.15(1.09)	66.47(66.47)	0.15(0.15)	4.60(5.50)	1.09(1.30)	86.79(79.39)	15.01(15.01)	0.99(0.79)	1.03(0.81)	0.55(0.50)	0.57(0.52)
	20	N/A	N/A	**4.96(5.52)(*)**	**61.62(2.05)**	**56.71(49.02)**	**N/A**	**6.55(5.42)**	**2.33(1.90)**	**77.58(80.26)**	**N/A**	**14.94(12.55)**	**1.01(0.83)**	**8.39(3.66)**	**0.56(0.25)**
				13.68(12.44)	399.69(4.95)	217.50(137.40)	N/A	21.14(19.85)	5.95(5.30)	161.94(217.50)	N/A	54.77(34.41)	3.19(2.29)	35.99(11.46)	2.40(0.72)
	30	N/A	N/A	**5.17(6.17)(*)**	**222.15(1.64)**	**106.12(64.45)**	**N/A**	**6.25(5.92)**	**1.79(1.49)**	**99.50(105.62)**	**N/A**	**44.34(88.65)**	**0.63(1.16)**	**85.01(33.47)**	**1.10(0.43)**
				9.87(14.51)	475.71(3.31)	211.69(162.91)	N/A	12.35(13.19)	3.62(2.30)	146.73(183.37)	N/A	304.11(173.01)	3.50(2.10)	294.62(95.70)	4.21(1.22)

(†) In each cell, the number outside (inside) the parentheses is the measure without (with) local improvement.
For each n, the 1st (2nd) row indicates the average (worst case) performance.

(*) Negative lower bounds from problems with no local improvement not included in the ratio. The actual average gap across 15 instances is higher.

(**) In this row, IF-AP is computed across instances for which a feasible solution exists.

Table 5: Effect of the Number of Machines and Jobs: Average and Worst Case Gaps of Heuristics and Bounds, CPU Times.

		AMPL/CPLEX		Percentage Gap (†)								CPU Time (sec.)(†)			
m	**n**	**#OP**	**#BI**	**LM2**				**ALM2**				**LM2**		**ALM2**	
				LB	**MA-LP**	**IF-LP**	**IF-AP**	**LB**	**MA-LP**	**IF-LP**	**IF-AP**	**MA**	**MA/LP**	**MA**	**MA/LP**
2	10	14	1	**16.59(19.88)**	**3.11(2.81)**	**21.68(20.58)**	**8.17(7.14)**	**18.01(16.15)**	**3.93(2.19)**	**28.26(25.61)**	**13.79(11.13)**	**0.39(0.22)**	**1.30(0.70)**	**0.37(0.20)**	**1.17(0.64)**
				53.23(82.18)	13.50(14.40)	62.13(57.10)	24.71(17.01)	98.15(51.78)	26.68(10.59)	68.82(86.03)	34.33(26.90)	0.65(0.39)	2.50(1.03)	1.74(0.32)	5.61(1.04)
	20	0	9(**)	**22.93(17.15)**	**5.49(3.52)**	**44.35(45.46)**	**-44.58(-43.61)**	**15.11(13.79)**	**5.64(4.11)**	**55.01(51.20)**	**-36.39(-42.68)**	**3.64(2.03)**	**1.37(0.77)**	**1.83(1.36)**	**0.72(0.58)**
				110.09(58.35)	14.91(9.67)	106.79(83.00)	14.23(7.63)	30.17(33.39)	17.73(14.52)	137.11(131.48)	49.46(28.58)	4.95(2.68)	2.19(1.28)	3.34(3.30)	1.90(2.20)
	30	N/A	N/A	**14.24(12.35)**	**4.39(3.01)**	**40.27(33.40)**	**N/A**	**12.87(10.75)**	**4.33(3.03)**	**42.01(43.75)**	**N/A**	**16.98(9.20)**	**1.35(0.73)**	**8.75(8.32)**	**0.71(0.66)**
				25.55(20.34)	8.58(7.96)	86.38(53.37)	N/A	23.44(24.28)	8.85(5.72)	76.06(107.31)	N/A	22.18(15.44)	1.97(1.38)	22.40(26.40)	1.87(1.89)
3	10	3	12	**4.91(5.36)**	**1.50(1.65)**	**17.94(18.57)**	**-18.14(-17.49)**	**4.44(4.35)**	**1.37(1.27)**	**60.04(43.21)**	**13.92(3.87)**	**0.55(0.39)**	**0.81(0.58)**	**0.27(0.23)**	**0.39(0.34)**
				18.67(21.51)	9.02(8.28)	56.17(58.71)	3.03(7.99)	18.63(21.51)	8.67(8.28)	208.43(131.34)	199.87(103.19)	1.25(0.71)	1.48(1.03)	0.72(0.48)	1.11(0.74)
	20	N/A	N/A	**9.12(8.72)**	**2.15(1.70)**	**48.77(52.64)**	**N/A**	**8.24(8.09)**	**1.96(1.40)**	**120.63(65.70)**	**N/A**	**8.35(4.70)**	**1.16(0.65)**	**4.01(2.22)**	**0.50(0.29)**
				22.52(18.98)	5.38(3.74)	119.85(121.35)	N/A	19.56(23.28)	4.48(3.76)	289.16(156.26)	N/A	17.21(8.13)	2.02(0.96)	14.58(5.56)	1.72(0.56)
	30	N/A	N/A	**11.40(9.42)**	**3.07(2.19)**	**59.35(59.52)**	**N/A**	**8.63(8.66)**	**2.89(2.08)**	**89.58(90.40)**	**N/A**	**52.65(39.95)**	**1.33(0.99)**	**28.46(14.68)**	**0.71(0.37)**
				29.25(21.35)	6.23(4.75)	93.89(93.77)	N/A	16.19(20.78)	4.88(3.40)	316.84(185.27)	N/A	88.56(119.99)	2.60(2.55)	75.77(30.88)	1.85(0.75)
4	10	1	7(**)	**2.11(2.04)**	**0.29(0.33)**	**14.72(12.99)**	**-43.30(-42.94)**	**1.99(1.98)**	**0.32(0.32)**	**92.45(107.27)**	**-18.00(-17.09)**	**0.63(0.50)**	**0.61(0.48)**	**0.21(0.21)**	**0.20(0.21)**
				4.99(4.86)	1.46(1.86)	92.35(58.21)	0.15(0.15)	4.63(4.70)	1.18(1.18)	196.47(409.21)	35.59(45.93)	1.20(1.11)	1.25(1.16)	0.43(0.46)	0.45(0.48)
	20	N/A	N/A	**6.75(5.71)**	**2.47(2.11)**	**46.30(49.13)**	**N/A**	**5.11(4.54)**	**2.48(1.86)**	**159.19(86.09)**	**N/A**	**13.96(9.37)**	**1.01(0.66)**	**5.48(3.38)**	**0.37(0.23)**
				21.03(15.68)	5.89(7.64)	125.21(146.00)	N/A	14.72(12.85)	9.83(6.42)	399.80(218.58)	N/A	31.38(16.53)	2.09(1.01)	16.62(8.16)	1.04(0.43)
	30	N/A	N/A	**6.59(6.70)**	**1.91(1.57)**	**63.04(67.25)**	**N/A**	**5.41(4.57)**	**1.80(1.31)**	**226.67(154.73)**	**N/A**	**96.01(69.44)**	**1.31(0.95)**	**39.54(14.61)**	**0.51(0.19)**
				12.21(18.83)	3.49(4.25)	159.75(164.30)	N/A	11.29(9.99)	4.31(2.21)	781.26(510.74)	N/A	143.88(111.72)	1.95(1.68)	110.82(40.25)	1.54(0.52)

(†) In each cell, the number outside (inside) the parentheses is the measure without (with) local improvement.
For each n, the 1^{st} (2^{nd}) row indicates the average (worst case) performance.

(**) In this row, IF-AP is computed across instances for which a feasible solution exists.

Table 6: Effect of the Number of Machines and Jobs: Average and Worst Case Gaps of Heuristics and Bounds, CPU Times.

			Percentage Gap (†)												CPU Time (sec.)(†)							
m	**n**	$p_{ij},\ \forall i$	**LM1**			**ALM1**			**LM2**			**ALM2**			**LM1**		**ALM1**		**LM2**		**ALM2**	
			LB	**MA-LP**	**IF-LP**	**LB**	**MA-LP**	**IF-LP**	**LB**	**MA-LP**	**IF-LP**	**LB**	**MA-LP**	**IF-LP**	**MA**	**MA/LP**	**MA**	**MA/LP**	**MA**	**MA/LP**	**MA**	**MA/LP**
2	10	U[1, 10]	**17.27**	**2.62**	**18.93**	**20.04**	**2.52**	**24.59**	**19.88**	**2.81**	**20.58**	**16.15**	**2.19**	**25.61**	**0.25**	**0.80**	**0.23**	**0.71**	**0.22**	**0.70**	**0.20**	**0.64**
			57.68	11.56	55.04	74.82	14.56	71.60	82.18	14.40	57.10	51.78	10.59	86.03	0.39	1.39	0.68	1.36	0.39	1.03	0.32	1.04
		U[1, 20]	**19.03**	**3.72**	**19.71**	**12.21**	**3.26**	**27.96**	**18.76**	**5.24**	**24.75**	**10.75**	**2.91**	**24.11**	**0.28**	**0.31**	**0.21**	**0.23**	**0.29**	**0.33**	**0.33**	**0.31**
			79.29	13.90	77.74	36.54	10.79	81.03	71.56	25.70	89.91	37.83	11.91	61.84	0.45	0.55	0.37	0.42	0.56	0.74	1.97	1.41
		U[1, 30]	**22.39**	**5.90**	**23.36**	**22.50**	**3.70**	**26.66**	**21.48**	**3.78**	**21.03**	**20.08**	**3.21**	**26.44**	**0.38**	**0.20**	**0.29**	**0.15**	**0.39**	**0.19**	**0.34**	**0.17**
			69.95	30.57	80.49	70.53	20.19	85.86	72.91	19.64	80.50	87.57	15.17	84.21	0.70	0.38	0.50	0.25	0.66	0.33	0.81	0.39
	20	U[1, 10]	**16.09**	**3.31**	**37.27**	**13.71**	**3.15**	**41.55**	**17.15**	**3.52**	**45.46**	**13.79**	**4.11**	**51.20**	**2.25**	**0.87**	**1.29**	**0.49**	**2.03**	**0.77**	**1.36**	**0.58**
			45.38	7.27	86.35	38.64	8.31	116.95	58.35	9.67	83.00	33.39	14.52	131.48	3.33	1.39	2.21	0.97	2.68	1.28	3.30	2.20
		U[1, 20]	**18.64**	**4.37**	**46.26**	**17.61**	**4.15**	**49.02**	**19.97**	**5.14**	**39.90**	**15.68**	**4.25**	**47.68**	**3.49**	**0.30**	**2.02**	**0.19**	**3.08**	**0.27**	**2.22**	**0.19**
			55.53	11.51	86.38	62.70	13.55	105.63	70.28	14.42	59.71	51.76	12.28	111.01	4.70	0.47	3.26	0.52	5.31	0.54	4.12	0.37
		U[1, 30]	**19.59**	**4.43**	**36.11**	**18.82**	**5.22**	**49.75**	**20.31**	**4.82**	**36.37**	**16.52**	**4.61**	**52.50**	**4.55**	**0.19**	**3.23**	**0.14**	**4.20**	**0.17**	**2.56**	**0.11**
			37.13	9.48	55.84	38.27	14.23	87.40	57.04	11.97	59.13	33.75	13.25	144.66	5.88	0.26	7.02	0.28	5.77	0.23	5.24	0.33
3	10	U[1, 10]	**6.40**	**1.29**	**18.05**	**5.78**	**1.25**	**22.92**	**5.36**	**1.65**	**18.57**	**4.35**	**1.27**	**43.21**	**0.34**	**0.50**	**0.22**	**0.33**	**0.39**	**0.58**	**0.23**	**0.34**
			23.56	8.65	62.17	26.19	8.28	58.56	21.51	8.28	58.71	21.51	8.28	131.34	0.91	1.05	0.43	0.66	0.71	1.03	0.48	0.74
		U[1, 20]	**7.91**	**1.52**	**31.01**	**7.82**	**1.70**	**47.74**	**7.95**	**1.82**	**37.00**	**7.25**	**1.99**	**67.45**	**0.45**	**0.22**	**0.34**	**0.17**	**0.50**	**0.23**	**0.32**	**0.15**
			54.98	12.07	99.35	53.16	15.67	135.14	53.53	16.48	142.89	48.01	19.55	234.12	1.01	0.57	0.55	0.32	1.28	0.47	0.63	0.28
		U[1, 30]	**8.64**	**1.09**	**33.99**	**6.77**	**0.74**	**44.65**	**7.77**	**1.06**	**30.73**	**7.01**	**1.01**	**91.37**	**0.53**	**0.09**	**0.39**	**0.07**	**0.64**	**0.12**	**0.38**	**0.07**
			51.20	5.00	173.11	28.05	2.33	154.31	37.54	5.03	111.55	36.67	4.71	232.77	1.13	0.20	0.86	0.10	1.30	0.19	0.77	0.10
	20	U[1, 10]	**8.31**	**1.61**	**51.16**	**7.76**	**1.43**	**63.66**	**8.72**	**1.70**	**52.64**	**8.09**	**1.40**	**65.70**	**5.65**	**0.80**	**2.80**	**0.36**	**4.70**	**0.65**	**2.22**	**0.29**
			21.20	3.49	115.36	18.04	4.00	142.54	18.98	3.74	121.35	23.28	3.76	156.26	10.41	1.55	12.13	1.21	8.13	0.96	5.56	0.56
		U[1, 20]	**8.24**	**2.05**	**58.03**	**7.33**	**1.69**	**80.87**	**7.09**	**1.77**	**57.27**	**6.77**	**1.73**	**80.23**	**7.37**	**0.31**	**2.99**	**0.13**	**5.92**	**0.27**	**3.65**	**0.13**
			32.74	5.82	134.21	22.45	3.58	210.58	16.61	4.21	151.20	29.32	4.84	213.62	16.29	0.65	6.50	0.23	9.95	0.49	17.90	0.32
		U[1, 30]	**10.21**	**2.56**	**60.14**	**10.91**	**2.52**	**80.22**	**10.42**	**2.51**	**57.36**	**8.70**	**2.40**	**117.48**	**10.67**	**0.17**	**5.86**	**0.10**	**10.74**	**0.18**	**4.65**	**0.08**
			33.00	9.37	128.06	30.23	8.18	208.62	28.95	10.51	122.57	30.03	10.94	261.16	22.95	0.32	12.57	0.18	20.49	0.29	9.03	0.16

(†) All problems are solved with local improvement. For each processing time distribution, the 1^{st} (2^{nd}) row indicates the average (worst case) performance.

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

A Appendix

A.1 Proof of Proposition 3.1

Proof. Clearly, LM1-PP1 is in $\mathcal{NP}$. To complete the proof, we map each instance of $1/r_j/\sum \beta_j C_j$ to an instance of LM1, and then show that the pricing problem corresponding to this instance of LM1 is equivalent to the original instance of the single-machine weighted completion time problem.

Now, given an instance of $1/r_j/\sum \beta_j C_j$, create an instance of Fm in which $m = 1,\ d_j = 0\ \forall j,\ h_{1j} = \beta_j\ \forall j$, and $\epsilon_j = \pi_j = 0\ \forall j$. The processing times and ready times are equal to those in the instance of the weighted completion time problem. Also, without loss of generality, we assume that the makespan in the weighted completion time problem is bounded by $\max_j r_j + \sum_{j=1}^{n} p_j$. Note that for any feasible schedule for the weighted completion time problem with a makespan greater than this quantity, there exists another feasible schedule with a makespan no larger than $\max_j r_j + \sum_{j=1}^{n} p_j$ and a smaller total cost. Then, applying the DW reformulation in Section 3 to the instance of Fm specified above, we obtain the LP master problem:

$$
\begin{aligned}
\min_{x_1^k} \quad & \sum_{k=1}^{K_1} \left[\sum_{j=1}^{n} \beta_j (C_{1j}^k - r_j - p_{1j}) \right] x_1^k \\
\text{s.t.} \quad & \\
(\gamma_1) \quad & \sum_{k=1}^{K_1} x_1^k = 1 \\
& x_1^k \geq 0 \qquad\qquad k = 1, \ldots, K_1
\end{aligned}
$$

The corresponding pricing problem has the form:

$$
\begin{aligned}
\min_{C_{1j}} \quad & \sum_{j=1}^{n} \beta_j C_{1j} - \sum_{j=1}^{n} \beta_j (r_j + p_{1j}) - \gamma_1 \\
\text{s.t.} \quad & \\
& \text{machine capacity constraints on machine 1} \\
& \underline{t}_{1j} \leq C_{1j} \leq \bar{t}_{1j} \qquad\qquad \forall j
\end{aligned}
$$

Observe that the expression $\left(- \sum_{j=1}^{n} \beta_j (r_j + p_{1j}) - \gamma_1 \right)$ is a constant in the pricing problem, where $\underline{t}_{1j} = r_{1j} + p_{1j} = r_j + p_{1j}$, and $\bar{t}_{1j} = t_{max} = \max_j \max(r_j, d_j) + \sum_{j=1}^{n} p_{1j} = \max_j r_j + \sum_{j=1}^{n} p_{1j}$. This completes the proof. ∎

A.2 Proof of Lemma 3.3

Proof. Let the optimal objective function value of LD1 be $z_{LD1} = z_{LR1(\mu^*_{ij})}$, and note that in LR1($\mu_{ij}$) the nonnegative variables $\{w_{ij}\}$ appear in the objective function only. First, assume that $\mu^*_{ij} > h_{ij}$ for some $i \geq 2$ and j. Then, the objective function coefficient of w_{ij} is negative, i.e., $(h_{ij} - \mu^*_{ij}) < 0$, and $z_{LR1(\mu^*_{ij})}$ is unbounded, contradicting its optimality for LD1. Hence, $\mu^*_{ij} \leq h_{ij}$ $(i = 2, \ldots, m,\ \forall j)$ which implies that $w^*_{ij} = 0$ $(i = 2, \ldots, m,\ \forall j)$ is optimal for LR1(μ^*_{ij}). The desired result follows by the equivalence of LM1 and LD1. ∎

A.3 Proof of Lemma 3.6

Proof. Consider the models LM1_∞ and LM2_∞ that correspond to LM1 and LM2, respectively, and in which we relax the upper bound constraints on the completion time variables in the schedules we generate. In these models, the set of all possible schedules on machine i, S_i, is replaced by $S_{i\infty}$, where in each schedule $S^k_{i\infty} \in S_{i\infty}$ the completion times $\{C^k_{ij}\}$ satisfy the operational ready time and the machine capacity constraints. The models LM1/LM1_∞ and LM2/LM2_∞ are identical in all other aspects, and clearly, they have the same optimal solutions.

Observe that for any optimal solution of LM1_∞ the dual variables must satisfy (3.32)-(3.34). Otherwise at least one of the pricing problems would be unbounded, i.e., we would be able to identify an infinite number of negatively priced schedules on machine i, contradicting the optimality of the current solution. Analogously, for any optimal solution of LM2_∞ the dual variables must satisfy (3.35)-(3.37). Thus, we conclude that for any optimal solution of LM1, (3.32)-(3.34) must hold. Similarly, any optimal solution of LM2 satisfies (3.35)-(3.37).

Then, the relationship (3.40) follows from (3.32), (3.33) and (3.38). To show (3.41) for μ^*_{ij}, consider the following inequalities:

$$\begin{aligned}
\mu^*_{2j} &\leq h_{1j} - h_{2j} \\
-\mu^*_{2j} + \mu^*_{3j} &\leq h_{2j} - h_{3j} \\
&\vdots \\
-\mu^*_{i-1j} + \mu^*_{ij} &\leq h_{i-1j} - h_{ij}
\end{aligned}$$

which belong to the set of inequalities (3.35) and (3.36). Their summation yields $\mu^*_{ij} \leq h_{1j} - h_{ij}$, and combined with (3.39), we obtain $\mu^*_{ij} \leq \min(0,\ h_{1j} - h_{ij})$. ∎

A.4 Proof of Lemma 3.8

Proof. By Lemma 3.6 and Definition 3.7, $\mu_{ij}^* \leq \min_{1\leq k\leq i} h_{kj} = h_{1j} < h_{ij}$ $(i = 2,\ldots,m,\ \forall j)$ which by complementary slackness and (3.13) implies that $w_{ij}^* = 0$ $(i = 2,\ldots,m,\ \forall j)$. ∎

A.5 Proof of Lemma 3.9

Proof. Let the optimal objective value of ALD1 be $z_{ALD1} = z_{ALR1(\mu_{ij}^*)}$, and note that in ALR1$(\mu_{ij})$ the nonnegative variables $\{v_{ij}\}$, $\{y_{ij}\}$ and $\{z_j\}$ appear in the objective function only. Thus, if at least one of these variables has a negative objective coefficient, $z_{ALR1(\mu_{ij}^*)}$ is unbounded, contradicting its optimality. Otherwise, it is optimal to set all artificial variables to zero. The case for ALD2 is proved analogously. ∎

A.6 Proof of Proposition 3.11

Proof. The result follows directly from the fact that the DW reformulation of TIFm is equivalent to IM1. To show this, note that we can reformulate TIFm using the same set of single-machine schedules $S_i = \{S_i^k\}$ $(i = 1,\ldots,m,\ k = 1,\ldots,K_i)$ as before, where $x_{ijt}^k = 1$ if $C_{ij}^k = t$, and zero otherwise. Let q_i^k be a binary variable that takes the value one if we choose the schedule S_i^k on machine i, and zero otherwise. Then, keeping the operation precedence constraints (2.15) in the integer programming master problem, the DW reformulation of TIFm is obtained as:

$$
\begin{aligned}
z_{ITIFm} = \min_{q_i^k,\ w_{ij}} \quad & \sum_{k=1}^{K_1}\left[\sum_{j=1}^{n}\sum_{t\in H_{1j}} h_{1j}(t-r_j-p_{1j})x_{1jt}^k\right] q_1^k \\
& + \sum_{i=2}^{m}\sum_{j=1}^{n} h_{ij}w_{ij} \\
& + \sum_{k=1}^{K_m}\left[\sum_{j=1}^{n}\sum_{\substack{t\in H_{mj}\\ t\leq d_j}} \epsilon_j(d_j-t)x_{mjt}^k + \sum_{\substack{t\in H_{mj}\\ t>d_j}} \pi_j(t-d_j)x_{mjt}^k\right] q_m^k \\
\textbf{(ITIFm)} \qquad \text{s.t.} \quad & \sum_{k=1}^{K_{i-1}}\left[\sum_{t\in H_{i-1j}} t x_{i-1jt}^k\right] q_{i-1}^k - \sum_{k=1}^{K_i}\left[\sum_{t\in H_{ij}} t x_{ijt}^k\right] q_i^k + w_{ij} = -p_{ij} \qquad i = 2,\ldots,m, \forall j \\
& \sum_{k=1}^{K_i} q_i^k = 1 \qquad i = 1,\ldots,m \\
& q_i^k \in \{0,\ 1\} \qquad i = 1,\ldots,m, \\
& \qquad\qquad\qquad\qquad\quad k = 1,\ldots,K_i \\
& w_{ij} \geq 0 \qquad j = 2,\ldots,m, \forall j
\end{aligned}
$$

Note that by (2.14) we have: $\sum_{t\in H_{ij}} t x_{ijt} = C_{ij}$, $\sum_{\substack{t\in H_{mj}\\ t\le d_j}} (d_j - t)x_{mjt} = E_j$, and $\sum_{\substack{t\in H_{mj}\\ t> d_j}} (t - d_j)x_{mjt} = T_j$. Substituting these equations into ITIFm, we observe that ITIFm is equivalent to IM1, and relaxing the integrality constraints on $\{q_i^k\}$ yields the linear programming master problem LTIFm. Then, we have $z_{LPTIFm} \le z_{LTIFm} = z_{LM1}$ where the inequality follows from the relationship of the LP relaxation of a model to the LP relaxation of its DW reformulation. See Wolsey (1998) for details. ∎

A.7 Proof of Lemma 3.12

Proof. Except for (3.34), all conditions in Lemmas 3.5 and 3.6 are explicitly satisfied by Steps 2-6 of Algorithm 1 for a dual vector from RLM1/RALM1. To see why (3.34) holds when Algorithm 1 terminates, note that at the end of Step 1, $\mu_{ij} \ge -\pi_j,\ i = 2, \dots, m$, for any job j. In Steps 2 through 6, the dual variables can only get smaller, however $\mu_{mj} \ge -\pi_j$ for all jobs is guaranteed by nonnegative inventory holding and tardiness costs.

Similarly, except for (3.37), all conditions in Lemmas 3.5 and 3.6 are satisfied by Steps 2-5 of Algorithm 2 for a dual vector from RLM2/RALM2. To show that (3.37) holds for all jobs when Algorithm 2 terminates, note that we have $\mu_{ij} \ge -\pi_j - h_{ij},\ i = 2, \dots, m$, for all jobs at the end of Step 1, and the inner "for"-loop in Steps 4-5 implies:

$$\mu_{mj} \le \mu_{ij} + \sum_{k=i}^{m-1} (h_{kj} - h_{k+1j}) = \mu_{ij} + h_{ij} - h_{mj},$$

where $\mu_{ij} + h_{ij} - h_{mj} \ge -\pi_j - h_{ij} + h_{ij} - h_{mj} = -\pi_j - h_{mj}$. Thus, at termination $\mu_{mj} \ge -\pi_j - h_{mj},\ \forall j$. ∎

A.8 Proof of Lemma 4.1

Proof. It is easy to see that the lemma holds for (4.1). To prove that (4.2) satisfies the conditions of Lemmas 3.5 and 3.6:

$$\begin{aligned}
\mu_{2j} = \min(h_{1j},\ h_{2j}) - h_{2j} \le h_{1j} - h_{2j} &\qquad \forall j \\
-\mu_{ij} + \mu_{i+1j} = h_{ij} - h_{i+1j} + (\min_{1\le k\le i+1} h_{kj} - \min_{1\le k\le i} h_{kj}) \le h_{ij} - h_{i+1j} &\qquad i = 2, \dots, m-1,\ \forall j \\
-\mu_{mj} = -\min_{1\le k\le m} h_{kj} + h_{mj} \le \pi_j + h_{mj} &\qquad \forall j \\
\mu_{ij} = \min_{1\le k\le i} h_{kj} - h_{ij} \le 0 &\qquad i = 2, \dots, m,\ \forall j \\
\mu_{ij} = \min_{1\le k\le i} h_{kj} - h_{ij} \le h_{1j} - h_{ij} &\qquad i = 2, \dots, m,\ \forall j
\end{aligned}$$

∎

References

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

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

Bülbül, K., Kaminsky, P., and Yano, C. (2001). Preemption in single machine earliness/tardiness scheduling. Submitted for publication. Department of IEOR, University of California at Berkeley.

Cattrysse, D., Salomon, M., Kuik, R., and van Wassenhove, L. (1993). A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times. *Management Science*, 39(4):477–486.

Chang, S.-C. and Liao, D.-Y. (1994). Scheduling flexible flow shops with no setup effects. *IEEE Transactions on Robotics and Automation*, 10(2):112–122.

Chen, H. and Luh, P. (1999). An alternative framework to Lagrangian relaxation approach for job shop scheduling. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 1, pages 913–918.

Chen, Z.-L. and Lee, C.-Y. (2002). Parallel machine scheduling with a common due window. *European Journal of Operations Research*, 136(3):512–527.

Chen, Z.-L. and Powell, W. (1999a). A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operations Research*, 116(1):220–232.

Chen, Z.-L. and Powell, W. (1999b). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1):78–94.

Dixon, P. and Poh, C. L. (1990). Heuristic procedures for multi-item inventory planning with limited storage. *IIE Transactions*, 22(2):112–123.

Drexl, A. and Kimms, A. (2001). Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, 52(3):340–351.

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.

Eilon, S. and Chowdhury, I. (1976). Due dates in job shop scheduling. *International Journal of Production Research*, 14(2):223–237.

Farley, A. (1990). A note on bounding a class of linear programming problems, including cutting stock problems. *Operations Research*, 38(5):922–923.

Federgruen, A. and Mosheiov, G. (1996). Heuristics for multimachine scheduling problems with earliness and tardiness costs. *Management Science*, 42(11):1544–1555.

Fisher, M. (1971). Optimal solution of scheduling problems using Lagrange multipliers. I. *Operations Research*, 21(5):1114–1127.

Goemans, M., Queyranne, M., Schulz, A., Skutella, M., and Wang, Y. (1999). Single machine scheduling with release dates. Technical report, CORE.

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.

Hall, N. and Posner, M. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Hoogeveen, J. and van de Velde, S. (1995). Stronger Lagrangian bounds by use of slack variables: applications to machine scheduling problems. *Mathematical Programming*, 70:173–190.

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

Kaskavelis, C. and Caramanis, M. (1998). Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. *IIE Transactions*, 30(11):1085–1097.

Kubiak, W., Lou, S., and Sethi, S. (1990). Equivalence of mean flow time problems and mean absolute deviation problems. *Operations Research Letters*, 9(6):371–374.

Lasdon, L. (1970). *Optimization theory for large systems*. Macmillan, New York.

Lee, C.-Y. and Chen, Z.-L. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics*, 47(2):145–165.

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

Park, M.-W. and Kim, Y.-D. (2000). A branch and bound algorithm for a production scheduling problem in an assembly system under due date constraints. *European Journal of Operations Research*, 123(3):504–518.

Pinedo, M. and Singer, M. (1999). A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics*, 46(1):1–17.

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

Queyranne, M. and Schulz, A. (1994). Polyhedral approaches to machine scheduling. Technical Report 408, Technische Universitaet Berlin.

Sarper, H. (1995). Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem. *Applied Mathematical Modelling*, 19(3):163–161.

Savelsbergh, M. and Sol, M. (1998). DRIVE: dynamic routing of independent vehicles. *Operations Research*, 46(4):474–490.

Schulz, A. and Skutella, M. (1997). Scheduling-LPs bear probabilities: randomized approximations for min-sum criteria. In *Proceedings of Fifth Annual European Symposium on Algorithms*, pages 416–429. Springer.

Sousa, J. and Wolsey, L. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 54:353–367.

Sundararaghavan, P. and Ahmed, M. (1984). Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. *Naval Research Logistics Quarterly*, 31(2):325–333.

Sung, C. and Min, J. (2001). Scheduling in a two-machine flowshop with batch processing machine(s) for earliness/tardiness measure under a common due date. *European Journal of Operations Research*, 131(1):95–106.

Van de Velde, S. (1990). Minimizing the sum of the job completion times in the two-machine flow shop by Lagrangian relaxation. *Annals of Operations Research*, 26(1–4):257–268.

Van den Akker, J., Hoogeveen, J., and van de Velde, S. (1999a). Parallel machine scheduling by column generation. *Operations Research*, 47(6):862–872.

Van den Akker, J., Hurkens, C., and Savelsbergh, M. (2000). Time-indexed formulations for machine scheduling problems: column generation. *INFORMS Journal on Computing*, 12(2):111–124.

Van den Akker, J., van Hoesel, C., and Savelsbergh, M. (1999b). A polyhedral approach to single-machine scheduling problems. *Mathematical Programming*, 85(3):541–572.

Van den Akker, M., Hoogeveen, H., and van de Velde, S. (2002). Combining column generation and Lagrangean relaxation to solve a single-machine common due date problem. *INFORMS Journal on Computing*, 14(1):37–51.

Vanderbeck, F. and Wolsey, L. (1996). An exact algorithm for IP column generation. *Operations Research Letters*, 19(4):151–159.

Wolsey, L. (1998). *Integer Programming*. Wiley, New York.