A Linear Programming-Based Method for Job Shop Scheduling

Kerem Bülbül
Sabancı University, Manufacturing Systems and Industrial Engineering, Orhanlı-Tuzla, 34956 Istanbul, Turkey
bulbul@sabanciuniv.edu

Philip Kaminsky
Industrial Engineering and Operations Research, University of California, Berkeley, CA
kaminsky@ieor.berkeley.edu

Abstract: We present a decomposition heuristic for a large class of job shop scheduling problems. This heuristic utilizes information from the linear programming formulation of the associated optimal timing problem to solve subproblems, can be used for any objective function whose associated optimal timing problem can be expressed as a linear program (LP), and is particularly effective for objectives that include a component that is a function of individual operation completion times. Using the proposed heuristic framework, we address job shop scheduling problems with a variety of objectives where intermediate holding costs need to be explicitly considered. In computational testing, we demonstrate the performance of our proposed solution approach.

Keywords: job shop; shifting bottleneck; intermediate inventory holding costs; non-regular objective; optimal timing problem; linear programming; sensitivity analysis; single machine; earliness/tardiness.

---

1. Introduction The job shop scheduling problem, in which each job in a set of orders requires processing on a unique subset of available resources, is a fundamental operations research problem, encompassing many additional classes of problems (single machine scheduling, flow shop scheduling, etc). While from an application perspective this model is traditionally used to sequence jobs in a factory, it is in fact much more general than this, as the resources being allocated can be facilities in a logistics network, craftsmen on a construction job site, etc. In light of both the practical and academic importance of this problem, many researchers have focused on various approaches to solving it. Exact optimization methods, however, have in general proved effective only for relatively small problem instances or simplified versions of the problem (certain single-machine and two-machine flow shop models, for example). Thus, in many cases, researchers who wish to use approaches more sophisticated than simple dispatch rules have been motivated to focus on heuristics for practically sized problem instances, typically metaheuristics (see Laha (2007) and Xhafa and Abraham (2008) and the references therein) or decomposition methods (see Ovacik and Uzsoy (1996) and the references therein). In almost all of this work, however, the objective is a function of the completion time of each job on its final machine, and is not impacted by the completion times of intermediate operations.

This is a significant limitation because objectives that are only a function of job completion times rather than a function of all operation completion times ignore considerations that are increasingly important as the focus on lean, efficient supply chains grows. For example, in many cases, intermediate inventory holding costs are an important cost driver, especially when entire supply chains are being modeled. Often, substantial value is added between processing stages in a production network, but intermediate products may be held in inventory for significant periods of time waiting for equipment to become available for the next processing or transfer step. Thus, in this case, significant savings may result from schedules that delay intermediate processing steps as much as possible. On the other hand, sometimes it makes sense for certain intermediate processing steps to be expedited. Consider, for example, processes when steel must be coated as soon as possible to delay corrosion, or when intermediates are unstable and degrade over time. Indeed, some supply chains and manufacturing processes may have certain steps that have to be expedited and other steps that have to be delayed in order to minimize costs.

Similarly, consider the so-called "rescheduling problem." Suppose that a detailed schedule exists, and all necessary arrangements have been made to accommodate that schedule. When a supply chain disruption of some kind occurs, so that the schedule has to be changed to account for changing demand or alternate resource availability, the impact of these changes can often be minimized if the new schedule

adheres as closely as possible to the old schedule. This can be accomplished by penalizing operations on machines that start at different times than those in the original schedule.

In these cases and others, a scheduling approach that considers only functions of the completion time of the job, even those that consider finished goods inventory holding cost (e.g., earliness cost) or explicitly penalize deviation from a targeted job completion time, may lead to a significantly higher cost solution than an approach that explicitly considers intermediate holding costs. We refer the reader to Bulbul (2002) and Bulbul et al. (2004) for further examples and a more in-depth discussion.

Unfortunately, the majority of previous work in this area (and of scheduling work in general) focuses on algorithms or approaches that are specific to an individual objective function, and are not adaptable to other objective functions in a straightforward way. Because each approach is highly specialized for a particular objective, it is difficult for a researcher or user to generalize insights for a particular approach to other objectives, and thus from an application point of view, software to solve scheduling problems is highly specialized and customized, and from a research point of view, scheduling research is fragmented. Indeed, published papers, algorithms, and approaches typically focus on a single objective: total completion time, flowtime, or tardiness, for example. It is quite uncommon to find an approach that is applicable to more than one (or one closely related set) of objectives.

Thus, there is a need for an effective, general approach to solve the growing class of scheduling problems that explicitly considers the completion time of intermediate operations. In this paper we address this need by developing an efficient, effective heuristic algorithmic framework useful for addressing job shop scheduling problems for a large class of objectives where operation completion times have a direct impact on the total cost. To clarify the exposition, we present our results in the context of explicitly minimizing intermediate holding costs, although our approach applies directly and without modification to other classes of problems where operation completion times are critical. This framework builds on the notion of the *optimal timing problem* for a job shop scheduling problem. For any scheduling problem where intermediate holding costs are considered, the solution to the problem is not fully defined by the sequence of operations on machines – it is necessary to specify the starting time of each operation on each machine, since the time that each job is idle between processing steps dictates intermediate holding costs. The problem of determining optimal start times of operations on machines *given the sequence of operations on machines* is known as the optimal timing problem, and for many job shop scheduling problems, this optimal timing problem can be expressed as an LP.

Specifically, **our algorithm applies to any job shop scheduling problem with operation completion time-related costs and any objective for which the optimal timing problem can be expressed as an LP**. As we will see, this includes, but is certainly not limited to, objectives that combine holding costs with total weighted completion time, total weighted tardiness, and makespan.

Our algorithm is a machine-based decomposition heuristic related to the *shifting-bottleneck* heuristic, an approach that was originally developed for the job shop makespan problem by Adams et al. (1988). Versions of this approach have been applied to job shop scheduling problems with maximum lateness (Demirkol et al. (1997)) and total weighted tardiness minimization objectives (Pinedo and Singer (1999), Singer (2001) and Mason et al. (2002)). None of these authors consider operation completion time-related costs, however, and each author presents a version of the heuristic specific to the objective they are considering. Our approach is general enough to encompass all of these objectives (and many more) combined with operation completion time-related costs. In addition, we believe and hope that other researchers can build on our ideas, particularly at the subproblem level, to further improve the effectiveness of our proposed approach.

In general, relatively little research exists on multi-machine scheduling problems with intermediate holding costs, even those focusing on very specific objectives. Avci and Storer (2004) develop effective local search neighborhoods for a broad class of scheduling problems that includes the job shop total weighted earliness and tardiness (E/T) scheduling 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 present approaches for very specific objective functions that do not fit into our framework, and use very different solution techniques. Ohta and Nakatanieng (2006) considers a job shop in which jobs must complete by their due dates, and develops a shifting bottleneck-based heuristic to minimize holding

costs. Thiagarajan and Rajendran (2005) and Jayamohan and Rajendran (2004) evaluate dispatch rules for related problems. In contrast, our approach applies to a much broader class of job shop scheduling problems.

2. Problem Description In the remainder of the paper, we restrict our attention to the job shop scheduling problem with intermediate holding costs in order to keep the discussion focused. However, we reiterate that many other job shop scheduling problems would be amenable to the proposed solution framework as long as their associated optimal timing problems are LP's.

Consider a non-preemptive job shop with m machines and n jobs, each of which must be processed on a subset of those machines. The operation sequence of job j is denoted by an ordered set M_j where the ith operation in M_j is represented by o_{ij}, $i = 1, \ldots, m_j =\mid M_j \mid$, and J_i is the set of operations to be processed on machine i. For clarity of exposition, from now on we assume that the ith operation o_{ij} of job j is performed on machine i in the definitions and model below. However, our proposed solution approach applies to general job shops, and for our computational testing we solve problems with more general routing.

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 for job j; and h_{ij}, the holding cost per unit time for job j while it is waiting in the queue before machine i. All ready times, processing times and due dates are assumed to be integer.

For a given schedule S, let w_{ij} be the time job j spends in the queue before machine i, and let C_{ij} be the time at which job j finishes processing on machine i. We are interested in objective functions with two components, each a function of a particular schedule: an intermediate holding cost component $H(S)$, and a $C(S)$ that is a function of the completion times of each job. The intermediate holding cost component can be expressed as follows:

$$H(S) = \sum_{j=1}^{n} \sum_{i=1}^{m_j} h_{ij} w_{ij}.$$

Before detailing permissible $C(S)$ functions, we formulate the m-machine job shop scheduling problem (**Jm**):

$$
\begin{aligned}
(\mathbf{Jm}) \quad \min \quad & H(S) + C(S) && && (1)\\
\text{s.t.} \quad & && && \\
& C_{1j} - w_{1j} = r_j + p_{1j} && \forall j && (2)\\
& C_{i-1j} - C_{ij} + w_{ij} = -p_{ij} && i = 2, \ldots, m_j,\ \forall j && (3)\\
& C_{ik} - C_{ij} \geq p_{ik} \ \textbf{or}\ C_{ij} - C_{ik} \geq p_{ij} && \forall i, \forall j, k \in J_i && (4)\\
& C_{ij}, w_{ij} \geq 0 && i = 1, \ldots, m_j, \forall j. && (5)
\end{aligned}
$$

Constraints (2) prevent processing of jobs before their respective ready times. Constraints (3), referred to as operation precedence constraints, prescribe that a job j follows its processing sequence $o_{1j}, \ldots, o_{m_j j}$. Machine capacity constraints (4) ensure that a machine processes only one operation at a time, and an operation is finished once started. Observe that even if the objective function is linear, due to constraints (4) the formulation is not linear (without a specified order of operations).

The technique we present in this paper is applicable to any objective function $C(S)$ that can be modeled as a linear objective term along with additional variables and linear constraints added to formulation (**Jm**). Although this allows a rich set of possible objectives, to clarify our exposition, for our computational experiments we focus on a specific formulation that we call ($\widehat{\mathbf{Jm}}$) that models total weighted earliness, total weighted tardiness, total weighted completion time, and makespan objectives. For this formulation, in addition to the parameters introduced above, d_j represents the due date for job j; ϵ_j is the earliness cost per unit time if job j completes its final operation before time d_j; π_j represents the tardiness cost per unit time if job j completes its final operation after time d_j; and γ represents the penalty per unit time associated with the makespan. Variables E_j and T_j model the earliness, $\max(d_j - C_{m_j j}, 0)$, and the tardiness, $\max(C_{m_j j} - d_j, 0)$, of job j, respectively. Consequently, the total weighted earliness and tardiness are expressed as $\sum_j \epsilon_j E_j$ and $\sum_j \pi_j T_j$, respectively. Note that if $d_j = 0\ \forall j$, then $T_j = C_j\ \forall j$, and the total weighted tardiness reduces to the total weighted completion time. The variable $C_{\max}$ represents the makespan and is set to $\max_j C_{m_j j}$ in the model, where s_j is the slack of job j with respect to the makespan.

Formulation ($\widehat{\mathbf{Jm}}$), which we present below, thus extends formulation (**Jm**) with additional variables and constraints. Constraints (7) relate the completion times of the final operations to the earliness and tardiness values, and constraints (8) ensure that the makespan is correctly identified.

$$(\widehat{\mathbf{Jm}}) \quad \min \quad \sum_{j=1}^{n}\sum_{i=1}^{m_j} h_{ij}w_{ij} + \sum_{j=1}^{n}(\epsilon_j E_j + \pi_j T_j) + \gamma C_{\max} \tag{6}$$

s.t.

$$(2)-(5)$$

$$C_{m_j j} + E_j - T_j = d_j \qquad \forall j \tag{7}$$

$$C_{m_j j} - C_{\max} + s_j = 0 \qquad \forall j \tag{8}$$

$$E_j, T_j, s_j \geq 0 \qquad \forall j \tag{9}$$

$$C_{\max} \geq 0. \tag{10}$$

Following the three field notation of Graham et al. (1979), $Jm/r_j/\sum_{j=1}^{n}\sum_{i=1}^{m_j} h_{ij}w_{ij} + \sum_{j=1}^{n}(\epsilon_j E_j + \pi_j T_j) + \gamma C_{\max}$ represents problem ($\widehat{\mathbf{Jm}}$). ($\widehat{\mathbf{Jm}}$) is strongly $\mathcal{NP}$-hard, as a single-machine special case of this problem with all inventory holding and earliness costs and the makespan cost equal to zero, i.e., the single-machine total weighted tardiness problem $1/r_j/\sum \pi_j T_j$, is known to be strongly $\mathcal{NP}$-hard (Lenstra et al. (1977)).

In the next section of this paper, Section 3, we explain our heuristic for this model. The core of our solution approach is the single-machine subproblem developed conceptually in Section 3.3.1 and analyzed in depth in Appendix A. In Section 4, we extensively test our heuristic, and compare it to the current best approaches for related problems. Finally, in Section 5, we conclude and explore directions for future research.

3. Solution Approach As mentioned above, we propose a shifting bottleneck (SB) heuristic for this problem, and our algorithm makes frequent use of the optimal timing problem related to our problem, and is best understood in the context of the disjunctive graph representation of the problem, so in the next two subsections, we review these. For reasons that will become clear in Section 3.3.1, we refer to our SB heuristic as the Shifting Bottleneck Heuristic Utilizing Timing Problem Duals (SB-TPD).

3.1 The Timing Problem Observe that (**Jm**) with a linear objective function would be an LP if we knew the sequence of operations on each machine (which would imply that we could pre-select one of the terms in constraint (4) of (**Jm**)). Indeed, researchers often develop two-phase heuristics for similar problems based on this observation, where first a processing sequence is developed, and then idle time is inserted by solving the optimal timing problem.

For our problem, once operations are sequenced, and assuming operations are renumbered in sequence order on each machine, the optimal schedule is obtained by solving the associated timing problem (**TTJm**), defined below. The variable i_{ij} denotes the time that operation j waits before processing on machine i (that is, the time between when machine i completes the previous operation, and the time that operation j begins processing on machine i) .

$$(\mathbf{TTJm}) \quad \min \quad H(S) + C(S) \tag{11}$$

s.t.

$$(2), (3), (5)$$

$$C_{ij-1} - C_{ij} + i_{ij} = -p_{ij} \qquad \forall i, j \in J_i, j \neq 1 \tag{12}$$

$$i_{ij} \geq 0 \qquad \forall i, j \in J_i, j \neq 1 \tag{13}$$

Specifically, in our approach, we construct operation processing sequences by solving the subproblems of a SB heuristic. Once the operation processing sequences are obtained, we find the optimal schedule given these sequences by solving (**TTJm**). The linear program (**TTJm**) has $O(nm)$ variables and constraints. As mentioned above, to illustrate our approach, we focus on a specific example, ($\widehat{\mathbf{Jm}}$). The timing problem for ($\widehat{\mathbf{Jm}}$), ($\widehat{\mathbf{TTJm}}$), follows.

$$(\widehat{\mathbf{TTJm}}) \quad \min \quad \sum_{j=1}^{n}\sum_{i=1}^{m_j} h_{ij}w_{ij} + \sum_{j=1}^{n}(\epsilon_j E_j + \pi_j T_j) + \gamma C_{\max} \tag{14}$$

s.t.

(2), (3), (5)
(7) – (10)
(12) – (13)

Also, for some of our computational work, it is helpful to add two additional constraints to formulation ($\widehat{\textbf{TTJm}}$),

$$C_{\max} \leq C_{\max}^{UB} \tag{15}$$

$$\sum_{j=1}^{n} \pi_j T_j \leq WT^{UB}, \tag{16}$$

where $C_{\max}^{UB}$ and WT^{UB} are upper bounds on $C_{\max}$ and the total weighted tardiness, respectively.

3.2 Disjunctive Graph Representation The disjunctive graph representation of the scheduling problem plays a key role in the development and illustration of our algorithm. Specifically, the disjunctive graph representation $G(N, A)$ for an instance of our problem is given in Figure 1, where the machine processing sequences for jobs 1, 2, 3 are given by $M_1 = \{o_{11}, o_{31}, o_{21}\}$, $M_2 = \{o_{22}, o_{12}, o_{32}\}$, and $M_3 = \{o_{23}, o_{13}, o_{33}\}$, respectively. There are three types of nodes in the node set N: one node for each operation o_{ij}, one dummy starting node S and one dummy terminal node T, and one dummy terminal node F_j per job associated with the completion of the corresponding job j. The arc set A consists of two types of arcs: the solid arcs in Figure 1 represent the operation precedence constraints (3) and are known as conjunctive arcs. The dashed arcs in Figure 1 are referred to as disjunctive arcs, and they correspond to the machine capacity constraints (4).

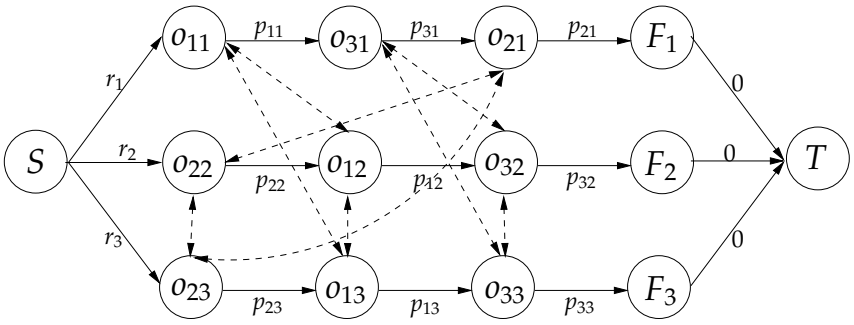


Figure 1: Disjunctive graph representation for (**Jm**).

Before a specific schedule is determined for a problem, there is initially a pair of disjunctive arcs between each pair of operations on the same machine (one in each direction). The set of conjunctive and disjunctive arcs are denoted by A_C and A_D, respectively, and we have $A = A_C \cup A_D$. Both conjunctive and disjunctive arcs emanating from a node o_{ij} have a length equal to the processing time p_{ij} of operation o_{ij}. The ready time constraints (2) are incorporated by connecting the starting node S to the first operation of each job j by an arc of length r_j. The start time of the dummy terminal node T marks the makespan.

A feasible sequence of operations for (**Jm**) corresponds to a selection of exactly one arc from each pair of disjunctive arcs (also referred to as fixing a pair of disjunctive arcs) so that the resulting graph $G'(N, A_C \cup A_D^S)$ is acyclic where A_D^S denotes the set of disjunctive arcs included in G'. However, recall that by itself, this fixing of disjunctive arcs does not completely describe a schedule for (**Jm**). The operation completion times and the objective value corresponding to G' are obtained by solving (**TTJm**) where the constraints (12) corresponding to A_D^S are included. Note that a disjunctive arc $(o_{ij}, o_{ik}) \in A_D^S$ corresponds to a constraint $C_{ij} - C_{ik} + i_{ik} = -p_{ik}$ in (**TTJm**).

3.3 Key Steps of the Algorithm SB-TPD is an iterative machine-based decomposition algorithm.

- **First**, a disjunctive graph representation of the problem is constructed. Initially, there are no machines scheduled, so that no disjunctive arcs are fixed, i.e., $A_D^S = \emptyset$. This implies that all

machine capacity constraints are initially ignored, and the machines are in effect allowed to process as many operations as required simultaneously.

- **At each iteration** of SB-TPD, one single-machine subproblem is solved for each unscheduled machine (we detail the single-machine subproblem below), the "bottleneck machine" is selected from among these (we detail bottleneck machine selection below), and the disjunctive arcs corresponding the schedule on this "bottleneck machine" are added to A_D^S. As we discuss below, the disjunctive graph is used to characterize single-machine problems at each iteration of the problem, and to identify infeasible schedules. Finally, the previous scheduling decisions are re-evaluated, and some machines are re-scheduled if necessary.
- **These steps are repeated** until all machines are scheduled and a feasible solution to the problem (**Jm**) is obtained.
- **A partial tree search** over the possible orders of scheduling the machines performs the loop in the previous steps several times. Multiple feasible schedules for (**Jm**) are obtained and the best one is picked as the final schedule produced by SB-TPD.

In the following subsections, we provide more detail.

3.3.1 The Single-Machine Problem The key component of any SB algorithm is defining an appropriate single-machine subproblem. The SB procedure starts with no machine scheduled and determines the schedule of one additional machine at each iteration. The basic rationale underlying the SB procedure dictates that we select the machine that hurts the *overall* objective the most as the next machine to be scheduled, given the schedules of the currently scheduled machines. Thus, the single-machine subproblem defined must capture accurately the effect of scheduling a machine on the overall objective function. In the following discussion, assume that the algorithm is at the start of some iteration, and let $\mathcal{M}$ and $\mathcal{M}^S \subset \mathcal{M}$ denote the set of all machines and the set of machines already scheduled, respectively. Since "machines being scheduled" corresponds to fixing disjunctive arcs, observe that at this stage of the algorithm, the partial schedule is represented by disjunctive graph $G'(N, A_C \cup A_D^S)$ where A_D^S corresponds to the selection of disjunctive arcs fixed for the machines in $\mathcal{M}^S$.

In our problem, the overall objective function value and the corresponding operation completion times are obtained by solving (**TTJm**). The formulation given in (11)-(13) requires all machine sequences to be specified. However, note that we can also solve an intermediate version of (**TTJm**) – one that only includes machine capacity constraints corresponding to the machines in $\mathcal{M}^S$ while omitting the capacity constraints for the remaining machines in $\mathcal{M} \setminus \mathcal{M}^S$. We refer to this intermediate optimal timing problem and its optimal objective value as (**TTJm**)($\mathcal{M}^S$) and $z_{(\mathbf{TTJm})}(\mathcal{M}^S)$, respectively, and say that (**TTJm**) is solved over the disjunctive graph $G'(N, A_C \cup A_D^S)$.

Observe that initially SB-TPD starts with no machine scheduled, i.e., $\mathcal{M}^S$ is initially empty. Therefore, at the initialization step (**TTJm**) is solved over a disjunctive graph $G'(N, A_C \cup A_D^S)$ where $A_D^S = \emptyset$ by excluding all machine capacity constraints (12) and yields a lower bound on the optimal objective value of the original problem (**Jm**).

Once again, assume that algorithm is at the start of an iteration, so that a set of machines $\mathcal{M}^S$ is already scheduled and the disjunctive arcs selected for these machines are included in A_D^S. The optimal completion times C_{ij}^* for all $i \in \mathcal{M}$ and $j \in J_i$ are also available from (**TTJm**)($\mathcal{M}^S$) . As the current iteration progresses, a new bottleneck machine i^b must be identified by determining which of the currently unscheduled machines $i \in \mathcal{M} \setminus \mathcal{M}^S$ will have the largest impact on the objective function of (**TTJm**)($\mathcal{M}^S \cup i$) if it is sequenced effectively (that is, in the way that minimizes its impact). Then, a set of disjunctive arcs for machine i^b corresponding to the sequence provided by the corresponding subproblem is added to A_D^S, and a new set of optimal completion times C'_{ij} for all $i \in \mathcal{M}$ and $j \in J_i$ is determined by solving (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$).

Clearly, the optimal objective value of (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$) is no less than that of (**TTJm**)($\mathcal{M}^S$), i.e., $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\}) \geq z_{(\mathbf{TTJm})}(\mathcal{M}^S)$ must hold. Therefore, a reasonable objective for the subproblem of machine i is to minimize the difference $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i\}) - z_{(\mathbf{TTJm})}(\mathcal{M}^S)$. In the remainder of this section, we show how this problem can be solved approximately as a single-machine E/T scheduling problem with distinct ready times and due dates.

For defining the subproblem of machine i, we note that if the completion times obtained from (**TTJm**)($\mathcal{M}^S$) for the set of operations J_i to be performed on machine i are equal to those obtained from (**TTJm**)($\mathcal{M}^S \cup \{i\}$) after adding the corresponding machine capacity constraints, i.e., if $C^*_{ij} = C'_{ij}$ for all $j \in J_i$, then we have $z_{(\mathbf{TTJm})}(\mathcal{M}^S) = z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i\})$. This observation implies that we can regard the current operation completion times C^*_{ij} provided by (**TTJm**)($\mathcal{M}^S$) as due dates in the single-machine subproblems. Early and late deviations from these due dates are discouraged by assigning them earliness and tardiness penalties, respectively. These penalties are intended to represent the impact on the overall problem objective if operations are moved earlier or later because of the way a machine is sequenced.

Specifically, for a machine $i \in \mathcal{M} \setminus \mathcal{M}^S$, some of the operations in J_i may overlap in the optimal solution of (**TTJm**)($\mathcal{M}^S$) because this timing problem excludes the capacity constraints for machine i. Thus, scheduling a currently unscheduled machine i implies removing the overlaps among the operations on this machine by moving them earlier/later in time. This, of course, may also affect the completion times of operations on other machines. For a given operation o_{ij} on machine i, assume that $C^*_{ij} = d_{ij}$ in the optimal solution of (**TTJm**)($\mathcal{M}^S$). Then, we can measure the impact of moving operation o_{ij} for $\delta > 0$ time units earlier or later on the overall objective function by including a constraint of the form

$$C_{ij} + s_{ij} = d_{ij} - \delta \quad (C_{ij} \le d_{ij} - \delta) \qquad \text{or} \tag{17}$$

$$C_{ij} - s_{ij} = d_{ij} + \delta \quad (C_{ij} \ge d_{ij} + \delta), \tag{18}$$

respectively, in the optimal timing problem (**TTJm**)($\mathcal{M}^S$) and resolving it, where the variable $s_{ij} \ge 0$ denotes the slack or surplus variable associated with (17) or (18), respectively, depending on the context. Of course, optimally determining the impact on the objective function for all values of δ is computationally prohibitive as we explain later in this section. However, as we demonstrate in Appendix A, the increase in the optimal objective value of (**TTJm**)($\mathcal{M}^S$) due to an additional constraint (17) or (18) can be bounded by applying sensitivity analysis to the optimal solution of (**TTJm**)($\mathcal{M}^S$) to determine the value of the dual variables associated with the new constraints.

Specifically, we show the following:

Proposition 3.1 *Consider the optimal timing problems* (**TTJm**)($\mathcal{M}^S$) *and* (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$) *solved in iterations* k *and* $k+1$ *of SB-TPD where* i^b *is the bottleneck machine in iteration* k*. For any operation* $o_{i^b j}$*, if* $C'_{i^b j} = C_{i^b j} - \delta$ *or* $C'_{i^b j} = C_{i^b j} + \delta$ *for some* $\delta > 0$*, then* $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\}) - z_{(\mathbf{TTJm})}(\mathcal{M}^S) \ge | \bar{\mathbf{y}}''_{m+1} | \delta \ge 0$*, where* $\bar{\mathbf{y}}''$ *is defined in Appendix A in* (36)-(37).

$\bar{\mathbf{y}}''_{m+1}$ is the value of the dual variable associated with (17) or (18) if we augment (**TTJm**)($\mathcal{M}^S$) with (17) or (18), respectively, and carry out a single dual simplex iteration. Thus, the cost increase characterized in Proposition 3.1 is in some ways related to the well-known shadow price interpretation of the dual variables. In Appendix A, we give a closed form expression for $\bar{\mathbf{y}}''$ that can be calculated explicitly using only information present in the optimal basic solution to (**TTJm**)($\mathcal{M}^S$). Thus, we can efficiently bound the impact of pushing an operation earlier or later by δ time units on the overall objective function from below. This allows us to formulate the single-machine subproblem of machine i in SB-TPD as a single-machine E/T scheduling problem $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$ with the following parameters: the ready time r_{ij} of job j on machine i is determined by the longest path from node S to node o_{ij} in the disjunctive graph $G'(N, A_C \cup A^S_D)$; the due date d_{ij} of job j on machine i is the optimal completion time C^*_{ij} of operation o_{ij} in the current optimal timing problem (**TTJm**)($\mathcal{M}^S$); the earliness and tardiness costs ϵ_{ij} and π_{ij} of job j on machine i are given by

$$\epsilon_{ij} = -\bar{\mathbf{y}}''_{m+1} = \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} = -\max_{k \ne j | \bar{\mathbf{A}}_{jk} > 0} \frac{\bar{\mathbf{c}}_k}{-\bar{\mathbf{A}}_{jk}} \quad \text{and} \quad \pi_{ij} = \bar{\mathbf{y}}''_{m+1} = -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} = -\max_{k | \bar{\mathbf{A}}_{jk} < 0} \frac{\bar{\mathbf{c}}_k}{\bar{\mathbf{A}}_{jk}}, \tag{19}$$

respectively, where these quantities are defined in (34) and (36)-(37). (If $C^*_{ij} = r_{ij} + p_{ij}$, then it is not feasible to push operation o_{ij} earlier, and ϵ_{ij} is set to zero.) As we detail in Appendix A, this cost function, developed for shifting a single operation o_{ij} earlier or later, is based on a single implicit dual simplex iteration after adding the constraint (17) or (18) to (**TTJm**)($\mathcal{M}^S$). **We are therefore only able to obtain a lower bound** on the actual change in cost that would result from changing C_{ij} from its current value C^*_{ij}. In general, the amount of change in cost would be a piecewise linear and convex function as illustrated in Figure 2. However, while the values of ϵ_{ij} and π_{ij} in (19) may be computed

efficiently based on the current optimal basis of (**TTJm**)($\mathcal{M}^S$) – see Appendix B for an example on ($\widehat{\textbf{TTJm}}$) –, we detail at the end of Appendix A how determining the actual cost functions requires solving one LP with a parametric right hand side for each operation, and is therefore computationally expensive. In addition, the machine capacity constraints are introduced simultaneously for all of the operations on the bottleneck machine in SB-TPD, and there is no guarantee that this combined effect is close to the sum of the individual effects. However, as we demonstrate in our computational experiments in Section 4, the single-machine subproblems provide reasonably accurate bottleneck information and lead to good operation processing sequences. We also note that the single-machine E/T scheduling problem $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$ is strongly $\mathcal{NP}$-hard because a special case of this problem with all earliness costs equal to zero, i.e., the single-machine total weighted tardiness problem $1/r_j/\sum \pi_j T_j$, is strongly $\mathcal{NP}$-hard due to Lenstra et al. (1977). Several efficient heuristic and optimal algorithms have been developed for $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$ in the last decade. See Bulbul et al. (2007), Tanaka and Fujikuma (2008), Sourd (2009), Kedad-Sidhoum and Sourd (2010). Our focus here is to develop an effective set of cost coefficients for the subproblems, and any of the available algorithms in the literature could be used in conjunction with the approach we present. For the computational experiments in Section 4, in some instances we solve the subproblem optimally using a time-indexed formulation, and in some instances we solve the subproblem heuristically using the algorithm of Bulbul et al. (2007). The basis of this approach is constructing good operation processing sequences from a tight preemptive relaxation of $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$. We note that it is possible to extend this preemptive lower bound to a general piecewise linear and convex E/T cost function with multiple pieces on either side of the due date. Thus, if one opts for constructing the actual operation cost functions explicitly at the expense of extra computational burden, it is possible to extend the algorithm of Bulbul et al. (2007) to solve the resulting subproblems.

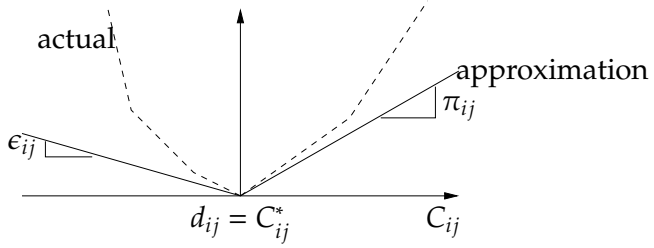


Figure 2: Effect of moving a single operation on the overall objective.

Also, an additional difficulty might arise at each iteration of the algorithm. We observe that when the set of disjunctive arcs in the graph $G'(N, A_C \cup A_D^S)$ is empty, then no path exists between any two operations o_{ik} and o_{ij} on a machine $i \in \mathcal{M}$. However, as we add disjunctive arcs to G', we may create paths between some operations of a currently unscheduled machine $i \notin \mathcal{M}^S$. In particular, a path from node o_{ik} to o_{ij} indicates a lower bound on the amount of time that must elapse between the starting times of these two operations. This type of path is an additional constraint on the final schedule, and is referred to as a delayed precedence constraint (DPC). Rather than explicitly incorporate these DPC's into our subproblem definition, we check for directed cycles while updating G', since violated DPC's imply cycles in the updated graph. If necessary, we remove cycles by applying local changes to the sequence of the current bottleneck machine.

We conclude this section with some comments on classical job shop scheduling problems with regular objective functions, such as $Jm//C_{\max}$, $Jm//\sum_j w_j C_j$, and $Jm//\sum_j w_j T_j$. The cost coefficients in (19) measure the marginal effect of moving operation o_{ij} earlier or later. The former is clearly zero for any regular objective function. Furthermore, π_{ij} is also zero if the job completion time is not affected by a marginal delay in the completion time of o_{ij}. Thus, SB-TPD may be ineffective for the classical objectives in the literature. The true benefits of our solution framework are only revealed when operation completion times have a direct impact on the total cost. Furthermore, for regular objectives, the task of estimating the actual piecewise linear operation cost functions is accomplished easily by longest path calculations in the disjunctive graph. Of course, solving the resulting single-machine subproblems with a general piecewise linear and convex weighted tardiness objective is a substantially harder task. Bulbul (2011) formalizes these concepts and develops a hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem by generalizing the algorithm of Bulbul et al. (2007) for solving the subproblems as discussed above.

3.3.2 Selecting the Bottleneck Machine As alluded to above, at each iteration of the algorithm, we solve the single-machine problem described above **for each of the remaining unscheduled machines**, and select the one with the highest corresponding subproblem objective value to be the current bottleneck machine i^b. Then, the disjunctive graph and the optimal timing problem are updated accordingly to include the machine capacity constraints of this machine where the sequence of operations on i^b are determined by the solution of the corresponding subproblem.

3.3.3 Rescheduling The last step of an iteration of SB-TPD is re-evaluating the schedules of the previously scheduled machines in $\mathcal{M}^S$ given the operation processing sequence on the current bottleneck machine i^b. It is generally observed that SB algorithms without a rescheduling step perform rather poorly (Demirkol et al. (1997)). We perform a classical rescheduling step, such as that in Pinedo and Singer (1999). For each machine $i \in \mathcal{M}^S$, we first delete the corresponding disjunctive arcs from the set A_D^S and construct a subproblem for machine i based on the solution of the optimal timing problem (**TTJm**)$(\mathcal{M}^S \setminus \{i\} \cup \{i^b\})$. Then, machine i is re-scheduled according to the sequence obtained from the subproblem by adding back the corresponding disjunctive arcs to A_D^S. The rescheduling procedure may be repeated several times until no further improvement in the overall objective is achieved.

3.3.4 Tree Search SB-TPD as outlined up until here terminates in m iterations with a single feasible schedule for (**Jm**) by scheduling one additional machine at each iteration. However, it is widely accepted in the literature that constructing multiple feasible schedules by picking different orders in which the machines are scheduled leads to substantially improved solution quality. This is typically accomplished by setting up a partial enumeration tree that conducts a search over possible orders of scheduling the machines. (See, for instance, Adams et al. (1988) and Pinedo and Singer (1999)). Each node in this enumeration tree corresponds to an *ordered* set $\mathcal{M}^S$ that specifies the order of scheduling the machines. The basic idea is to rank the machines in $\mathcal{M} \setminus \mathcal{M}^S$ in non-increasing order of their respective subproblem objective function values and create a child node for the β_l most critical machines in $\mathcal{M} \setminus \mathcal{M}^S$, where $l = \mid \mathcal{M}^S \mid$. Thus, an m-dimensional vector $\beta = (\beta_0, \ldots, \beta_{m-1})$ prescribes the maximum number of children at each level of the tree. This vector provides us with a direct mechanism to trade-off solution time and quality. Our solution approach incorporates no random components, and we can expand the search space with the hope of identifying progressively better solutions by adjusting β appropriately. For more details and a discussion of the fathoming rule that further restricts the size of the search tree, the reader is referred to Bulbul (2011).

4. Computational Experiments The primary goal of our computational study is to demonstrate that the proposed solution approach is general enough that it can produce good quality solutions to different types of job shop scheduling problems. To this end, we consider three special cases of ($\widehat{\textbf{Jm}}$). In all cases, the fundamental insight is that SB-TPD performs quite well, and in particular, its performance relative to that of alternative approaches improves significantly as the percentage of the total cost attributed to inventory holding costs grows.

In Section 4.1, $\gamma = 0$ and we solve a job shop total weighted E/T problem with intermediate holding costs. For small 4×10 ($m \times n$) instances, we illustrate the performance of the algorithm in an absolute sense by benchmarking it against a time-indexed (TI) formulation of the problem (see Dyer and Wolsey (1990)). However, directly solving the TI formulation is impractical (and often, impossible) for larger instances. As there are no directly competing viable algorithm in the literature, we follow a different path to assess the performance of our algorithm on larger 10×10 instances. We consider 22 well-known job shop total weighted tardiness instances due to Pinedo and Singer (1999) and modify them as necessary. In particular, the unit inventory holding costs h_{ij}, $i = 2, \ldots, m_j$, including the unit earliness cost ϵ_j that represents the finished goods inventory holding cost per unit time, are non-decreasing for a job j through processing stages, and the unit tardiness cost π_j is larger than ϵ_j. Depending on the magnitude of π_j relative to the other cost parameters and the tightness of the due dates, we would expect that a good schedule constructed specifically for the job shop total weighted tardiness problem does also perform well under the presence of inventory holding costs in addition to tardiness penalties. Thus, for 10×10 instances we compare the performance of SB-TPD against those of algorithms specifically designed for the job shop total weighted tardiness problem. This instance generation mechanism ensures a fair comparison. In Sections 4.2.1 and 4.3.1, we utilize a similar approach to assess the performance of the algorithm for the job shop total weighted completion time and makespan minimization problems with

intermediate inventory holding costs, respectively.

The results reported in Section 4.1.1 for the TI formulation are obtained by `IBM ILOG OPL Studio 5.5` running on `IBM ILOG CPLEX 11.0`. The algorithms we developed were implemented in Visual Basic (VB) under Excel. The optimal timing problem ($\widehat{\mathbf{TTJm}}$) and the preemptive relaxation of the single-machine subproblem $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$ formulated as a transportation problem as described by Bulbul et al. (2007) are solved by `IBM ILOG CPLEX 9.1` through the VB interface provided by the `IBM ILOG OPL 3.7.1` Component Libraries. All runs were completed on a single core of an HP Compaq DX 7400 computer with a 2.40 GHz Intel Core 2 Quad Q6600 CPU and 3.25 GB of RAM running on Windows XP. The ease and speed of development is the main advantage of the Excel/VB environment. However, we note that an equivalent C/C++ implementation would probably be several times faster. This point should be taken into account while evaluating the times reported in our study.

4.1 Job Shop Total Weighted E/T Problem with Intermediate Inventory Holding Costs

4.1.1 Benchmarking against the TI formulation As mentioned above, for benchmarking against the TI formulation of ($\widehat{\mathbf{Jm}}$), we created 10 instances of size 4×10. All jobs visit all machines in random order. The processing times are generated from an integer uniform distribution $U[1, 10]$. For jobs that start their processing on machine i, the ready times are distributed as integer $U[0, P_i]$, where P_i refers to the sum of the processing times of the first operations to be performed on machine i. Then, the due date of job j is determined as $d_j = r_j + \lfloor f \sum_{i=1}^{m_j} p_{ij} \rfloor$, where f is the due date tightness factor. For each job, the inventory holding cost per unit time at the first stage of processing is distributed as $U[1, 10]$. At subsequent stages, the inventory holding cost per unit time is obtained by multiplying that at the immediately preceding stage by a uniform random number $U[100, 150]$%. The tardiness cost per unit time, π_j, is distributed as $U[100, 200]$% times ϵ_j. For each instance, the due date tightness factor is varied as $f = 1.0, 1.3, 1.5, 1.7, 2.0$, yielding a total of 50 instances. Experimenting with different values of f while keeping all other parameters constants allows us to observe the impact of increasing slack in the schedule. Another 50 instances are generated by doubling the unit tardiness cost for all jobs in a given instance.

In the TI formulation of ($\widehat{\mathbf{Jm}}$), the binary variable x_{ijt} takes the value 1 if o_{ij} completes processing at time t. The machine capacity constraints are formulated as described by Dyer and Wolsey (1990), and for modeling the remaining constraints (2), (3), (8), we represent C_{ij} by $\sum_t t x_{ijt}$. A time limit of 7,200 seconds (2 hours) is imposed on the TI formulation, and the best incumbent solution is reported if the time limit is exceeded without a proven optimal solution.

For the tree search, $\beta = (3, 3, 2, 1)$ and at most 18 feasible schedules are constructed for ($\widehat{\mathbf{Jm}}$) in the partial enumeration tree (see Section 3.3.4). At each node of the tree, we perform rescheduling for up to three full cycles. We do two experiments for each of the 100 instances. In the first run, the single-machine subproblems are solved optimally by a conventional TI formulation ("SB-TPD-OptimalSubprob"), and then in the second run, we only seek a good feasible solution in the subproblems by adopting the approach of Bulbul et al. (2007) ("SB-TPD-HeuristicSubprob").

The results of our experiments are summarized in Tables 1 and 2. The instance names are listed in the first column of Table 1. In the upper half of this table, we report the results for the first 50 instances, where π_j is determined as $U[100, 200]$% times ϵ_j. Re-solving each instance after doubling π_j for all jobs in a given instance yields the results in the bottom half of the table. The objective function values associated with the optimal/best incumbent solutions from the TI formulation appear in columns 2-6 as a function of the due date tightness factor f. Applying SB-TPD by solving the subproblems optimally provides us with the objective function values in columns 7-11, and the percentage gaps with respect to the TI formulation are calculated in columns 12-16. A gap is negative if SB-TPD-OptimalSubprob returns a better solution than the TI formulation. The corresponding results obtained by solving the subproblems

Table 1: Benchmarking against the TI formulation† on job shop E/T instances with intermediate inventory holding costs.

	$\pi_j \sim \epsilon_j \cdot \mathbf{U}(100,200)\%$																								
	Time-Indexed (TI)					**SB-TPD-OptimalSubprob**										**SB-TPD-HeuristicSubprob**									
	OFV					**OFV**					**Gap to TI(%)**					**OFV**					**Gap to TI(%)**				
$f =$	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**
Jm_1	4295	3133	2474	**2038***	1962	4383	3457	2568	2065	1954	2.1	10.3	3.8	1.3	-0.4	4568	3665	2773	2098	2044	6.4	17.0	12.1	2.9	4.2
Jm_2	4087	3170	2487	**2211***	**1968***	4109	3568	2752	2223	1968	0.5	12.6	10.6	0.6	0.0	4509	3232	2616	2231	2052	10.3	2.0	5.2	0.9	4.3
Jm_3	2857	**2045***	**1652***	**1482***	**1508***	2922	2199	1702	**1482***	1528	2.3	7.5	3.0	0.0	1.4	3014	2200	1698	1486	1517	5.5	7.6	2.8	0.3	0.6
Jm_4	**2308***	**1678***	**1394***	**1197***	**1233***	2574	1877	1400	**1197***	1236	11.5	11.9	0.4	0.0	0.3	2705	1970	1397	1334	1366	17.2	17.4	0.2	11.5	10.8
Jm_5	**4365***	**3351***	**2844***	**2578***	2756	4630	3428	**2844***	**2578***	2484	6.1	2.3	0.0	0.0	-9.9	4486	3490	3098	2612	2583	2.8	4.1	8.9	1.3	-6.3
Jm_6	4034	2917	2243	**1999***	2014	4607	3148	2584	2037	2081	14.2	7.9	15.2	1.9	3.3	4647	3399	2271	2087	2102	15.2	16.5	1.2	4.4	4.4
Jm_7	3195	**2072***	2242	2214	2077	3214	**2072***	2266	1993	2181	0.6	0.0	1.1	-10.0	5.0	3317	2362	2222	2205	2187	3.8	14.0	-0.9	-0.4	5.3
Jm_8	**2530***	**1765***	**1431***	1571	**1568***	**2530***	**1765***	1470	1697	1755	0.0	0.0	2.7	8.0	11.9	2946	1856	1704	1677	1868	16.4	5.2	19.1	6.8	19.1
Jm_9	**2734***	2237	1779	**1488***	1448	2851	2241	1923	1544	1425	4.3	0.2	8.1	3.7	-1.6	2923	2186	1841	1664	1563	6.9	-2.3	3.5	11.8	7.9
Jm_10	**3081***	**2119***	2044	**1714***	1676	3282	**2119***	1973	**1714***	1554	6.5	0.0	-3.4	0.0	-7.3	**3081***	**2119***	2100	1750	1577	0.0	0.0	2.8	2.1	-5.9
Avg.	6.4	9.0	10.4	26.8	10.8					**Avg.**	4.8	5.3	4.2	0.6	0.3					**Avg.**	8.5	8.1	5.5	4.2	4.5
Med.	6.3	9.7	5.8	26.8	9.1					**Med.**	3.3	4.9	2.9	0.3	0.1					**Med.**	6.6	6.4	3.1	2.5	4.3
Min	3.4	3.4	3.1	6.6	0.3					**Min**	0.0	0.0	-3.4	-10.0	-9.9					**Min**	0.0	-2.3	-0.9	-0.4	-6.3
Max	9.7	13.2	22.0	47.0	24.1					**Max**	14.2	12.6	15.2	8.0	11.9					**Max**	17.2	17.4	19.1	11.8	19.1
	$\pi_j \sim \epsilon_j \cdot \mathbf{U}(200,400)\%$																								
Jm_1	**7115***	5232	**3888***	**2856***	2347	7750	5571	4093	2944	2340	8.9	6.5	5.3	3.1	-0.3	8265	6158	4315	2957	2347	16.2	17.7	11.0	3.5	0.0
Jm_2	7155	5506	3712	2960	**2158***	7209	5726	4222	2956	2158	0.8	4.0	13.7	-0.1	0.0	8513	5260	4399	3024	2181	19.0	-4.5	18.5	2.2	1.1
Jm_3	5206	**3366***	2496	**1981***	**1694***	5228	3587	2574	2048	**1694***	0.4	6.6	3.1	3.4	0.0	5361	3626	2611	2028	1728	3.0	7.7	4.6	2.4	2.0
Jm_4	**4052***	**2684***	**2016***	**1533***	**1300***	4517	2995	2216	1621	1319	11.5	11.6	10.0	5.8	1.5	4737	2812	2251	1764	1491	16.9	4.8	11.6	15.1	14.7
Jm_5	**7747***	**5564***	**4394***	**3693***	3702	8176	5632	**4394***	**3693***	3175	5.5	1.2	0.0	0.0	-14.2	7929	5685	4758	3760	3358	2.3	2.2	8.3	1.8	-9.3
Jm_6	**6982***	5520	**3298***	**2537***	**2285***	7740	5272	3762	2669	2336	10.8	-4.5	14.1	5.2	2.2	7401	5600	**3298***	2690	2354	6.0	1.4	0.0	6.0	3.0
Jm_7	5547	**3219***	3099	2657	**2272***	5629	**3219***	3313	2609	2435	1.5	0.0	6.9	-1.8	7.2	5839	3765	3353	2609	2470	5.3	17.0	8.2	-1.8	8.7
Jm_8	**4418***	**2764***	**1900***	**1925***	**1610***	4506	**2764***	**1900***	2162	**1610***	2.0	0.0	0.0	12.3	0.0	4838	2919	1980	2236	1849	9.5	5.6	4.2	16.2	14.8
Jm_9	**4842***	**3560***	3136	2623	**1674***	5048	3718	3008	2192	**1674***	4.3	4.4	-4.1	-16.5	0.0	5183	3664	3210	2404	1750	7.0	2.9	2.3	-8.4	4.5
Jm_10	**5417***	**3493***	2871	2440	1927	**5417***	**3493***	2976	2327	1791	0.0	0.0	3.7	-4.6	-7.1	**5417***	**3493***	2976	2327	1814	0.0	0.0	3.7	-4.6	-5.8
Avg.	5.4	19.1	9.2	22.6	26.5					**Avg.**	4.6	3.0	5.3	0.7	-1.1					**Avg.**	8.5	5.5	7.2	3.2	3.4
Med.	5.0	22.2	4.9	16.3	30.4					**Med.**	3.1	2.6	4.5	1.5	0.0					**Med.**	6.5	3.9	6.4	2.3	2.5
Min	0.9	8.1	4.0	9.2	12.1					**Min**	0.0	-4.5	-4.1	-16.5	-14.2					**Min**	0.0	-4.5	0.0	-8.4	-9.3
Max	10.3	27.0	27.7	48.4	37.1					**Max**	11.5	11.6	14.1	12.3	7.2					**Max**	19.0	17.7	18.5	16.2	14.8

† The time limit is 7200 seconds.

* Optimal solution.

heuristically are specified in columns 17-26. Optimal solutions in the table are designated with a '*' and appear in bold. The average, median, minimum, and maximum percentage gaps are computed in rows labeled with the headers "Avg.," "Med.," "Min," and "Max," respectively. For columns 2-6, these statistics are associated with the optimality gaps of the incumbent solutions reported by CPLEX at the time limit. Table 2 presents statistics on the CPU times until the best solutions are identified for SB-TPD-OptimalSubprob and SB-TPD-HeuristicSubprob.

The TI formulation terminates with an optimal solution in 59 out of 100 cases. Among these 59 cases, SB-TPD-OptimalSubprob and SB-TPD-HeuristicSubprob identify 19 and 5 optimal solutions, respectively. Over all 100 instances, the solution gaps of SB-TPD-OptimalSubprob and SB-TPD-HeuristicSubprob with respect to the optimal/incumbent solution from the TI formulation are 2.75% and 5.86%, respectively. We achieve these optimality gaps in just 31.9 and 3.1 seconds on average with SB-TPD-OptimalSubprob and SB-TPD-HeuristicSubprob, respectively. We therefore conclude that the subproblem definition properly captures the effect of the new sequencing decisions on the currently unscheduled machines, and that SB-TPD yields excellent feasible solutions to this difficult job shop scheduling problem in short CPU times. We observe that SB-TPD-OptimalSubprob is about an order of magnitude slower than the SB-TPD-HeuristicSubprob. Based on the quality/time trade-off, we opt for solving the subproblems heuristically in the rest of our computational study.

For all algorithms, the objective values are almost always non-increasing as a function of $f = 1.0, 1.3, 1.5, 1.7$. For f large enough, tardiness costs are virtually eliminated, and increasing f further leads to an increase in the objective function value. Therefore, we occasionally observe that for some problem instances the objective increases from $f = 1.7$ to $f = 2.0$. Furthermore, the performance of the SB-TPD variants improves significantly as f increases. This may partially be attributed to the relatively lower quality of the incumbent solutions for large f values. The optimality gaps reported by CPLEX for incumbents at termination tend to grow with f. Note that larger f values imply longer planning horizons and increase the size of the TI formulation. As a final remark, doubling the unit tardiness costs does not lead to a visible pattern in solution quality for the SB-TPD variants.

Table 2: CPU time statistics (in seconds) for the results in Table 1.

	$\pi_j \sim \epsilon_j \cdot \mathbf{U(100, 200)}\%$									
	SB-TPD-OptimalSubprob					**SB-TPD-HeuristicSubprob**				
$f =$	**1.0**	**1.3**	**1.5**	**1.7**	**2.0**	**1.0**	**1.3**	**1.3**	**1.7**	**2.0**
Avg.	43.0	19.8	23.6	29.9	41.5	2.8	2.3	3.7	2.8	3.0
Med.	39.5	9.0	10.1	19.9	38.5	2.1	1.4	3.3	2.5	2.5
Min	4.0	4.0	4.5	3.6	4.9	0.4	0.4	0.4	1.0	0.7
Max	81.7	79.2	67.1	84.6	81.6	7.2	6.4	7.4	6.5	6.3
	$\pi_j \sim \epsilon_j \cdot \mathbf{U(200, 400)}\%$									
Avg.	45.4	26.7	26.9	26.4	35.8	3.6	2.9	3.3	2.9	4.2
Med.	50.0	14.0	29.8	14.8	35.0	2.9	1.7	3.0	2.1	3.8
Min	3.9	4.1	3.7	3.6	3.7	0.3	0.5	0.3	0.6	0.5
Max	86.8	91.8	48.6	61.0	76.2	8.8	6.5	8.0	6.8	7.6

4.1.2 Benchmarking Against Heuristics As we mentioned at the beginning of Section 4, the major obstacle to demonstrating the value of our heuristic for large problem instances is the lack of directly competing algorithms in the literature. To overcome this, we pursue an unconventional path. Instead of simply benchmarking against a set of dispatch rules, we adopt a data generation scheme that is tailored toward algorithms specifically developed for the job shop total weighted tardiness problem (JS-TWT). In particular, we suitably modify 22 well-known standard benchmark instances originally proposed for $Jm//C_{\max}$ for our problem. Note that this same set of instances were adapted to JS-TWT by Pinedo and Singer (1999) and are commonly used for benchmarking in papers focusing on JS-TWT, such as Pinedo and Singer (1999), Kreipl (2000), Bulbul (2011). In the original 10×10 makespan instances, all jobs visit all machines, all ready times are zero, and the processing times are distributed between 1 and 100. For our purposes, all processing times are scaled as $p_{ij} \leftarrow \lceil p_{ij}/10 \rceil, j = 1, \ldots, n, i = 1, \ldots, m_j$, in order to reduce the total computational burden because the effort required in the approach adopted for

solving the subproblems depends on the sum of the processing times. (Recall that our goal in this paper is to develop an effective set of cost coefficients for the subproblems, so that we could have employed other algorithms in the literature that do not have this limitation for solving the subproblems.) The due dates and the inventory holding, earliness, and tardiness costs per unit time are set following the scheme described in Section 4.1. Two levels of the unit tardiness costs and five values of f for each makespan instance yield a total of 220 instances for our problem. As we observe later in this section, under tight due dates the majority of the total cost is due to the tardiness of the jobs, and we expect that good schedules constructed specifically for minimizing the total weighted tardiness in these instances also perform well in the presence of intermediate inventory holding and earliness costs in addition to tardiness penalties. In other words, we have specifically designed an instance generation mechanism to ensure a fair comparison.

We use these instances to demonstrate that our (non-objective-specific) heuristic SB-TPD fares quite well against state-of-the-art algorithms developed for JS-TWT for small values of f. On the other hand, as more slack is introduced into the schedule by setting looser due dates and holding costs become increasingly more significant, our approach dominates alternative approaches.

A total of five different algorithms are run on each instance. We apply SB-TPD by solving the subproblems heuristically. We test this against the large-step random walk local search algorithm ("LSRW") by Kreipl (2000) and the SB heuristic for JS-TWT ("SB-WT") due to Pinedo and Singer (1999). Both of these algorithms generate very high quality solutions for JS-TWT. In general, LSRW performs better than SB-WT. These observations are based on the original papers and are also verified by our computational testing in this section. We note that Pinedo and Singer (1999) and Kreipl (2000) demonstrate the performance of their algorithms on the same 22 benchmark instances considered here, except that they consider a different tardiness cost structure and set $f = 1.3, 1.5, 1.6$. Preliminary runs indicated that the LSRW generally improves very little after 120 seconds of run time. Thus, the time limit for this algorithm is set to 120 seconds. Due to the probabilistic nature of this algorithm, we run it 5 times for each instance and report the average objective function value. We also run the general purpose SB algorithm ("Gen-SB") by Asadathorn (1997) that also supports a variety of objectives. Finally, we construct a schedule using the Apparent Tardiness Cost ("ATC") dispatch rule proposed for JS-TWT by Vepsalainen and Morton (1987). The scaling parameter for the average processing time in this rule is set to 4 for $f = 1.0, 1.3$, to 3 for $f = 1.5$, and to 2 for $f = 1.7, 2.0$. For these settings, see Vepsalainen and Morton (1987), Kutanoglu and Sabuncuoglu (1999). These last four algorithms are all implemented in LEKIN®- Flexible Job-Shop Scheduling System (2002) which allows us to easily test these algorithms in a stable and user-friendly environment. For these algorithms, we first solve JS-TWT by ignoring the inventory holding and earliness costs in a given instance. Then, we compute the corresponding objective value for the job shop E/T problem with intermediate holding costs by applying the earliness, tardiness and intermediate inventory holding costs to the constructed schedule. The results are presented in Table 3.

Table 3: Results for the job shop total weighted E/T instances with intermediate inventory holding costs.

	$\pi_j \sim \epsilon_j \cdot \mathbf{U(100,200)}\%$						$\pi_j \sim \epsilon_j \cdot \mathbf{U(200,400)}\%$					
					f **=1.0**							
	B-WT /	**Gap to B-OFV(%)**					**B-WT /**	**Gap to B-OFV(%)**				
	B-OFV(%)	**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**	**B-OFV(%)**	**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**
Avg.	87.5	10.4	0.1	9.4	43.0	44.9	93.3	9.0	0.3	9.9	41.9	45.0
Med.	87.4	9.8	0.0	9.1	42.6	42.0	93.3	9.4	0.0	8.0	40.3	42.9
Min	83.9	0.0	0.0	0.0	4.1	16.4	91.2	0.0	0.0	0.0	3.8	14.7
Max	90.1	28.3	2.0	26.6	77.8	99.3	95.2	17.4	6.1	25.7	74.4	99.2
					f **=1.3**							
	B-WT /	**Gap to B-OFV(%)**					**B-WT /**	**Gap to B-OFV(%)**				
	B-OFV(%)	**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**	**B-OFV(%)**	**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**
Avg.	54.1	12.1	1.0	10.7	70.6	123.0	69.6	14.6	0.5	11.5	80.1	147.4
Med.	54.2	11.9	0.0	10.6	54.6	111.1	70.2	12.3	0.0	10.7	67.3	133.4
Min	34.1	0.0	0.0	0.0	27.0	51.0	50.9	0.0	0.0	0.0	36.5	64.8

	$\pi_j \sim \epsilon_j \cdot \mathbf{U}(100,200)\%$						$\pi_j \sim \epsilon_j \cdot \mathbf{U}(200,400)\%$					
Max	73.8	37.2	10.9	30.6	203.7	293.6	85.0	40.8	4.0	38.8	276.6	397.0
					f =**1.5**							
	B-WT / B-OFV(%)	**Gap to B-OFV(%)**					**B-WT / B-OFV(%)**	**Gap to B-OFV(%)**				
		SB-TPD	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**		**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**
Avg.	13.0	2.3	16.2	21.8	57.9	142.2	20.4	7.2	7.3	14.3	78.0	196.2
Med.	9.7	0.0	12.5	16.0	51.2	138.5	16.4	3.4	1.2	9.7	61.0	184.9
Min	0.0	0.0	0.0	0.0	10.0	71.3	0.0	0.0	0.0	0.0	13.3	96.1
Max	40.0	21.2	63.2	70.5	141.2	293.6	56.4	42.5	55.5	63.2	212.1	475.8
					f =**1.7**							
	B-WT / B-OFV(%)	**Gap to B-OFV(%)**					**B-WT / B-OFV(%)**	**Gap to B-OFV(%)**				
		SB-TPD	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**		**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**
Avg.	1.3	0.0	52.7	54.3	51.2	127.2	2.1	0.0	44.9	47.1	59.4	165.5
Med.	0.0	0.0	48.8	55.0	47.4	119.5	0.0	0.0	38.6	51.5	52.7	150.8
Min	0.0	0.0	19.8	17.2	22.8	57.4	0.0	0.0	5.3	6.3	24.1	92.4
Max	12.1	0.0	90.6	99.3	144.4	282.6	21.3	0.0	96.8	92.0	150.3	350.8
					f =**2.0**							
	B-WT / B-OFV(%)	**Gap to B-OFV(%)**					**B-WT / B-OFV(%)**	**Gap to B-OFV(%)**				
		SB-TPD	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**		**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**ATC**
Avg.	0.0	0.0	97.6	116.8	55.6	149.7	0.0	0.0	89.3	106.3	51.5	155.1
Med.	0.0	0.0	99.0	110.3	47.8	132.8	0.0	0.0	77.4	103.9	48.3	136.9
Min	0.0	0.0	33.6	68.4	30.9	76.7	0.0	0.0	30.1	45.6	23.0	88.3
Max	0.0	0.0	245.5	240.9	82.6	314.3	0.0	0.0	244.5	190.2	80.2	373.8

For each instance, we calculate the best objective function value ("B-OFV") obtained over five alternate algorithms, and all gaps in Table 3 are calculated with respect to the best available solutions. Furthermore, in order to justify our benchmarking strategy against algorithms developed for JS-TWT we compute the minimum total weighted tardiness cost ("B-TWT") over all algorithms applied to an instance, and we report statistics on the ratio of B-TWT to B-OFV in the first column of Table 3. For $f = 1.0, 1.3$, the average of the ratio B-TWT/B-OFV is 90.4% and 61.9%, respectively. Thus, for these instances we expect that the schedules obtained from algorithms designed for JS-TWT perform very well.

For $f = 1.0$, LSRW is the best contender. SB-TPD performs on a par with SB-WT, and both of these algorithms have an average gap of 9-10% from the best available solution. The fact that the tardiness costs dictate the schedule is also reflected in the gaps obtained by considering tardiness costs only. These figures (not reported here) are close to their counterparts with inventory holding and earliness costs. Both Gen-SB and the ATC dispatch rule have average gaps of more than 40% with respect to the best available solution.

For $f = 1.3$, LSRW again outperforms the other algorithms. SB-TBD performs slightly worse than SB-WT. The average gap of SB-TPD is on average 12.1% and 14.6% with respect to the best available solution for instances with small and large tardiness costs, respectively. The corresponding figures for SB-WT are 10.7% and 11.5%, respectively. Gen-SB has an average gap of 70.6% and 80.1% with respect to to the best available solution for instances with small and large tardiness costs, respectively. For ATC, these average gaps are at 123.0% and 147.4%, respectively.

For $f = 1.5$, the average of the ratio B-WT to B-OFV drops to 16.7%. That is, the inventory holding and earliness costs become crucial. In this case, SB-TPD is superior to all other algorithms. For small tardiness costs, the average gaps with respect to the best available solution are 2.3%, 16.2%, and 21.8% for SB-TPD, LSRW, and SB-WT, respectively. The corresponding average gaps for large tardiness costs are obtained as 7.2%, 7.3%, and 14.3%, respectively. The two other algorithms lag by a large margin as for f=1.0 and f=1.3.

For $f = 1.7$ and $f = 2.0$, the tardiness costs can almost always be totally eliminated. For these instances, SB-TPD always produces the best schedule. All other algorithms have average gaps of at least 45% with respect to our algorithm.

In SB-TPD, the time until the best solution identified is 782 seconds on average over all instances with no clear trend in solution times as a function of f or the relative magnitude of the unit tardiness costs to the unit earliness costs. However, on average 68% of this time is spent on calculating the single-machine cost functions which requires inverting the optimal basis of the optimal timing problem in the Excel/VB environment. This time can be eliminated totally if the algorithm is implemented in C/C++ using the corresponding `CPLEX` library which provides direct access to the inverse of the optimal basis. The second main component of the solution time is expended while solving the preemptive relaxation of $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$ as part of the single-machine subproblem and constitutes about 9% of the total solution time. On the other hand, the time required by `CPLEX` for solving the optimal timing problems is only about 4% of the total time on average. Clearly, SB-TPD has great potential to provide excellent solutions in short CPU times. Furthermore, by pursuing the different branches of the search tree on different processors, SB-TPD can be parallelized in a straightforward manner and the solution times may be reduced further.

In general, we expect to obtain high-quality solutions early during SB-TPD if the subproblem definition is appropriate and the associated solution procedure is effective. In Appendix C, we present a detailed analysis of the rate at which good incumbent solution are found and improved. In general, our procedure finds very good solutions (and often the best solutions found) early during the heuristic run.

4.2 Job Shop Total Weighted Completion Time Problem with Intermediate Inventory Holding Costs

4.2.1 Benchmarking Against Heuristics The instances in the previous section are converted into total weighted completion time instances by setting the due dates to zero. The same set of algorithms are applied to the resulting 44 instances, except that the ATC dispatch rule is substituted by the Weighted Shortest Processing Time (WSPT) dispatch rule which is more appropriate for weighed completion time problems. Note that the WSPT rule implemented in LEKIN®- Flexible Job-Shop Scheduling System (2002) computes the priority of an operation o_{ij} by taking into account the total remaining processing time of job j. The results are presented in Table 4.

Table 4: Results for the job shop total weighted completion time instances with intermediate inventory holding costs.

	$\pi_j \sim \epsilon_j \cdot \mathbf{U(100,200)}\%$						$\pi_j \sim \epsilon_j \cdot \mathbf{U(200,400)}\%$					
	B-WC / B-OFV(%)	**Gap to B-OFV(%)**					**B-WC / B-OFV(%)**	**Gap to B-OFV(%)**				
		SB-TPD	**LSRW**	**SB-WT**	**Gen-SB**	**WSPT**		**SB-TPD**	**LSRW**	**SB-WT**	**Gen-SB**	**WSPT**
Avg.	96.7	2.0	0.4	1.8	10.8	10.7	98.3	1.8	0.4	1.8	9.9	10.1
Med.	96.6	1.9	0.0	1.5	9.8	10.2	98.3	1.8	0.0	1.5	8.1	9.5
Min	95.6	0.0	0.0	0.0	2.3	3.4	97.7	0.0	0.0	0.0	3.2	3.2
Max	97.9	6.7	3.3	5.6	24.6	21.7	98.9	3.9	2.2	6.6	23.7	20.8

As in Section 4.1.2, we calculate the best objective function value ("B-OFV") obtained over five alternate algorithms for each instance, and the gaps reported in Table 4 are based on the best available solutions. Statistics on the ratio of the minimum total weighted completion time cost ("B-WC") over all algorithms to B-OFV are provided in the first column of Table 4 and justify our benchmarking strategy. For all instances, the ratio B-WC / B-OFV stands above 95%. The job shop total weighted completion time problem with inventory holding costs appears to be easier in practice compared to its counterpart with tardiness costs. SB-WT and SB-TPD perform on a par, while LSRW exhibits slightly better gaps. For small unit completion time (tardiness) costs, the average gaps with respect to the best available solution are 2.0%, 0.4%, and 1.8% for SB-TPD, LSRW, and SB-WT, respectively. Doubling the unit completion time costs leads to the average solution gaps 1.8%, 0.4%, and 1.8% for these three algorithms, respectively. The two other algorithms Gen-SB and WSPT are on average about 10% off the best available solution.

For the total weighted completion time instances with inventory holding costs, SB-TPD takes an average of 817 seconds until the best solution is identified with a similar composition to that in Section 4.1.2. SB-TPD is very effective for the job shop total weighted completion time problem with inventory

holding costs. In Appendix C, we again observe that we typically find good (and often the best found) solutions early in the heuristic run.

4.3 Job Shop Makespan Problem with Intermediate Inventory Holding Costs

4.3.1 Benchmarking Against Heuristics The instances in this section are identical to the corresponding total weighted completion time instances except that all unit completion time costs are set to zero and an appropriate unit cost for $C_{\max}$ is assigned in each instance. In order to determine this cost parameter γ in the objective function (6) of $(\widehat{\mathbf{Jm}})$, we first run LSRW on a given instance once in order to minimize $C_{\max}$, record the resulting makespan and compute the associated total inventory holding cost. Then, for each instance with small unit completion time costs ($\pi_j \sim \epsilon_j \cdot U(100,200)\%$) we set γ so that the total cost $C(C_{\max})$ due to the makespan is 50% of the total cost. The same procedure is repeated for total weighted completion time instances with large unit completion time costs ($\pi_j \sim \epsilon_j \cdot U(200,400)\%$), and γ is determined so that 90% of the total cost is attributed to $C_{\max}$. Thus, we create a total of 44 instances for the job shop makespan problem with intermediate inventory holding costs.

SB-TPD, LSRW, Gen-SB, and the Longest Processing Time (LPT) dispatch rule are applied to each instance. The original paper Kreipl (2000) solves well-known "hard" instances of $Jm//C_{\max}$ by LSRW and achieves near-optimal solutions. This is the reason why LSRW is the algorithm of choice in setting the γ values as described above. The implementation of the LPT dispatch rule in LEKIN®- Flexible Job-Shop Scheduling System (2002) takes into account the processing times of all remaining operations of the associated job while determining the priority of an operation. Therefore, this dispatch rule becomes equivalent to the Most Work Remaining Rule (MWKR) in the job shop environment which has been demonstrated to work well for the job shop makespan problem in the literature (see Demirkol et al. (1997) and Chang et al. (1996) for details). The results are depicted in Table 5.

Table 5: Results for the job shop makespan instances with intermediate inventory holding costs.

	$C(C_{\max})$ / **OFV ≈ 0.50 for LSRW**					$C(C_{\max})$ / **OFV ≈ 0.90 for LSRW**				
	$C(C_{\max})$ / **OFV(%)**	**Gap to B-OFV(%)**				$C(C_{\max})$ / **OFV(%)**	**Gap to B-OFV(%)**			
		SB-TPD	**LSRW**	**Gen-SB**	**LPT**		**SB-TPD**	**LSRW**	**Gen-SB**	**LPT**
Avg.	48.4	0.0	25.1	48.1	69.7	89.4	7.5	0.0	11.6	24.3
Med.	49.5	0.0	25.8	43.7	61.4	89.8	7.3	0.0	9.1	23.3
Min	41.3	0.0	6.0	15.7	42.7	86.4	0.0	0.0	4.3	12.0
Max	52.2	0.0	38.2	110.4	134.5	90.8	11.8	0.4	35.0	48.2

In the first column of Table 5, we report statistics on the percentage of the total cost attributed to the makespan for the schedules produced by LSRW which is the best competing algorithm from the literature. This assures us of a fair comparison of SB-TPD to the other algorithms considered. For half of the instances, the total inventory holding cost is approximately 10% of the total cost for LSRW. Thus, for these instances we expect that the schedules obtained by LSRW perform very well. For the remaining instances, this ratio – referred to as the I/T ratio below – is about 50%. If the I/T ratio is about 10%, LSRW has the best performance. In this case, the average gaps with respect to the best available solutions are 7.5%, 0.0%, 11.6%, and 24.3% for SB-TPD, LSRW, Gen-SB, and LPT, respectively. If the I/T ratio is increased to about 50%, SB-TPD has the best performance for all instances. In this case, the corresponding average gaps with respect to the best available solutions are obtained as 0.0%, 25.1%, 48.1%, and 69.7%.

SB-TPD takes an average of 704 seconds until the best solution is identified for these instances with a similar composition to those in Sections 4.1.2 and 4.2.1. Once again, in Appendix C, we see that we typically find good (and often the best found) solutions early in the heuristic run.

5. Concluding Remarks We developed a general method to solve job shop scheduling problems with objectives that are a function of both job completion times and intermediate operation completion times. This class of models is growing in importance as considerations such as holding cost reduction and rescheduling become more important, and our approach works on any job shop scheduling problem

with operation completion time-related costs and any objective function for which the optimal timing problem can be expressed as a linear program.

We use a decomposition approach to solve our problem, a variation of the celebrated shifting bottleneck heuristic where the single-machine problems are defined using the dual variables from a linear program to solve the optimal timing problem of partial schedules. To the best of our knowledge, this is the first paper utilizing a linear program combined with a decomposition heuristic in this fashion, and the first shifting-bottleneck-based heuristic that is broadly applicable to a large set of objectives without modification. Our computational study focuses on problems with intermediate holding costs and a variety of objectives, and demonstrates that our approach performs well on problems for which we can determine optimal solutions, and is competitive with existing less general heuristics designed for specific problems and objectives, particularly as holding cost becomes a more significant part of the total cost.

There are several directions in which this research can be extended. The algorithms can be tested in a variety of other settings, with different operation-related costs and objectives. Alternate subproblem solution techniques can be evaluated. It might be possible to analytically bound the performance of this approach. We are encouraged by the performance of our algorithms, and hope that the framework outlined in this paper is adopted and extended by other researchers.

References

Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401.

Asadathorn, N. (1997). *Scheduling of Assembly Type of Manufacturing Systems: Algorithms and Systems Developments*. PhD thesis, Department of Industrial Engineering, New Jersey Institute of Technology, Newark, New Jersey.

Avci, S. and Storer, R. (2004). Compact local search neighborhoods for generalized scheduling. Working paper.

Bulbul, K. (2002). *Just-In-Time Scheduling with Inventory Holding Costs*. PhD thesis, University of California at Berkeley.

Bulbul, K. (2011). A hybrid shifting bottleneck-tabu search heuristic for the job shop total weighted tardiness problem. *Computers & Operations Research*, 38(6):967–983. http://dx.doi.org/10.1016/j.cor.2010.09.015.

Bulbul, K., Kaminsky, P., and Yano, C. (2004). Flow shop scheduling with earliness, tardiness and intermediate inventory holding costs. *Naval Research Logistics*, 51(3):407–445.

Bulbul, K., Kaminsky, P., and Yano, C. (2007). Preemption in single machine earliness/tardiness scheduling. *Journal of Scheduling*, 10(4-5):271–292.

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.

Chang, Y. L., Sueyoshi, T., and Sullivan, R. (1996). Ranking dispatching rules by data envelopment analysis in a jobshop environment. *IIE Transactions*, 28(8):631–642.

Demirkol, E., Mehta, S., and Uzsoy, R. (1997). A computational study of shifting bottleneck procedures for shop scheduling problems. *Journal of Heuristics*, 3(2):111–137.

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.

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.

Jayamohan, M. and Rajendran, C. (2004). Development and analysis of cost-based dispatching rules for job shop scheduling. *European Journal of Operations Research*, 157(2):307–321.

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

Kedad-Sidhoum, S. and Sourd, F. (2010). Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers & Operations Research*, 37(8):1464–1471.

Kreipl, S. (2000). A large step random walk for minimizing total weighted tardiness in a job shop. *Journal of Scheduling*, 3(3):125–138.

Kutanoglu, E. and Sabuncuoglu, I. (1999). An analysis of heuristics in a dynamic job shop with weighted tardiness objectives. *International Journal of Production Research*, 37(1):165–187.

Laha, D. (2007). Heuristics and metaheuristics for solving scheduling problems. In Laha, D. and Mandal, P., editors, *Handbook of Computational Intelligence in Manufacturing and Production Management*, chapter 1, pages 1–18. Idea Group, Hershey, PA.

LEKIN®- Flexible Job-Shop Scheduling System (2002). Version 2.4. http://www.stern.nyu.edu/om/software/lekin/index.htm.

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

Mason, S., Fowler, J., and Carlyle, W. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3):247–262.

Ohta, H. and Nakatanieng, T. (2006). A heuristic job-shop scheduling algorithm to minimize the total holding cost of completed and in-process products subject to no tardy jobs. *International Journal of Production Economics*, 101(1):19–29.

Ovacik, I. M. and Uzsoy, R. (1996). *Decomposition Methods for Complex Factory Scheduling Problems*. Springer.

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.

Singer, M. (2001). Decomposition methods for large job shops. *Computers & Operations Research*, 28(3):193–207.

Sourd, F. (2009). New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, 21(1):167–175.

Tanaka, S. and Fujikuma, S. (2008). An efficient exact algorithm for general single-machine scheduling with machine idle time. In *IEEE International Conference on Automation Science and Engineering, 2008. CASE 2008*, pages 371 –376.

Thiagarajan, S. and Rajendran, C. (2005). Scheduling in dynamic assembly job-shops to minimize the sum of weighted earliness, weighted tardiness and weighted flowtime of jobs. *Computers and Industrial Engineering*, 49(4):463–503.

Vepsalainen, A. P. J. and Morton, T. E. (1987). Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035–1047.

Xhafa, F. and Abraham, A., editors (2008). *Metaheuristics for Scheduling in Industrial and Manufacturing Applications*, volume 128 of *Studies in Computational Intelligence*. Springer.

Appendix A. Development of ϵ_{ij} and π_{ij} Recall that our goal is to demonstrate that the increase in the optimal objective value of (**TTJm**)($\mathcal{M}^S$) due to an additional constraint (17) or (18) can be bounded from below by applying sensitivity analysis to the optimal solution of (**TTJm**)($\mathcal{M}^S$). This analysis provides the unit earliness and tardiness costs of job j in the subproblem for machine i denoted by ϵ_{ij} and π_{ij}, respectively. We assume that the linear program (**TTJm**)($\mathcal{M}^S$) is solved by the simplex algorithm, and an optimal basic sequence $\mathcal{B}$ is available at the beginning of the current iteration.

Before we proceed with the analysis, we note that (**TTJm**)($\mathcal{M}^S$) is an LP in standard form:

$$\min \quad \mathbf{cx} \tag{20}$$
$$\mathbf{Ax} = \mathbf{b} \tag{21}$$
$$\mathbf{x} \geq \mathbf{0} \tag{22}$$

where $\mathbf{A}$ is an $(m \times n)$ matrix of the coefficients of the structural constraints, $\mathbf{c}$ is a $(1 \times n)$ row vector of the objective coefficients, and $\mathbf{b}$ is an $(m \times 1)$ column vector of the right hand side coefficients. Given a basic sequence $\mathcal{B}$ corresponding to the indices of the m basic variables and the associated nonbasic sequence $\mathcal{N}$, we also define $\mathbf{A}_\mathcal{B}$ as the basis matrix, $\bar{\mathbf{x}} = \begin{pmatrix} \bar{\mathbf{x}}_\mathcal{B} \\ \bar{\mathbf{x}}_\mathcal{N} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_\mathcal{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}} \\ \mathbf{0} \end{pmatrix}$ as the basic solution, $\bar{\mathbf{y}} = \mathbf{c}_\mathcal{B}\mathbf{A}_\mathcal{B}^{-1}$ as the vector of dual variables, $\bar{\mathbf{c}} = \mathbf{c} - \bar{\mathbf{y}}\mathbf{A}$ as the vector of reduced costs, and $\bar{\mathbf{A}} = \begin{pmatrix} \bar{\mathbf{A}}_\mathcal{B} & \bar{\mathbf{A}}_\mathcal{N} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_\mathcal{B}^{-1}\mathbf{A}_\mathcal{B} & \mathbf{A}_\mathcal{B}^{-1}\mathbf{A}_\mathcal{N} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \bar{\mathbf{A}}_\mathcal{N} \end{pmatrix}$. An optimal basic sequence $\mathcal{B}$ satisfies the conditions:

$$\mathbf{A}\bar{\mathbf{x}} = \mathbf{b},\ \bar{\mathbf{x}} \geq \mathbf{0} \quad \text{(primal feasibility)}, \tag{23}$$
$$\bar{\mathbf{c}} \geq \mathbf{0} \quad \text{(dual feasibility)}, \tag{24}$$
$$\bar{\mathbf{c}}\bar{\mathbf{x}} = 0 \quad (\text{or } \mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{y}}\mathbf{b}) \quad \text{(complementary slackness)}. \tag{25}$$

In our notation, $\mathbf{Z}_{i.}$ and $\mathbf{Z}_{.j}$ denote the ith row and jth column of a matrix $\mathbf{Z}$, respectively.

Now, assume that (**TTJm**)($\mathcal{M}^S$) is solved by the simplex method, and an optimal basic sequence $\mathcal{B}$ is available along with the optimal operation completion times C^*_{ij}. Then, we either add

$$\begin{aligned} C_{ij} + s_{ij} &= d_{ij} - \delta \\ s_{ij} &\geq 0 \end{aligned} \quad \text{or} \quad \begin{aligned} C_{ij} - s_{ij} &= d_{ij} + \delta \\ s_{ij} &\geq 0 \end{aligned} \tag{26}$$

to this model, where $d_{ij} = C^*_{ij}$. In either case, the model is expanded by one more constraint and one more variable, and the current optimal solution does not satisfy (26) because δ is strictly greater than zero and $s_{ij} \geq 0$ is required. However, we can easily construct a new basic solution to restart the optimization. This basic solution violates primal feasibility (23) while preserving dual feasibility (24) as we show below. Thus, we can continue with the dual simplex method in order to find the next optimal solution. In the remainder of this section, we demonstrate that the first iteration of the dual simplex method can be performed implicitly based on data readily available in the optimal basic solution of (**TTJm**)($\mathcal{M}^S$). In addition, the dual variable associated with (26) obtained from this iteration provides us with the unit earliness or tardiness cost for job j in the single-machine subproblem of machine i.

The updated problem data after adding (26) and s_{ij} are denoted by a prime: $\mathbf{A}' = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{A}_{(m+1).} & 1 \end{pmatrix}$ or $\mathbf{A}' = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{A}_{(m+1).} & -1 \end{pmatrix}$, $\mathbf{b}' = \begin{pmatrix} \mathbf{b} \\ \mathbf{b}_{m+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ d_{ij} - \delta \end{pmatrix}$ or $\mathbf{b}' = \begin{pmatrix} \mathbf{b} \\ \mathbf{b}_{m+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ d_{ij} + \delta \end{pmatrix}$, $\mathbf{c}' = \begin{pmatrix} \mathbf{c} & \mathbf{c}_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{c} & 0 \end{pmatrix}$, where the last column in $\mathbf{A}'$ corresponds to the $(n+1)$st variable s_{ij}, $\mathbf{A}_{(m+1).}$ represents the coefficients of the original variables in the $(m+1)$st constraint (26), $\mathbf{b}_{m+1}$ is the right hand side of (26), and $\mathbf{c}_{n+1}$ is the objective coefficient of s_{ij}. In order to compute a basic solution, we need to associate a basic variable with (26). A natural candidate is the slack/surplus variable s_{ij} which leads to the new basic sequence $\mathcal{B}' = \mathcal{B} \cup \{n+1\}$. Then, the basis matrix is constructed as:

$$\mathbf{A}'_{\mathcal{B}'} = \begin{pmatrix} \mathbf{A}_\mathcal{B} & \mathbf{0} \\ \mathbf{A}_{m+1,\mathcal{B}} & 1 \end{pmatrix} \quad \text{or} \quad \mathbf{A}'_{\mathcal{B}'} = \begin{pmatrix} \mathbf{A}_\mathcal{B} & \mathbf{0} \\ \mathbf{A}_{m+1,\mathcal{B}} & -1 \end{pmatrix}, \tag{27}$$

where the last column corresponds to s_{ij} and $\mathbf{A}_{m+1,\mathcal{B}} = (0\ 0\ \ldots\ 1\ \ldots\ 0\ 0)$ is a $(1 \times m)$ row vector of the coefficients of the basic variables $\mathcal{B}$ in (26). Clearly, the coefficient 1 in $\mathbf{A}_{m+1,\mathcal{B}}$ corresponds to C_{ij} because

the completion time variables are always positive and basic. Assuming that C_{ij} is the jth basic variable, the inverse of the basis is obtained as:

$$(\mathbf{A}'_{\mathcal{B}'})^{-1} = \begin{pmatrix} \mathbf{A}_{\mathcal{B}}^{-1} & \mathbf{0} \\ -\mathbf{A}_{m+1,\mathcal{B}}\mathbf{A}_{\mathcal{B}}^{-1} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{\mathcal{B}}^{-1} & \mathbf{0} \\ -(\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & 1 \end{pmatrix} \quad \text{or}$$
$$(\mathbf{A}'_{\mathcal{B}'})^{-1} = \begin{pmatrix} \mathbf{A}_{\mathcal{B}}^{-1} & \mathbf{0} \\ \mathbf{A}_{m+1,\mathcal{B}}\mathbf{A}_{\mathcal{B}}^{-1} & -1 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{\mathcal{B}}^{-1} & \mathbf{0} \\ (\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & -1 \end{pmatrix} \tag{28}$$

where $(\mathbf{A}_{\mathcal{B}}^{-1})_{j.}$ is the row of $\mathbf{A}_{\mathcal{B}}^{-1}$ associated with the basic variable C_{ij}. The resulting basic solution is

$$\bar{\mathbf{x}}' = \begin{pmatrix} \bar{\mathbf{x}}'_{\mathcal{B}'} \\ \hline \bar{\mathbf{x}}'_{\mathcal{N}'} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}}' \\ \hline \mathbf{0} \end{pmatrix} = \begin{pmatrix} (\mathbf{A}'_{\mathcal{B}'})^{-1}\mathbf{b}' \\ \hline \mathbf{0} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}} \\ -\delta \\ \hline \mathbf{0} \end{pmatrix} \tag{29}$$

in both cases and violates primal feasibility because the new basic variable $s_{ij} = -\delta$ is strictly negative. Similarly, we compute

$$\bar{\mathbf{y}}' = \mathbf{c}'_{\mathcal{B}'}(\mathbf{A}'_{\mathcal{B}'})^{-1} = \begin{pmatrix} \mathbf{c}_{\mathcal{B}} & 0 \end{pmatrix}(\mathbf{A}'_{\mathcal{B}'})^{-1} = \begin{pmatrix} \bar{\mathbf{y}} & 0 \end{pmatrix} \tag{30}$$

and

$$\bar{\mathbf{c}}' = \mathbf{c}' - \bar{\mathbf{y}}'\mathbf{A}' = \begin{pmatrix} \mathbf{c} & 0 \end{pmatrix} - \begin{pmatrix} \bar{\mathbf{y}} & 0 \end{pmatrix}\mathbf{A}' = \begin{pmatrix} \bar{\mathbf{c}} & 0 \end{pmatrix} \tag{31}$$

in both cases. Note that $\bar{\mathbf{c}}' \geq \mathbf{0}$ since $\bar{\mathbf{c}} \geq \mathbf{0}$ because $\mathcal{B}$ is an optimal basic sequence for (**TTJm**)($\mathcal{M}^S$). Thus, $\mathcal{B}'$ is primal infeasible and dual feasible which suggests that we can continue with the dual simplex method. The basic variable $s_{ij} < 0$ is the only candidate for the leaving variable. The entering variable t is determined by the following ratio test in the dual simplex method:

$$\frac{\bar{\mathbf{c}}'_t}{\bar{\mathbf{A}}'_{(m+1)t}} = \max_{k|\bar{\mathbf{A}}'_{(m+1)k}<0} \frac{\bar{\mathbf{c}}'_k}{\bar{\mathbf{A}}'_{(m+1)k}} \tag{32}$$

where $\bar{\mathbf{A}}'_{(m+1).} = (\mathbf{A}'_{\mathcal{B}'})^{-1}_{(m+1).}\mathbf{A}'$. Plugging in the appropriate expressions for $(\mathbf{A}'_{\mathcal{B}'})^{-1}_{(m+1).}$ from (28) and $\mathbf{A}'$ leads to the following explicit set of formulas for $\bar{\mathbf{A}}'_{(m+1).}$:

$$\bar{\mathbf{A}}'_{(m+1)k} = \begin{Bmatrix} -\bar{\mathbf{A}}_{jj} + 1 = 0 & \text{for } C_{ij} \\ 1 & \text{for } s_{ij} \\ -\bar{\mathbf{A}}_{jk} & \text{o/w} \end{Bmatrix} \quad \text{or} \quad \bar{\mathbf{A}}'_{(m+1)k} = \begin{Bmatrix} \bar{\mathbf{A}}_{jj} - 1 = 0 & \text{for } C_{ij} \\ 1 & \text{for } s_{ij} \\ \bar{\mathbf{A}}_{jk} & \text{o/w} \end{Bmatrix}. \tag{33}$$

Recall that s_{ij} is the basic variable associated with row $m+1$, and hence $\bar{\mathbf{A}}'_{(m+1)(n+1)} = 1$ must hold as above. In addition, all components of $\bar{\mathbf{A}}'_{(m+1).}$ corresponding to the remaining basic variables must be zero, including those corresponding to the completion time variables. Inserting (31) and (33) into (32), the ratio test takes the form:

$$\frac{\bar{\mathbf{c}}'_t}{\bar{\mathbf{A}}'_{(m+1)t}} = \max_{k\neq j|\bar{\mathbf{A}}_{jk}>0} \frac{\bar{\mathbf{c}}_k}{-\bar{\mathbf{A}}_{jk}} = \frac{\bar{\mathbf{c}}_t}{-\bar{\mathbf{A}}_{jt}} \leq 0 \quad \text{or} \quad \frac{\bar{\mathbf{c}}'_t}{\bar{\mathbf{A}}'_{(m+1)t}} = \max_{k|\bar{\mathbf{A}}_{jk}<0} \frac{\bar{\mathbf{c}}_k}{\bar{\mathbf{A}}_{jk}} = \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \leq 0. \tag{34}$$

The crucial observation is that all quantities required to perform the ratio test (34) are computed based on the current optimal basic sequence $\mathcal{B}$ of (**TTJm**)($\mathcal{M}^S$).

Replacing s_{ij} by variable t in the basic sequence $\mathcal{B}'$ leads to the new basic sequence $\mathcal{B}'' = \mathcal{B} \cup \{t\}$. Next, we can carry out the pivoting operations in the simplex method. Here, we only need the updated dual variables $\bar{\mathbf{y}}''$ and the new objective function value. The dual variables are determined using the formula

$$\bar{\mathbf{y}}'' = \bar{\mathbf{y}}' + \frac{\bar{\mathbf{c}}'_t}{\bar{\mathbf{A}}'_{(m+1)t}}(\mathbf{A}'_{\mathcal{B}'})^{-1}_{(m+1).}. \tag{35}$$

Plugging in the relevant expressions, we obtain

$$\bar{\mathbf{y}}'' = \begin{pmatrix} \bar{\mathbf{y}} & 0 \end{pmatrix} - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \begin{pmatrix} -(\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & 1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{y}} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \end{pmatrix} \quad \text{or} \tag{36}$$

$$\bar{\mathbf{y}}'' = \begin{pmatrix} \bar{\mathbf{y}} & 0 \end{pmatrix} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \begin{pmatrix} (\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & -1 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{y}} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j.} & -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \end{pmatrix}. \tag{37}$$

Based on the conditions of the ratio test (34), we observe that the dual variable $\bar{\mathbf{y}}''_{m+1}$ associated with the new constraint (26) is nonpositive for $C_{ij} + s_{ij} = d_{ij} - \delta$ ($C_{ij} \leq d_{ij} - \delta$) and nonnegative for $C_{ij} - s_{ij} = d_{ij} + \delta$ ($C_{ij} \geq d_{ij} + \delta$), as expected. We also note that information already present in the optimal basic solution of (**TTJm**)($\mathcal{M}^S$) is sufficient to compute $\bar{\mathbf{y}}''$. Then, the objective value associated with $\mathcal{B}''$ is calculated by:

$$\bar{\mathbf{y}}''\mathbf{b}' = \begin{pmatrix} \bar{\mathbf{y}} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot} & -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ d_{ij} - \delta \end{pmatrix} = \bar{\mathbf{y}}\mathbf{b} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot}\mathbf{b} - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(d_{ij} - \delta)$$
$$= \bar{\mathbf{y}}\mathbf{b} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}\delta = \bar{\mathbf{y}}\mathbf{b} - \bar{\mathbf{y}}''_{m+1}\delta \geq \bar{\mathbf{y}}\mathbf{b} \qquad \text{(because } \bar{\mathbf{y}}''_{m+1} \leq 0 \text{ and } \delta > 0) \tag{38}$$

or

$$\bar{\mathbf{y}}''\mathbf{b}' = \begin{pmatrix} \bar{\mathbf{y}} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot} & -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ d_{ij} + \delta \end{pmatrix} = \bar{\mathbf{y}}\mathbf{b} + \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot}\mathbf{b} - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}(d_{ij} + \delta)$$
$$= \bar{\mathbf{y}}\mathbf{b} - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}}\delta = \bar{\mathbf{y}}\mathbf{b} + \bar{\mathbf{y}}''_{m+1}\delta \geq \bar{\mathbf{y}}\mathbf{b} \qquad \text{(because } \bar{\mathbf{y}}''_{m+1} \geq 0 \text{ and } \delta > 0), \tag{39}$$

where $(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot}\mathbf{b} = C^*_{ij} = d_{ij}$ since $(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot}\mathbf{b}$ provides the optimal value of the jth basic variable in (**TTJm**)($\mathcal{M}^S$).

Finally, we prove Proposition 3.1, our main result in this section, which allows us to specify the appropriate E/T cost parameters in the single-machine subproblems. Consider two successive iterations k and $k+1$ of SB-TPD. In iteration k, the optimal timing problem (**TTJm**)($\mathcal{M}^S$) yields an optimal objective value $z_{(\mathbf{TTJm})}(\mathcal{M}^S)$ and the optimal completion times C^*_{ij} for all j and for all $i = 1, \ldots, m_j$. Next, machine i^b is selected to be scheduled, and the optimal timing problem (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$) is solved providing a new set of optimal completion times C'_{ij} for all j and for all $i = 1, \ldots, m_j$. Then, Proposition 3.1, which we repeat below for completeness, establishes a lower bound on the increase in the objective value of the optimal timing problem from iteration k to $k+1$.

Proposition 3.1 *Consider the optimal timing problems* (**TTJm**)($\mathcal{M}^S$) *and* (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$) *solved in iterations k and $k+1$ of SB-TPD where i^b is the bottleneck machine in iteration k. For any operation o_{i^bj}, if $C'_{i^bj} = C_{i^bj} - \delta$ or $C'_{i^bj} = C_{i^bj} + \delta$ for some $\delta > 0$, then $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\}) - z_{(\mathbf{TTJm})}(\mathcal{M}^S) \geq | \bar{\mathbf{y}}''_{m+1} | \delta \geq 0$, where $\bar{\mathbf{y}}''$ is as defined in* (36)-(37).

Proof. We refer to the optimal timing problem (**TTJm**)($\mathcal{M}^S$) with the appropriate additional constraint $C_{i^bj} \leq d_{i^bj} - \delta$ or $C_{i^bj} \geq d_{i^bj} + \delta$ as (**TTJm**)$'$($\mathcal{M}^S$). The optimal objective value of (**TTJm**)$'$($\mathcal{M}^S$) is denoted by $z_{(\mathbf{TTJm})'}(\mathcal{M}^S)$.

The optimal solution of (**TTJm**)($\mathcal{M}^S \cup \{i^b\}$) satisfies all constraints present in (**TTJm**)$'$($\mathcal{M}^S$), in addition to the machine capacity constraints for machine i^b. Therefore, $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\})$ can be no less than $z_{(\mathbf{TTJm})'}(\mathcal{M}^S)$, and we can prove the desired result by showing that $z_{(\mathbf{TTJm})'}(\mathcal{M}^S) \geq z_{(\mathbf{TTJm})}(\mathcal{M}^S) - \bar{\mathbf{y}}''_{m+1}\delta$ or $z_{(\mathbf{TTJm})'}(\mathcal{M}^S) \geq z_{(\mathbf{TTJm})}(\mathcal{M}^S) + \bar{\mathbf{y}}''_{m+1}\delta$ as appropriate. Clearly, we can solve (**TTJm**)$'$($\mathcal{M}^S$) by starting from the optimal solution of (**TTJm**)($\mathcal{M}^S$) and applying the dual simplex method as discussed above. From (38)-(39), we already know that the increase in the objective function in the first iteration of the dual simplex method is at least $-\bar{\mathbf{y}}''_{m+1}\delta \geq 0$ or $+\bar{\mathbf{y}}''_{m+1}\delta \geq 0$ if $C_{i^bj} \leq d_{i^bj} - \delta$ or $C_{i^bj} \geq d_{i^bj} + \delta$ is added to (**TTJm**)($\mathcal{M}^S$), respectively. The proof is completed by noting that the dual simplex method produces non-decreasing objective values over the iterations. So, we have $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\}) \geq z_{(\mathbf{TTJm})'}(\mathcal{M}^S) \geq z_{(\mathbf{TTJm})}(\mathcal{M}^S) - \bar{\mathbf{y}}''_{m+1}\delta$ or $z_{(\mathbf{TTJm})}(\mathcal{M}^S \cup \{i^b\}) \geq z_{(\mathbf{TTJm})'}(\mathcal{M}^S) \geq z_{(\mathbf{TTJm})}(\mathcal{M}^S) + \bar{\mathbf{y}}''_{m+1}\delta$ as desired. □

Based on Proposition 3.1, if the completion time of o_{i^bj} decreases by δ time units in the optimal timing problem after inserting some set of disjunctive arcs on machine i^b, then the increase in the optimal objective function is no less than $-\bar{\mathbf{y}}''_{m+1}\delta$. This allows us to interpret the quantity $-\bar{\mathbf{y}}''_{m+1}$ as the unit earliness cost of operation o_{i^bj} in the subproblem of machine i^b, i.e., we set $\epsilon_{ij} = -\bar{\mathbf{y}}''_{m+1} = \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} = -\max_{k \neq j | \bar{\mathbf{A}}_{jk} > 0} \frac{\bar{\mathbf{c}}_k}{-\bar{\mathbf{A}}_{jk}}$ as in (19). A similar argument leads to $\pi_{ij} = \bar{\mathbf{y}}''_{m+1} = -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} = -\max_{k | \bar{\mathbf{A}}_{jk} < 0} \frac{\bar{\mathbf{c}}_k}{\bar{\mathbf{A}}_{jk}}$ for the corresponding unit tardiness cost.

Next, we investigate whether the approach presented here can be extended to multiple implicit dual simplex iterations that would allow us to construct a better approximation of the actual cost function for

C_{ij} depicted in Figure 2. To this end, using the formula

$$\bar{\mathbf{c}}'' = \bar{\mathbf{c}}' - \frac{\bar{\mathbf{c}}'_t}{\bar{\mathbf{A}}'_{(m+1)t}} \bar{\mathbf{A}}'_{(m+1)\cdot} \tag{40}$$

and inserting the expressions for $\bar{\mathbf{c}}'$ and $\bar{\mathbf{A}}'_{(m+1)\cdot}$ from (31) and (33) respectively, we first calculate the set of reduced costs $\bar{\mathbf{c}}''$ resulting from the first pivoting operation:

$$\bar{\mathbf{c}}''_k = \left\{ \begin{array}{ll} \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{-\bar{\mathbf{A}}_{jt}} \cdot 0 = 0 & \text{for } C_{ij} \\ \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{-\bar{\mathbf{A}}_{jt}} \cdot 1 = \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} & \text{for } s_{ij} \\ \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \bar{\mathbf{A}}_{jk} & \text{o/w} \end{array} \right\} \quad \text{or} \quad \bar{\mathbf{c}}''_k = \left\{ \begin{array}{ll} \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \cdot 0 = 0 & \text{for } C_{ij} \\ \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \cdot 1 = -\frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} & \text{for } s_{ij} \\ \bar{\mathbf{c}}_k - \frac{\bar{\mathbf{c}}_t}{\bar{\mathbf{A}}_{jt}} \bar{\mathbf{A}}_{jk} & \text{o/w} \end{array} \right\}, \tag{41}$$

where all required quantities are readily available in the optimal basic solution of (**TTJm**)($\mathcal{M}^S$) as before. We also observe that the updated reduced cost of s_{ij} is nonnegative in both cases as required in the dual simplex method. Then, $\bar{\mathbf{b}}''$ is computed based on:

$$\bar{\mathbf{b}}''_i = \left\{ \begin{array}{ll} \bar{\mathbf{b}}'_i - \frac{\bar{\mathbf{A}}'_{it}}{\bar{\mathbf{A}}'_{(m+1)t}} \bar{\mathbf{b}}'_{m+1} & i \neq m+1 \\ \frac{1}{\bar{\mathbf{A}}'_{(m+1)t}} \bar{\mathbf{b}}'_{m+1} & i = m+1 \end{array} \right\}. \tag{42}$$

Substituting $\bar{\mathbf{b}}'$ and $\bar{\mathbf{A}}'_{(m+1)t}$ from (29) and (33) respectively, we obtain the new values of the basic variables:

$$\bar{\mathbf{b}}''_i = \left\{ \begin{array}{ll} \bar{\mathbf{b}}_i + \frac{\bar{\mathbf{A}}'_{it}}{-\bar{\mathbf{A}}_{jt}} \delta & i \neq m+1 \\ \frac{\delta}{\bar{\mathbf{A}}_{jt}} & i = m+1 \end{array} \right\} \quad \text{or} \quad \bar{\mathbf{b}}''_i = \left\{ \begin{array}{ll} \bar{\mathbf{b}}_i + \frac{\bar{\mathbf{A}}'_{it}}{\bar{\mathbf{A}}_{jt}} \delta & i \neq m+1 \\ \frac{-\delta}{\bar{\mathbf{A}}_{jt}} & i = m+1 \end{array} \right\}, \tag{43}$$

where $\bar{\mathbf{b}}''_{m+1} > 0$ in both cases for the new basic variable t. Until now, we have only used information that is readily available in the current basic optimal solution of (**TTJm**)($\mathcal{M}^S$) and made no assumptions other than linearity regarding the specific objective or constraints of the job shop scheduling problem (**Jm**). However, computing (43) cannot be accomplished without compromising the generality of the analysis. Observe that the entering variable t is determined by the ratio test in (34) and may be different for each operation o_{ij}. Furthermore, in (43) we have $\bar{\mathbf{A}}'_{\cdot t} = (\mathbf{A}'_{\mathcal{B}'})^{-1} \mathbf{A}'_{\cdot t}$, where the entries of $\mathbf{A}'_{\cdot t}$ are given by the specific formulation under consideration. Consequently, the next basic sequence $\mathcal{B}''$, the associated values for the basic variables, and the resulting leaving variable all depend on o_{ij} and the specific job shop problem of interest. We conclude that we no longer have a single basis that allows us to compute all required quantities as efficiently as in (19). Estimating the actual cost function in Figure (2) more accurately boils down to solving an LP with a parametric right hand side per operation o_{ij}, $i \in \mathcal{M} \setminus \mathcal{M}^S$, $j \in J_i$. We refrain from this in order to retain the generality of our proposed approach and avoid the extra computational burden. Furthermore, our numerical results in Section 4 clearly demonstrate that the information retrieved from the subproblems is sufficiently accurate in a great majority of cases.

Appendix B. Computing ϵ_{ij} and π_{ij} Efficiently A fundamental implementation issue is computing the cost coefficients ϵ_{ij} and π_{ij} in the single-machine subproblems efficiently. The analysis in Appendix A reveals that the only information required to this end is the reduced costs of the nonbasic variables and the rows of $\bar{\mathbf{A}}$ associated with the completion time variables in a basic optimal solution of the current optimal timing problem. This information may be directly available from the linear programming solver employed to solve the optimal timing problem. Otherwise, the computational burden of computing $\mathbf{A}_{\mathcal{B}}^{-1}$ renders this step time consuming. In this section, we show how $\mathbf{A}_{\mathcal{B}}^{-1}$ and $\bar{\mathbf{A}}$ can be computed efficiently for ($\widehat{\mathbf{TTJm}}$) by exploiting the structure and sparsity of $\mathbf{A}$. In this analysis, we only assume that the basic sequence $\mathcal{B}$ and the corresponding nonbasic sequence $\mathcal{N}$ associated with the optimal basis of the current timing problem are available. Denoting the number of operations by $t = \sum_{j=1}^{n} m_j$, we note that the optimal basis is a $q \times q$ matrix and has the following block form

$$\mathbf{A}_{\mathcal{B}} = \begin{pmatrix} \mathbf{T} & \mathbf{U} \\ \mathbf{V} & \mathbf{W} \end{pmatrix}, \tag{44}$$

where $\mathbf{T}$ is a $t \times t$ square matrix composed of the coefficients of the completion time variables (all completion time variables are always basic) in the ready time and operation precedence constraints (2)-(3), $\mathbf{U}$ is a $t \times (q-t)$ matrix of the coefficients of the remaining basic variables in the same constraints,

$\mathbf{V}$ is a $(q-t)\times t$ matrix of the coefficients of the completion time variables in the last $q-t$ constraints (7),(8), (12), and $\mathbf{W}$ is a $(q-t)\times(q-t)$ matrix of the coefficients of the remaining basic variables in the same constraints. Then, $\mathbf{A}_{\mathcal{B}}^{-1}$ is obtained as

$$\mathbf{A}_{\mathcal{B}}^{-1}=\begin{pmatrix}\mathbf{T}^{-1}+\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}\mathbf{V}\mathbf{T}^{-1} & -\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}\\ -\mathbf{Z}^{-1}\mathbf{V}\mathbf{T}^{-1} & \mathbf{Z}^{-1}\end{pmatrix}, \tag{45}$$

where $\mathbf{Z}=\mathbf{W}-\mathbf{V}\mathbf{T}^{-1}\mathbf{U}$ is the Schur complement of $\mathbf{T}$. It is a simple exercise to show that $\mathbf{T}$ is invertible, and $\mathbf{Z}$ must be invertible because $\mathbf{A}_{\mathcal{B}}$ is invertible. An important property of ($\widehat{\mathbf{TTJm}}$) is that all variables in ($\widehat{\mathbf{TTJm}}$) except for the completion time variables and the makespan variable are either slack or surplus variables with a single nonzero coefficient in the corresponding column of $\mathbf{A}$.

For computing the values of ϵ_{ij} and π_{ij} for an operation o_{ij} as defined in (19), we only need the entries of the row $\bar{\mathbf{A}}_{j\cdot}=(\mathbf{A}_{\mathcal{B}}^{-1})_{j\cdot}\mathbf{A}$ corresponding to the nonbasic variables, where C_{ij} is the jth basic variable as in Appendix A. We denote these entries by $\bar{\mathbf{A}}_{j\mathcal{N}}$. Thus, computing the cost coefficients in all single-machine subproblems requires no more than computing the first t rows of $\bar{\mathbf{A}}$ given by the first row of (45). We detail the steps of this task below, where representing a matrix $\mathbf{M}$ in sparse form refers to storing the nonzero entries of $\mathbf{M}$ column by column in a single vector while the corresponding row indices are saved in a separate vector. A third vector holds a pointer to the last entry of each column in the other two vectors.

$\mathbf{T}^{-1}$ is computed only once during the initialization of the algorithm and saved in sparse form. Then, we perform the following steps each time we need to set up the single-machine subproblems after optimizing ($\widehat{\mathbf{TTJm}}$):

1. Store $\mathbf{U}$ in sparse form by noting that all nonzero coefficients in $\mathbf{U}$ correspond to the waiting time variables.
2. Compute $\mathbf{V}\mathbf{T}^{-1}$ row by row in dense form by observing the following:
   a. In a row of $\mathbf{V}$ corresponding to a due date constraint (7) or a $C_{\max}$ constraint (8) for job j, there is a single entry +1 in the column corresponding to $C_{m_j j}$.
   b. A row of $\mathbf{V}$ corresponding to a machine capacity constraint (12) includes exactly two nonzero coefficients +1 and −1 corresponding to two consecutive operations performed on the same machine.
3. Compute $-\mathbf{V}\mathbf{T}^{-1}\mathbf{U}$ in dense form by re-arranging the columns of $\mathbf{V}\mathbf{T}^{-1}$ and observing the following:
   a. There is exactly a single nonzero entry +1 or −1 in the columns of $\mathbf{U}$ corresponding to the waiting time variables.
   b. All other columns of $\mathbf{U}$ are identical to zero.
4. Compute $\mathbf{Z}=\mathbf{W}-\mathbf{V}\mathbf{T}^{-1}\mathbf{U}$ in dense form. $\mathbf{W}$ is retrieved column by column in sparse form, and the nonzero entries are added to the appropriate entries of $-\mathbf{V}\mathbf{T}^{-1}\mathbf{U}$ calculated in the previous step. We can completely disregard the waiting time variables in this step because the associated coefficients in $\mathbf{W}$ are all zero.
5. Compute $\mathbf{Z}^{-1}$.
6. Compute $\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}$ and $-\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}$ in dense form:
   a. The nonzero entries in a given row of $\mathbf{T}^{-1}\mathbf{U}$ may be determined by using $\mathbf{T}^{-1}$ and $\mathbf{U}$ already stored in sparse form. Since there is at most a single nonzero entry in each column of $\mathbf{U}$, we can traverse the columns of $\mathbf{T}^{-1}$ in the order specified by the associated row indices of the nonzero entries in $\mathbf{U}$ and easily calculate the nonzero entries of $\mathbf{T}^{-1}\mathbf{U}$ in the specified row.
   b. $\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}$ and $-\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}$ are then computed row by row in dense form by taking linear combinations of the rows of $\mathbf{Z}^{-1}$ as specified by the nonzero entries in the rows of $\mathbf{T}^{-1}\mathbf{U}$ calculated above.
7. Compute $\mathbf{T}^{-1}+\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}\mathbf{V}\mathbf{T}^{-1}$ in dense form by multiplying $\mathbf{T}^{-1}\mathbf{U}\mathbf{Z}^{-1}$ and $\mathbf{V}\mathbf{T}^{-1}$, and then adding $\mathbf{T}^{-1}$ available in sparse form to the result.
8. For the objective function coefficients of the operations in the single-machine subproblems, we need to compute the first t rows of $\bar{\mathbf{A}}_{\cdot\mathcal{N}}=\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_{\cdot\mathcal{N}}$. This is performed column by column for each $k\in\mathcal{N}$. We have $\bar{\mathbf{A}}_{\cdot k}=\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_{\cdot k}$, where $\mathbf{A}_{\cdot k}$ includes a single nonzero entry +1 or −1 because all nonbasic variables in ($\widehat{\mathbf{TTJm}}$) are either slack or surplus variables. Thus, we simply need to retrieve the first t entries in the proper column of $\mathbf{A}_{\mathcal{B}}^{-1}$ computed previously and multiply them with −1 if necessary.

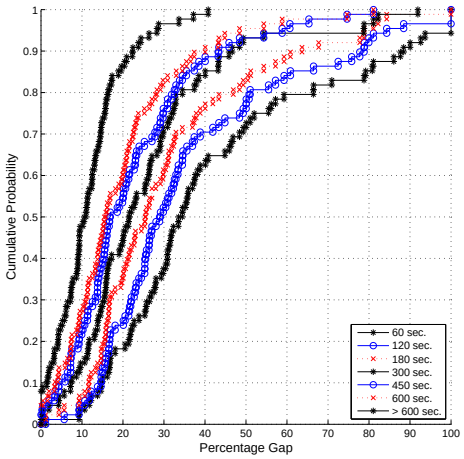
Appendix C. Solution Quality vs. Time

C.1 Job Shop Total Weighted E/T Problem With Intermediate Inventory Holding Costs To investigate the behavior of SB-TPD over the course of the algorithm, we present a detailed analysis of the solution quality versus the solution time and the number of feasible schedules constructed in SB-TPD in Figure 3.

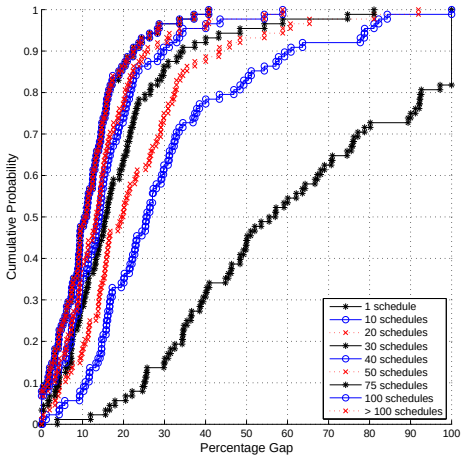
To this end, we take a snapshot of the optimality gaps after 60, 120, 180, 300, 450, and 600 seconds of solution time and depict the empirical cumulative distributions of these gaps for $f = 1.0, 1.3$, $f = 1.5$, and $f = 1.7, 2.0$ in 3(a), 3(c), and 3(e), respectively. In these figures, gaps larger than 100% appear as 100%. After 120 seconds of solution time, 4.5% (4 out of 88), 27.3% (12 out of 44), and 73.9% (65 out of 88) of the instances are within 10% of the best solution for $f = 1.0, 1.3$, $f = 1.5$, and $f = 1.7, 2.0$, respectively. The corresponding figures after 300 seconds of solution time are obtained as 14.8% (13 out of 88), 54.5% (24 out of 44), and 89.8% (79 out of 88). For $f = 1.0, 1.3$, the gaps are no more than 5% and 10% for 13.6% (12 out of 88) and 27.3% (24 out of 88) of the instances after 600 seconds of solution time, respectively. For $f = 1.5$, 40.9% (18 out of 44) and 65.9% (29 out of 44) of the instances are within 3% and 10% of the best solution after 600 seconds of solution time, respectively. For $f = 1.7, 2.0$, the best solution is obtained for 44.3% (39 out of 88) of the instances in 600 seconds of solution time, and the gap is within 5% for 85.2% (75 out of 88) of the instances while this number increases to 95.5% (84 out of 88) for a gap of at most 10%. Similar figures are produced for the number feasible schedules constructed in SB-TPD and appear on the right hand side in Figure 3. Note that each leaf node in the search tree in SB-TPD corresponds to a feasible schedule for $(\widehat{\mathbf{Jm}})$. Intuitively, if the shifting bottleneck framework performs well, then good solutions should be identified early in the search tree. These figures indicate that with as few as 100 feasible schedules constructed over the entire course of the algorithm high quality solutions are obtained. Furthermore, these figures attest to the enhanced performance of our heuristic as f grows. With increasing f, we come across excellent solutions very early in the algorithm. For instance, for $f = 1.5$ the best available solutions are identified in more than 20% of the instances with at most 20 feasible schedules. This figure increases to above 25% for $f = 1.7, 2.0$.

C.2 Job Shop Total Weighted Completion Time Problem with Intermediate Inventory Holding Costs For this case, we see in Figure 4(b) that the very first schedule constructed achieves a gap of no more than 10% with respect to the best available solution in more than 1/3 of the instances. With at most 10 feasible schedules constructed, the gap is reduced to less than 5% in more than 50% of the instances.

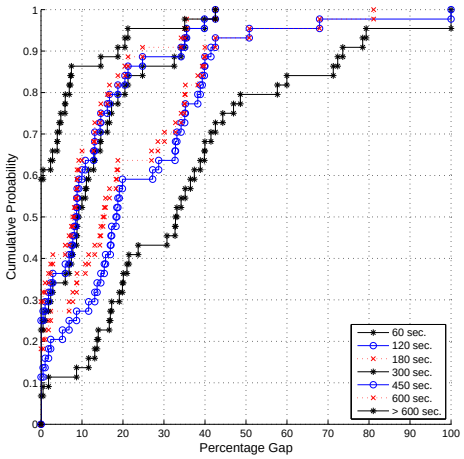
C.3 Job Shop Makespan Problem with Intermediate Inventory Holding Costs The effectiveness of SB-TPD for this problem is readily apparent from Figure 5(b). The very first schedule constructed is no more than 15% away from the best known solution for 30% (13 out of 44) of the instances. With at most 10 feasible schedules constructed, this gap falls below 10% in 50% of the instances.



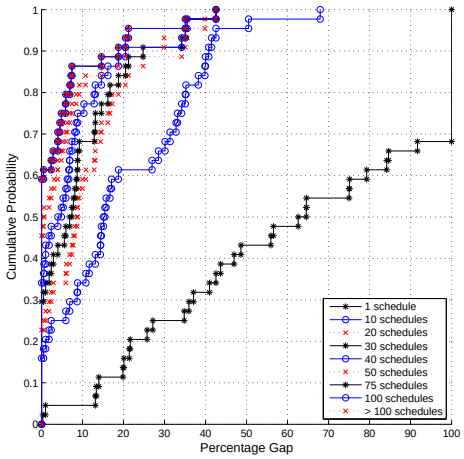
(a) $f = 1.0, 1.3$.



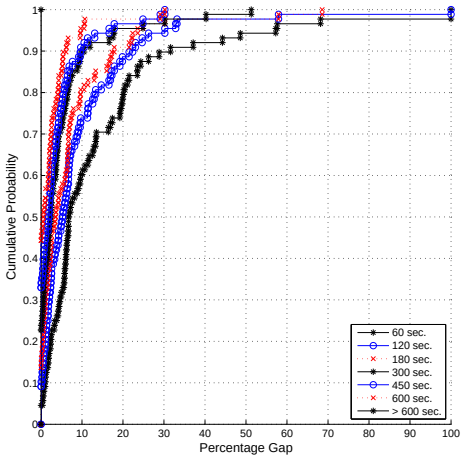
(b) $f = 1.0, 1.3$.



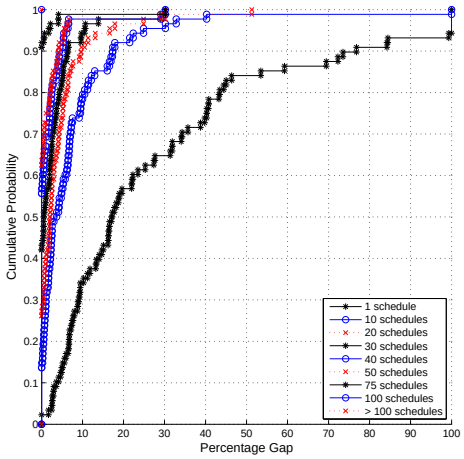
(c) $f = 1.5$.



(d) $f = 1.5$.

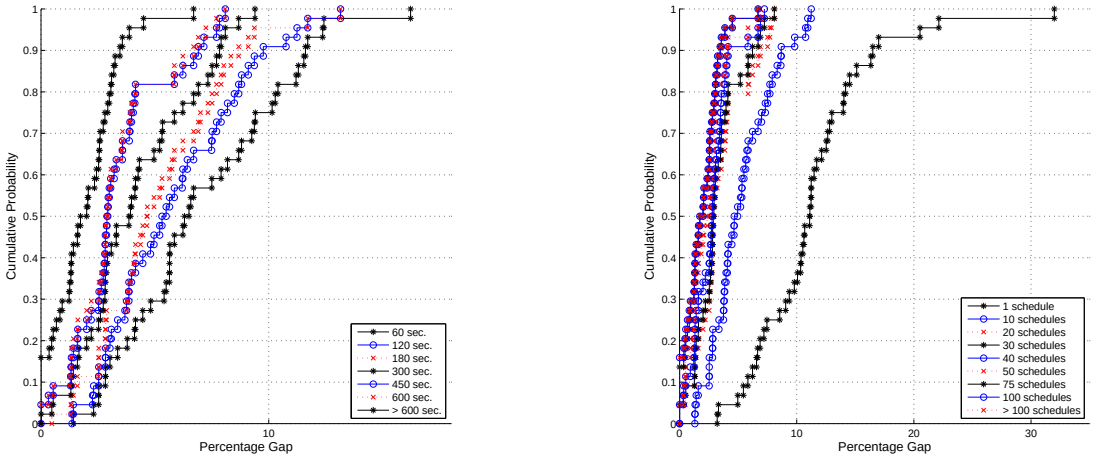


(e) $f = 1.7, 2.0$.



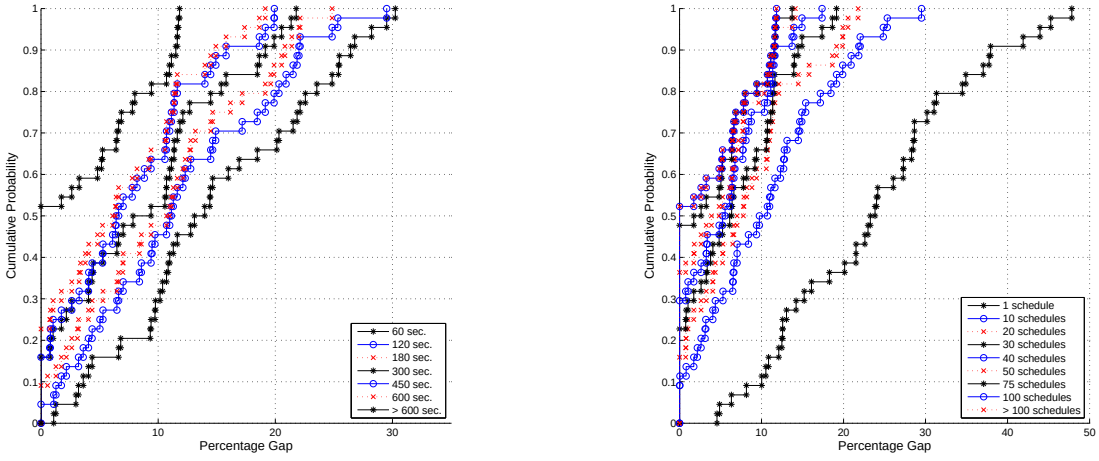
(f) $f = 1.7, 2.0$.

Figure 3: The progress of the solution gaps with respect to the best available solution for the job shop total weighted E/T problem with intermediate inventory holding costs.



(a) (b)

Figure 4: The progress of the solution gaps with respect to the best available solution for the job shop total weighted completion time problem with intermediate inventory holding costs.



(a) (b)

Figure 5: The progress of the solution gaps with respect to the best available solution for the job shop makespan problem with intermediate inventory holding costs.