J Sched (2007) 10: 271–292
DOI 10.1007/s10951-007-0028-6

Preemption in single machine earliness/tardiness scheduling

Kerem Bülbül · Philip Kaminsky · Candace Yano

Published online: 10 August 2007

Abstract We consider a single machine earliness/tardiness scheduling problem with general weights, ready times and due dates. Our solution approach is based on a time-indexed preemptive relaxation of the problem. For the objective function of this relaxation, we characterize cost coefficients that are the best among those with a piecewise linear structure with two segments. From the solution to the relaxation with these best objective function coefficients, we generate feasible solutions for the original non-preemptive problem. We report extensive computational results demonstrating the speed and effectiveness of this approach.

Keywords Single-machine scheduling · Earliness · Tardiness · Preemption · Transportation problem

1 Introduction

In the single machine earliness/tardiness (E/T) scheduling problem, a set of jobs, each with an associated due date, has to be scheduled on a single machine. Each job has a penalty per unit time associated with completing before its due date, and a penalty per unit time associated with completing after its due date. E/T problems have been a popular topic of research since the early 1980's because they reflect the just-in-time (JIT) philosophy, in which early shipments are discouraged and result in inventory holding costs in addition to the tardiness penalties associated with the loss of customer goodwill. Although the single machine E/T problem is useful in its own right, our interest in it is further motivated by its appearance as a subproblem in solution approaches for more complex scheduling problems (see Ovacik and Uzsoy 1997 and Bülbül et al. 2004). In these more general contexts, it is necessary to solve many such problems very rapidly. As solution speed is very important, we focus on the development of fast, effective heuristics for this problem.

Consider a non-preemptive single machine scheduling problem with n jobs. Associated with each job j, $j = 1, \dots, n$, are several parameters: p_j, the processing time for job j; r_j, the ready time for job j; d_j, the due date for job j; ϵ_j, the earliness cost per unit time if job j completes processing before d_j; and π_j, the tardiness cost per unit time if job j completes processing after d_j. We assume that the processing times, ready times and due dates are integers. Let s_j be the time at which job j starts processing, $C_j = s_j + p_j$ be the completion time of job j, and $E_j = \max(0, d_j - C_j)$ and $T_j = \max(0, C_j - d_j)$ be the earliness and tardiness of job j, respectively. The objective is to minimize the sum of costs for all jobs. Then our problem is stated as:

$$\textbf{(P1)} \quad \min \sum_{j=1}^{n} (\epsilon_j * E_j + \pi_j * T_j), \tag{1.1}$$

$$s_j \geq r_j, \quad \forall j, \tag{1.2}$$

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

K. Bülbül (✉)
Manufacturing Systems and Industrial Engineering, Sabancı University, Istanbul, Turkey
e-mail: bulbul@sabanciuniv.edu

P. Kaminsky · C. Yano
Industrial Engineering and Operations Research, University of California, Berkeley, CA, USA

C. Yano
Haas School of Business, University of California, Berkeley, CA, USA

$$s_j + p_j + E_j - T_j = d_j, \quad \forall j, \tag{1.4}$$

$$E_j, T_j \geq 0, \quad \forall j. \tag{1.5}$$

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

In classifying scheduling problems, we follow the three field notation of Graham et al. (1979). Problem P1 is represented as $1/r_j/\sum(\epsilon_j E_j + \pi_j T_j)$, where, in the first field, 1 indicates a single machine problem, and the entry r_j in the second field denotes that the ready times may be unequal. P1 is strongly $\mathcal{NP}$-hard because the problem $1/r_j/\sum \pi_j T_j$, which is obtained from P1 by setting all earliness costs equal to zero, is known to be strongly $\mathcal{NP}$-hard (Lenstra et al. 1977).

We observe that P1 would be a linear program (LP) if we knew the sequence of jobs. This is a property of E/T scheduling problems that is often used to develop a two-phase heuristic: in the first phase, a good job processing sequence is determined and then, in the second phase, idle time is inserted either by solving a linear program or by using a specialized algorithm that exploits the structure of the optimal solution with fixed job processing sequences. The second phase is usually referred to as the *optimal timing* problem. Researchers have observed that the total cost in E/T scheduling problems is dominated by the sequencing decisions in the first phase; the timing decisions have a smaller impact. We develop a two-phase heuristic in this paper and focus on the construction of a good job processing sequence based on these observations.

Any feasible schedule for P1 is a sequence of *blocks* separated by idle time, and the cost of a block is a piecewise linear convex function of the start time of the block. (See Yano and Kim 1991.) Optimal timing algorithms of low order polynomial complexity have been developed for several special cases of P1 using this property (Garey et al. 1988; Davis and Kanet 1993; Szwarc and Mukhopadhyay 1995; Lee and Choi 1995). Also, see Kanet and Sridharan (2000) for an overview of different timing algorithms. For our problem, once jobs are sequenced and assuming jobs are renumbered in sequence order, the optimal schedule is found by solving the linear program TT–P1 below.

$$\textbf{(TT–P1)} \quad \min \sum_{j=1}^{n} (\pi_j * T_j + \epsilon_j * E_j),$$
$$(1.2),\ (1.4\text{–}1.5),$$
$$s_j + p_j \leq s_{j+1}, \quad \forall j \neq n.$$

Although there exists an $O(n^2)$ algorithm for solving the optimal timing problem for P1 (Nandkeolyar et al. 1993), we use the LP above to determine the final schedule in our heuristics. LPs for the timing problem are small and take very little CPU time even if several of them are solved for a single problem instance.

Many different types of algorithms have been proposed in the literature for various single machine E/T scheduling problems. A survey of the early research on earliness/tardiness problems appears in Baker and Scudder (1990). More recent research can be found in Kanet and Sridharan (2000) and in references therein. In our discussion here, we restrict our attention to papers that allow idle time in the schedule. Since the problem P1 is strongly $\mathcal{NP}$-hard, most approaches for P1 are either branch and bound (B&B) algorithms or dispatch heuristics based on dominance properties for the job processing sequence. The B&B algorithms also make use of these dominance conditions to reduce the size of the search tree, but none of them is effective for problems with more than 20 to 30 jobs, due to the lack of strong lower bounds for E/T scheduling problems. Szwarc (1993) develops precedence and decomposition rules for the problem $1//\epsilon \sum E_j + \pi \sum T_j$. He proposes a branching scheme that can handle small problems ($n = 10$) without a lower bound. Yano and Kim (1991) devise a B&B algorithm for the special case of the weighted earliness/tardiness problem, where earliness and tardiness costs are proportional to the processing times. This property allows them to develop dominance properties that are effective, and they can solve problems with up to 20 jobs optimally. The authors report that their lower bounds are loose and further research is needed for improved lower bounds for their problem. Kim and Yano (1994) and Fry et al. (1996) develop B&B algorithms for the mean earliness and tardiness problem. Both papers use various adjacency conditions and propose lower bounds based on elimination of time conflicts. In these papers the authors solve problems with up to 20 jobs optimally with moderate computation times. Fry and Keong Leong (1987) formulate a mixed integer program for the problem $1//w \sum C_j + \epsilon \sum E_j$, which can be expressed as an equivalent E/T problem, and try to solve it using commercial software. For the same problem, Hoogeveen and van de Velde (1996) develop several dominance conditions and lower bounds, and attempt to solve instances with up to 20 jobs. They conclude their paper by: "We have considered an $\mathcal{NP}$-hard machine scheduling problem in which the insertion of idle time may be advantageous, and we have presented a branch-and-bound algorithm for its solution. The allowance of machine idle time complicates the design of the algorithm substantially. Also, the performance of the algorithm is quite bleak: this is because it is very difficult to compute strong lower bounds," which summarizes the difficulties of all B&B approaches for E/T scheduling problems. In fact, one of our major contributions in this paper is the development of a strong lower bound for the problem $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$. Also, note that two lower bounds that are closely related to ours were proposed by Sourd and

Kedad-Sidhoum (2003) and Sourd (2004) and were successfully implemented in B&B algorithms to solve the problem $1//\sum \epsilon_j E_j + \pi_j T_j$. These authors obtain optimal solutions for problem instances with 20 jobs within a few seconds, and 30-job instances can be solved to optimality within at most a few minutes. We discuss the similarities and differences between our research and the approaches presented by Sourd and Kedad-Sidhoum (2003) and Sourd (2004) in more detail at the end of this section.

The difficulty of designing effective optimal procedures for E/T problems prompted the development of many heuristics for these problems. Most of these heuristics follow a two-phase approach as mentioned above. Examples of such approaches can be found in Fry et al. (1987, 1990), Lee and Choi (1995), Wan and Yen (2002), and Nandkeolyar et al. (1993). The heuristics of Fry et al. (1987) provide solutions for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$, whose objective values are on average less than 2% above the optimal objective values for problem instances with up to 15 jobs. Similarly, for the problem $1//\sum E_j + T_j$, Fry et al. (1990) report that their heuristics provide solutions that are on average within 2.5% of optimality for instances with up to 16 jobs. Lee and Choi (1995) design a genetic algorithm for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$ in which the sequence information is coded into the chromosomes, and the fitness of the chromosomes is evaluated by an optimal timing algorithm. Their computational results indicate that their approach is superior to the heuristics proposed by Yano and Kim (1991). The tabu search procedure suggested by Wan and Yen (2002) considers a generalization of the problem $1//\sum \epsilon_j E_j + \pi_j T_j$ to due windows. For relatively small problem instances with 15 or 20 jobs, the tabu search finds the optimal solution in over 20% of the cases, and for larger problems it compares favorably to incumbent solutions from a B&B algorithm. Ventura and Radhakrishnan (2003) develop a Lagrangian-relaxation-based heuristic for $1//\sum E_j + T_j$. They report that the gap between their best lower bound and best feasible solution is less than 3% for problem instances with up to 100 jobs. Nandkeolyar et al. (1993), Sridharan and Zhou (1996) and Mazzini and Armentano (2001) consider the problem $1/r_j/\sum \epsilon_j E_j + \pi_j T_j$. The latter two papers differ from the rest of the literature in that they combine sequencing with insertion of idle time. Sridharan and Zhou (1996) employ a look-ahead procedure at each decision point in order to determine whether to keep the machine idle or schedule a job immediately. Their solutions are approximately 6% above the optimal solution for one problem set with eight-job instances, and very close to optimality in another problem set with up to 40 jobs. Mazzini and Armentano (2001) first sequence the jobs in non-decreasing order of their respective *target* start times, i.e., $\max(r_j, d_j - p_j)$, and then insert jobs into the current partial schedule one by one, considering possible idle time. For problems with up to 80 jobs, their algorithms provide solutions that are on average within 4.5% of optimality. (For $n \geq 20$, the optimality gaps are computed with the help of the lower bound developed in this paper.) Note that these authors present one of the very few heuristics that simultaneously considers different ready times and weighted earliness and tardiness penalties, and combines sequencing with insertion of idle time. Thus, we compare the performance of our heuristics to that of Mazzini and Armentano (2001) in Sect. 5.3.3 in order to demonstrate the effectiveness of our algorithms. Most of the heuristics mentioned above can solve problems quickly. However, generally it is not possible to quantify the quality of their solutions for large problem instances because neither optimal solutions nor tight lower bounds are available for such instances. One exception here is the Lagrangian-relaxation-based heuristic of Ventura and Radhakrishnan (2003), which provides a lower bound for the problem $1//\sum E_j + T_j$. However, their subgradient algorithm is very slow, taking on average 18 minutes to solve their 100-job instances.

In this paper our main contribution is a new strategy for solving P1. Unlike existing dispatch rules or pairwise interchange heuristics for this problem, our algorithms take into account all economic trade-offs at the same time and thus are not myopic. We develop a tight lower bound for P1 based on a preemptive solution that is easy to compute, and achieves an excellent balance between solution speed and quality. We have two major goals. First, we use the optimal preemptive solution in developing a good, non-preemptive solution for P1. Second, we obtain a tight lower bound for specific instances that can be computed quickly, and is useful for quantifying the quality of heuristic solutions. Obtaining tight lower bounds has been regarded as a major challenge.

Our algorithms depend on a preemptive relaxation that is computed by solving a time-indexed problem, in which we allow a job to be preempted at integer points in time. (See Sect. 2.) In the time-indexed formulation, we need to consider a planning horizon whose length depends not only on the sum of the processing times but also the due dates because both earliness and tardiness costs are present in the objective function. Hence, from a theoretical viewpoint we compute a lower bound for P1 in pseudo-polynomial time. However, the time-indexed formulation can be solved quickly once it is constructed from the original problem data, and we demonstrate the effectiveness of our approach in the computational experiments in Sect. 5 by solving problem instances of realistic size (with up to 200 jobs). We also note that this same lower bound was successfully used in solving subproblems in a column generation algorithm applied to an m-machine flow shop E/T scheduling problem (Bülbül et al. 2004), which is further evidence of the efficiency and effectiveness of our lower bound. In addition, the optimal solution of the preemptive relaxation has sufficient structure to serve as a starting point for constructing several feasible solutions which contribute to improving the quality of the final heuristic solution for P1. Once the lower bound is obtained, feasible solutions are generated very quickly by solving TT–P1 several times with different job processing sequences. Theoretically, we could even use a special optimal timing algorithm of complexity $O(n^2)$ as mentioned earlier.

The approaches presented by Sourd and Kedad-Sidhoum (2003) and Sourd (2004), developed independently from our approaches (also see Bülbül 2002), deserve special attention here because the lower bounds proposed in these papers are closely related to ours. In both of these papers, B&B algorithms are developed to solve the problem $1//\sum \epsilon_j E_j + \pi_j T_j$, and the focus is on obtaining good lower bounds. In fact, the two B&B algorithms differ only in the computation of the lower bounds, which are based on different preemptive relaxations. Two major factors affect the quality of the lower bounds obtained from these relaxations: the preemption scheme, i.e., when and for how long a job can be preempted, and the objective function of the relaxation. (See Sect. 2.) Sourd and Kedad-Sidhoum (2003) consider a time-indexed preemptive relaxation, in which a job can be preempted at integer points in time, as in our approach. However, Sourd (2004) proposes a different approach, in which each job may be preempted at any point in time. A lower bound for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$ can be computed in polynomial time, when based on a continuous relaxation rather than a time-indexed one. These authors observe that the two B&B algorithms perform similarly despite the theoretical differences in the lower bounds. Both Sourd and Kedad-Sidhoum (2003) and Sourd (2004) consider a family of valid objective functions for their respective relaxations and heuristically choose one for their computational experiments. Here, we follow a different path and develop an objective function for our preemptive relaxation based on an intuitively simple observation in Sect. 2. In Sect. 3.2 this objective function is shown to be optimal with respect to certain criteria and, interestingly, it also turns out to be closely related to the objective function developed for the continuous relaxation by Sourd (2004). In other words, our research provides the link between the "discrete" and "continuous" lower bounds proposed by Sourd and Kedad-Sidhoum (2003) and Sourd (2004), respectively. We elaborate further on this issue in the next section. Finally, one of our major emphases is the construction of good feasible solutions for the original problem after the optimal solution of the preemptive problem is obtained. Sourd and Kedad-Sidhoum (2003) and Sourd (2004) pay less attention to this phase of the solution procedure. In Sect. 5 we demonstrate that our lower bound exhibits some desirable properties that lead to a more stable behavior in obtaining good feasible solutions for P1.

In Sect. 2 we review some general properties of preemptive E/T problems that lead us to an appropriate preemption strategy for P1. Then we develop a penalty scheme for this preemptive relaxation and we show that a formulation based on this scheme leads to an excellent lower bound for P1. In Sect. 3 we derive some properties of our lower bound. In Sect. 4 we present an heuristic that uses the information from the optimal solution of the preemptive relaxation to create a feasible solution for P1. Finally, in Sect. 5 we present the results of our computational experiments which demonstrate the effectiveness of our lower bound and of our heuristics. We conclude in Sect. 6.

2 A lower bound for P1

The complexity of a preemptive scheduling problem depends strongly on the penalty scheme applied to different job segments. If only the last portion of a job may incur a non-zero cost in a preemptive E/T scheduling problem $1/prmp/\sum(\epsilon_j E_j + \pi_j T_j)$, then earliness costs can be made negligibly small by scheduling an arbitrarily small portion of duration $\delta > 0$ to complete at time d_j for any early job j (Mosheiov 1996). Hence, under this cost structure we can treat $1/prmp/\sum(\epsilon_j E_j + \pi_j T_j)$ as a preemptive weighted tardiness problem $1/prmp/\sum \pi_j T_j$, for which every preemptive schedule can be transformed into a non-preemptive schedule with no larger objective value (McNaughton 1959). Therefore, both problems $1/prmp/\sum \pi_j T_j$ and $1/prmp/\sum(\epsilon_j E_j + \pi_j T_j)$ are as hard as the non-preemptive weighted tardiness problem $1//\sum \pi_j T_j$, which is known to be $\mathcal{NP}$-hard (see Lenstra et al. 1977). So, we conclude that a preemptive E/T scheduling problem, in which only the last portion of a job may incur a non-zero cost, is unlikely to provide a tight lower bound for P1. However, we highlight an important property of the weighted tardiness problem. If all jobs have *unit processing times*, then a non-preemptive weighted tardiness problem can be solved in polynomial time because it is equivalent to a transportation problem. This result leads us to a different preemption scheme in order to develop a lower bound for P1.

Our approach for constructing a lower bound builds upon the ideas of Gelders and Kleindorfer (1974) for the single machine weighted tardiness problem and the results of Verma and Dessouky (1998) for the single machine non-preemptive weighted E/T scheduling problem with unit processing times. To obtain a lower bound, Gelders and Kleindorfer divide each job into unit-duration intervals (jobs) and associate a cost with each unit job. We utilize their idea of unit-duration jobs and allow a job to be preempted at integer points in time. Their other results, however, cannot be used directly because their objective function differs from ours, and non-delay schedules are optimal in weighted tardiness problems, but may be suboptimal when earliness costs are considered. We also note that the form of preemption that we consider is often implemented in practical applications, due to the nature of the manufacturing process or the structure of the work schedule, by simply dividing customer orders into smaller processing batches in an appropriate way. Thus, although our study was motivated partly by theoretical considerations, the results may be useful in practice.

We first present some properties of the optimal solution of P1 that lead us to a transportation problem, whose optimal objective value is a lower bound on that of P1. We state the following property without proof. The reader is referred to Sourd and Kedad-Sidhoum (2003) for a formal proof of a similar property for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$.

Property 2.1 *If processing times, due dates and ready times are integers, then there exists an optimal solution for P1 in which all job completion times are integers.*

Without loss of generality, Property 2.1 indicates that the solution space can be restricted to integer start times. Intuitively, an approximation to P1 could be obtained by dividing each job j into p_j unit-duration jobs, associating a penalty with each unit job (rather than only with the last one completed) and planning for a horizon consisting of an appropriate integral number time periods. Before doing so, we restate P1 as an equivalent binary non-linear program. Then we discuss modifications to this problem that lead to a relaxation, from which we derive a lower bound. In the time-indexed formulations in this paper, a period k spans from time $k-1$ to k; hence, a job j with ready time r_j can be processed no earlier than in period r_j+1. Let $X_{jk}=1$ if job j is processed in period k, and 0 otherwise; and let $(x)^+=\max(x,0)$. So we have:

$$(\mathbf{P2})\quad \min \sum_j \sum_{k\in H} \left[\epsilon_j (d_j-k)^+ + \pi_j (k-d_j)^+\right] * (X_{jk}-X_{jk+1})^+, \tag{2.1}$$

$$\sum_{k\in H} X_{jk} = p_j, \quad \forall j, \tag{2.2}$$

$$\sum_j X_{jk} \le 1, \quad \forall k\in H, \tag{2.3}$$

$$\sum_{k\in H} (X_{jk}-X_{jk+1})^+ = 1, \quad \forall j, \tag{2.4}$$

$$X_{j t_{\max}+1} = 0, \quad \forall j, \tag{2.5}$$

$$X_{jk}\in\{0,1\}, \quad \forall j, k\in H\cup\{t_{\max}+1\}. \tag{2.6}$$

Let $t_{\min}=\min_j r_j+1$, $t_{\max}=\max_j \max(r_j,d_j)+P$, where $P=\sum_j p_j$ is the sum of processing times. Also let the planning horizon be H, and define $H=\{k \mid k\in\mathbb{Z}, k\in[t_{\min},t_{\max}]\}$. Thus, the maximum value of k in H is greater than or equal to the latest period in which any portion of any job could conceivably be processed in an optimal solution, accounting for potential idle time. In the non-preemptive problem, the expression $(X_{jk}-X_{jk+1})^+$ takes the value one only if k is the completion time of job j and is zero, otherwise. So the term in square brackets in the objective function is the cost incurred if job j finishes in period k. The constraints (2.2) and the contiguity constraints (2.4–2.5) together ensure that each job j is processed in p_j consecutive periods. The machine can process at most one job in any given period as specified by constraints (2.3).

Now we present a new formulation obtained by omitting the contiguity constraints and replacing the objective function with a simpler expression, in which each unit-duration portion of each job has an associated cost coefficient:

$$(\mathbf{TR})\quad \min \sum_j \sum_{\substack{k\in H\\ k\ge r_j+1}} c_{jk} X_{jk}, \tag{2.7}$$

$$\sum_{\substack{k\in H\\ k\ge r_j+1}} X_{jk} = p_j, \quad \forall j, \tag{2.8}$$

$$\sum_{\substack{j\\ k\ge r_j+1}} X_{jk} \le 1, \quad \forall k\in H, \tag{2.9}$$

$$X_{jk}\ge 0, \quad \forall j, k\in H,\ k\ge r_j+1. \tag{2.10}$$

This new formulation is equivalent to a transportation problem. Hence, the binary constraints do not need to be stated explicitly, and the problem can be solved very efficiently. The idea underlying our proposed cost structure is intuitive. If a non-preemptively scheduled job j finishes exactly at d_j, and if we think of each of the p_j unit-duration segments in job j as different jobs, then the average completion time of these unit-duration segments is

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

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

$$c_{jk} = \begin{cases} \frac{\epsilon_j}{p_j}\left[\left(d_j-\frac{p_j}{2}\right)-\left(k-\frac{1}{2}\right)\right], & k\le d_j,\\ \frac{\pi_j}{p_j}\left[\left(k-\frac{1}{2}\right)-\left(d_j-\frac{p_j}{2}\right)\right], & k> d_j. \end{cases} \tag{2.11}$$

Below we provide our main result in this section: a proof that the solution of TR with the coefficients given above provides a lower bound on the optimal objective function value of P1. Furthermore, in Sect. 3.2 we prove that this set of cost coefficients is a very good candidate to maximize the value of the lower bound obtained from TR. Now let S_{P} represent a feasible schedule for problem P with a total cost of $TC(S_{\mathrm{P}})$. An optimal schedule is denoted by an asterisk.

Theorem 2.2 *The optimal objective value of TR, $TC(S^*_{\mathrm{TR}})$, is a lower bound on the optimal objective value $TC(S^*_{\mathrm{P1}})$ of P1.*

Proof We show that for any optimal solution S^*_{P1} for P1, there exists a corresponding feasible schedule S_{TR} for TR such that $TC(S_{\mathrm{TR}}) \le TC(S^*_{\mathrm{P1}})$. In particular, we consider a solution S_{TR} for TR constructed by converting S^*_{P1} into a feasible solution of TR. This is accomplished by dividing each job in S^*_{P1} into contiguous unit-duration segments. We demonstrate that for a schedule S_{TR} constructed in this manner, $TC(S_{\mathrm{TR}}) \le TC(S^*_{\mathrm{P1}})$. Clearly, an optimal solution S^*_{P1} for P1 exists in which all job completion times belong to $H = \{k \mid k \in \mathbb{Z}, k \in [t_{\min}, t_{\max}]\}$, which is the same time horizon considered in problem TR. Our strategy is to consider each job in S^*_{P1} separately. If $C_j \le d_j$ in S^*_{P1}, then the cost that job j incurs in S_{TR} is given by:

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

as in S^*_{P1}. If job j is tardy in S^*_{P1}, then we need to distinguish between two cases. If $C_j \ge d_j + p_j$, then the cost that job j incurs in S_{TR} is given by:

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

as in S^*_{P1}. However, if $d_j + 1 \le C_j \le d_j + p_j - 1$ when $p_j \ge 2$, then $x = C_j - d_j$ unit jobs of job j incur a tardiness cost in S_{TR}, while the remaining $(p_j - x)$ unit jobs incur an earliness cost. In this case the cost incurred by job j in S_{TR} consists of two parts:

$$\sum_{k=C_j-p_j+1}^{C_j} c_{jk} = \sum_{k=d_j+x-p_j+1}^{d_j+x} c_{jk}$$
$$= \sum_{k=d_j+x-p_j+1}^{d_j} c_{jk} + \sum_{k=d_j+1}^{d_j+x} c_{jk}, \qquad (2.12)$$

where $1 \le x \le p_j - 1$. We first examine the costs incurred by the unit jobs completed at or before d_j:

$$\sum_{k=d_j+x-p_j+1}^{d_j} c_{jk} = \frac{\epsilon_j}{p_j}\left[\frac{-xp_j}{2} + \frac{x^2}{2}\right]$$
$$= \frac{\epsilon_j}{p_j}\left[\frac{x}{2}(x - p_j)\right] < 0$$

$$\text{(because } x < p_j). \qquad (2.13)$$

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

$$\sum_{k=d_j+1}^{d_j+x} c_{jk} = \frac{\pi_j}{p_j}\left[\frac{x}{2}(x + p_j)\right] = \pi_j x \left[\frac{x + p_j}{2p_j}\right] < \pi_j x$$

$$\text{(because } x < p_j). \qquad (2.14)$$

Therefore, we have $\sum_{k=C_j-p_j+1}^{C_j} c_{jk} < \pi_j x = \pi_j (C_j - d_j)$ when $d_j + 1 \le C_j \le d_j + p_j - 1$. Finally, summing over all jobs, we obtain $TC(S^*_{\mathrm{TR}}) \le TC(S_{\mathrm{TR}}) \le TC(S^*_{\mathrm{P1}})$, as desired, because the cost incurred by any job j in S_{TR} is no larger than that in S^*_{P1}. □

Note that we must account for the possibility that some unit jobs may be early and some others may be tardy for a job j of duration $p_j > 1$ in a non-preemptive feasible solution of TR. This property complicates the task of finding an appropriate cost structure, and the proof of Theorem 2.2 provides a critical insight: job j incurs the same cost in S^*_{P1} and S_{TR}, unless it finishes in the time interval $[d_j + 1, d_j + p_j - 1]$. We use this property in proving some of our results in Sect. 3, where we derive the cost that job j incurs in S_{TR} when $d_j \le C_j \le d_j + p_j$. From (2.12–2.14), we have

$$\sum_{k=C_j-p_j+1}^{C_j} c_{jk} = f(x) = \sum_{k=d_j+x-p_j+1}^{d_j+x} c_{jk}$$
$$= \frac{1}{2}\left(\frac{\epsilon_j}{p_j} + \frac{\pi_j}{p_j}\right)x^2 + \frac{1}{2}(\pi_j - \epsilon_j)x, \qquad (2.15)$$

where $0 \le x = C_j - d_j \le p_j$.

Sourd and Kedad-Sidhoum (2003) propose a lower bound similar to ours for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$ based on a transportation problem with the following cost coefficients:

$$c'_{jk} = \begin{cases} \left\lfloor \frac{(d_j-k)}{p_j} \right\rfloor \epsilon_j, & k \le d_j - p_j, \\ 0, & d_j - p_j + 1 \le k \le d_j, \\ \left\lceil \frac{(k-d_j)}{p_j} \right\rceil \pi_j, & k \ge d_j + 1. \end{cases} \qquad (2.16)$$

Note that these coefficients form a (discrete) step function, and they stay constant for p_j consecutive periods. So they have a different structure than our cost coefficients given in (2.11). Also, the cost coefficients in (2.16) satisfy $\sum_{k=C_j-p_j+1}^{C_j} c'_{jk} = \epsilon_j E_j + \pi_j T_j$ for all possible completion times C_j. Recall that this condition is not satisfied by our cost coefficients when job j finishes in the time interval $[d_j + 1, d_j + p_j - 1]$. However, we note that $c'_{jk} < c_{jk}$ for some periods k. Therefore, it is not clear *a priori* which set of cost coefficients will perform better in practice. In Sect. 5, we compare these cost coefficients computationally, and conclude that TR generally yields tighter lower bounds when our cost coefficients in (2.11) are used.

Sourd (2004) develops a polynomial-time lower bound for the problem $1//\sum \epsilon_j E_j + \pi_j T_j$ based on the continuous assignment problem (see Sect. 1). In other words, he allows a job to be preempted at any point in time, and defines a cost function $f_j(t)$ such that $\int_{C_j-p_j}^{C_j} f_j(t)\,dt \le \epsilon_j E_j + \pi_j T_j$ for all possible completion times C_j. In particular, he proposes using

$$f_j(t) = \begin{cases} -\frac{\epsilon_j}{2} + \frac{\epsilon_j}{p_j}(d_j - t), & t \le d_j, \\ \frac{\pi_j}{2} + \frac{\pi_j}{p_j}(t - d_j), & t > d_j, \end{cases} \tag{2.17}$$

when $\epsilon_j > \pi_j$, and a slightly different cost function when $\epsilon_j \le \pi_j$. We observe that our cost coefficients in (2.11) can be obtained by setting:

$$c_{jk} = \int_{k-1}^{k} f_j(t)\,dt. \tag{2.18}$$

Hence, we have $\int_{C_j-p_j}^{C_j} f_j(t)\,dt = \sum_{k=C_j-p_j+1}^{C_j} c_{jk}$. In other words, our "discrete" lower bound is closer to the "continuous" lower bound of Sourd (2004) than to the discrete lower bound presented in Sourd and Kedad-Sidhoum (2003). Sourd (2004) reports that the lower bound based on the continuous assignment problem and (2.17) generally performs better than the discrete lower bound based on (2.16), which also agrees with our computational experience in Sect. 5. Furthermore, in Sect. 3.2 we prove that the cost coefficients given in (2.11) are, in some sense, the optimal cost coefficients for TR when the cost coefficients are restricted to lie on a piecewise linear function with only two segments (see Theorem 3.2). So there is both theoretical and practical justification for using our cost coefficients in order to obtain a good lower bound for P1.

In TR there are $n + (t_{\max} - t_{\min} + 1)$ constraints and at most $n * (t_{\max} - t_{\min} + 1)$ variables. It is possible, however, to state an equivalent transportation problem with at most $n + 2nP$ constraints and at most $2nP$ variables by observing that there exists an optimal solution to TR such that X_{jk} must be zero if k does not belong to the set $H_j = \{k \mid k \in \mathbb{Z}, k \in [\max(r_j + 1, d_j - P + 1), d_j + P]\}$. This property is similar to Lemma 2.1 in Verma and Dessouky (1998). In our computational experiments, we observed that although this property does not usually change the number of nodes in the transportation problem (the sum of the number of jobs and the number of time periods in the planning horizon), it decreases the number of arcs (number of variables) by 30 to 50%, depending on the problem instance.

From a theoretical perspective, obtaining a lower bound for P1 from TR is accomplished in pseudo-polynomial time, since the planning horizon depends on the sum of the processing times (also see Sect. 1). However, in practice there are several advantages to this approach that allow us to develop effective algorithms for solving P1 based on the optimal solution of TR. First, the transportation problem can be solved quickly even for very large instances. Sourd and Kedad-Sidhoum (2003) note that all p_j unit jobs of a job j are identical, and they exploit this property to develop an adaptation of the Hungarian algorithm for solving their transportation problem. They reduce the complexity of the Hungarian algorithm from $O(|H|^3)$ to $O(n^2|H|)$ for this class of transportation problems, where $|H|$ is the number of periods in the planning horizon. (We do not use this specialized algorithm in our computational experiments.) Second, the optimal solution of the transportation problem allows us to construct several feasible solutions for P1 with very little additional computational effort once problem TR is solved. Clearly, this increases the likelihood of finding a near-optimal feasible solution for P1 (see Sects. 4 and 5). Finally, we observe that the size of the transportation problem is relatively smaller for computationally hard instances of P1. Intuitively, if the due dates are close to each other and jobs compete with each other to be scheduled in the same time periods, then such an instance is harder to solve. However, when the due dates are close to each other, then we have a relatively smaller instance of the transportation problem.

3 Properties of the lower bound

In this section we characterize a portion of the gap between the objective from our lower bound, $TC(S^*_{\mathrm{TR}})$, and the optimal objective value. We observe that any gap between the cost of S^*_{TR} and that of S^*_{P1} is partly due to preemption in the transportation solution and partly due to the different cost structures. In this section we explore the second effect in detail.

First, consider a single-job instance of problem P1. Clearly, if there is only a *single* job j and $r_j + p_j \le d_j$, then for all possible values of $\epsilon_j > 0$ and $\pi_j > 0$, job j is scheduled in the interval $[d_j - p_j, d_j]$ in an optimal solution S^*_{P1} with a total cost of zero. Similarly, job j is scheduled non-preemptively in the interval $[d_j - p_j, d_j]$ in an optimal transportation solution S^*_{TR} with a total cost of zero if $\pi_j \ge \epsilon_j$. We can show this by using the convexity of f

and $f'(0) \geq 0$, where $f(x)$ is the cost incurred by job j in TR when it is scheduled non-preemptively in the interval $[d_j + x - p_j, d_j + x]$, as described in (2.15). However, when $\pi_j < \epsilon_j$, there is an incentive to complete job j strictly after its due date in TR. Note that $f'(0) < 0$ if $\pi_j < \epsilon_j$. In fact, when $p_j > 1$, we can easily create single-job instances for which S^*_{TR} differs from S^*_{P1}, and we can make $TC(S^*_{\mathrm{TR}})$ arbitrarily smaller than $TC(S^*_{\mathrm{P1}})$ by increasing the difference $(\epsilon_j - \pi_j)$.

From the discussion above we conclude that, as long as $\epsilon_j \leq \pi_j\ \forall j$, the optimal objective value of TR cannot be strictly less than zero, and we cannot create pathological instances for which TR yields arbitrarily poor lower bounds. We observe that the cost structure $\epsilon_j \leq \pi_j\ \forall j$ implies that the cost of loss of customer goodwill per unit time is at least as expensive as the finished goods inventory holding cost per unit time. In general, we would expect this to be reasonable for most just-in-time systems, as just-in-time usually implies minimizing inventory costs, subject to meeting customer due dates. However, we emphasize again that our lower bound and our heuristics for P1 do not make any specific assumption about the cost structure. In fact, our computational results in Sect. 5.3.3 indicate that our algorithms are not sensitive to it.

Next, for any given non-preemptive schedule we characterize the largest possible difference between its associated objective function values in problems TR and P1. Then, in Sect. 3.2, we prove that the cost coefficients developed for TR in the previous section are, in some sense, the best possible cost coefficients.

3.1 Cost function difference for identical schedules

In Sect. 2 we proved that $TC(S^*_{\mathrm{TR}})$ is a lower bound on $TC(S^*_{\mathrm{P1}})$. In this section we give two characterizations of the portion of the gap attributable to the difference between the objective functions of P1 and TR. To motivate the discussion, suppose that we develop an algorithm that converts S^*_{TR} into a non-preemptive schedule S_{P1}, and we are interested in computing a bound on the worst case difference between $TC(S_{\mathrm{P1}})$ and $TC(S^*_{\mathrm{TR}})$. To accomplish this, one would need to account for both the increase in the objective of TR while constructing S_{P1} from S^*_{TR}, and the increase in cost from applying the original objective function to S_{P1}. The following lemma characterizes the latter increase both in terms of the absolute and relative difference in cost, when the objective functions of TR and P1 are applied to the same (not necessarily optimal) non-preemptive schedule. In this lemma, if job j completes processing at time C_j in a feasible schedule for the original non-preemptive problem P1, then $X_{jk} = 1$ for all $k = C_j - p_j + 1, \ldots, C_j$ in the corresponding transportation solution S_{TR}. Refer to Bülbül (2002) for a proof.

Lemma 3.1 *Let S_{P1} be a feasible schedule for the original non-preemptive problem P1 with a cost of $TC(S_{\mathrm{P1}})$, and let S_{TR} be the corresponding transportation solution with a cost of $TC(S_{\mathrm{TR}})$. Then*

$$TC(S_{\mathrm{P1}}) - TC(S_{\mathrm{TR}}) \leq \frac{1}{8}\sum_{j=1}^{n}(\pi_j + \epsilon_j)p_j. \tag{3.1}$$

If, in addition, $\epsilon_j \leq \pi_j,\ \forall j$, then we also have

$$0 \leq TC(S_{\mathrm{TR}}) \leq TC(S_{\mathrm{P1}}) \leq p_{\max} * TC(S_{\mathrm{TR}}). \tag{3.2}$$

The proof of (3.2) also suggests that our lower bound becomes tighter, when the tardiness costs increase relative to the earliness costs. This implies that our lower bound may also be useful for solving weighted tardiness problems. In addition, according to (3.2), the optimal objective function value of TR is the same as that of P1 if all jobs are of unit length, i.e., $p_{\max} = 1$. Note that this is true even if there exists a job j such that $\epsilon_j > \pi_j$ (see Verma and Dessouky 1998). Finally, we note that the bound provided in (3.2) has the desirable property of being independent of the earliness and tardiness costs.

3.2 Optimal linear cost coefficient function

At the beginning of Sect. 3 we discussed that the difference between the cost function of TR (for a single job) and the cost function of P1 can be relatively large in the worst case. A natural question to ask is whether we can find cost coefficients that will decrease the difference between the costs incurred by the same non-preemptive schedule in P1 and in TR, and, thereby, improve the bounds given in Lemma 3.1. We restrict ourselves to linear cost coefficient functions, as defined below. This choice has a natural appeal for three reasons. First, the original problem P1 also has a linear penalty structure. Second, the lower bound obtained from TR based on the piecewise linear cost structure in (2.11), and the non-preemptive schedules adapted from S^*_{TR} perform very well, as we demonstrate in the computational experiments. Finally, linearity enables us to use linear programming duality to prove our result. Specifically, we look for cost coefficients c_{jk} for job j that satisfy the following conditions, where m_E and m_T represent the slope of the cost function before and after the due date d_j, respectively

$$c_{jk} = c_{jd_j} + (d_j - k) * m_E, \quad \forall k \leq d_j, \tag{3.3}$$

$$c_{jk} = c_{jd_j+1} + (k - d_j - 1) * m_T, \quad \forall k \geq d_j + 1, \tag{3.4}$$

$$m_E, m_T \geq 0. \tag{3.5}$$

Next, we need to choose an appropriate objective function that reflects how "good" the cost coefficients are. Note that all we know about the completion time C_j of job j in an optimal non-preemptive schedule is that it lies within some interval H (see Sect. 2). This implies that we need cost coefficients that perform well over a range of possible completion times rather than for a specific completion time. Now assume that C_j lies in some interval $[t^j_{\min}, t^j_{\max}]$ that includes d_j, and let $TC_j(S_{\mathrm{P}})$ denote the cost incurred by job j in problem P. As alternatives for the objective function, we consider minimizing the maximum or the total difference of $TC_j(S_{\mathrm{P1}}) - TC_j(S_{\mathrm{TR}})$ across all C_j values in the interval $[t^j_{\min}, t^j_{\max}]$. Note that we concern ourselves with the cost function of a single job here, and we can, therefore, restrict our attention to non-preemptive schedules without loss of generality. We observed computationally that the optimal cost coefficients are dependent on the choices of $t^j_{\min}$ and $t^j_{\max}$ for the minimax criterion. However, as we prove in this section, the optimal cost coefficients for the minisum objective are not contingent on the specific values of these two parameters under some mild conditions. Additionally, by solving several instances, we also observed that the optimal solution for the minimax criterion becomes more similar to the optimal solution of the minisum criterion as $t^j_{\max} - t^j_{\min}$ increases. Therefore, we use the minisum criterion and characterize the structure of its optimal solution in the following discussion.

The problem of finding cost coefficients that satisfy (3.3) through (3.5) and minimize the objective function $\sum_{C_j \in [t^j_{\min}, t^j_{\max}]}(TC_j(S_{\mathrm{P1}}) - TC_j(S_{\mathrm{TR}}))$ when job j is scheduled non-preemptively to finish at the same time both in S_{P1} and S_{TR} can be represented as a linear programming problem. Let a_E and a_T denote c_{jd} and c_{jd+1}, respectively, and D_t be $TC_j(S_{\mathrm{P1}}) - TC_j(S_{\mathrm{TR}})$ if job j finishes processing at time t. Also, let $t^j_{\min}$ and $t^j_{\max}$ be the smallest and largest possible completion times of job j, respectively. In general, we allow $t^j_{\min} \geq r_j + p_j$ and $t^j_{\max} \leq d_j + P$, and later in the discussion we identify some mild conditions that need to be imposed on $t^j_{\min}$ and $t^j_{\max}$. In the following linear programming problem we omit the job index j to simplify the notation:

$$(\mathbf{CP}) \quad \min \sum_{t=t_{\min}}^{t_{\max}} D_t, \tag{3.6}$$

$$(\gamma_t) \quad D_t + \sum_{k=t-p+1}^{t} \big(a_E + (d-k)m_E\big) = \epsilon(d-t),$$

$$t = t_{\min}, \ldots, d, \tag{3.7}$$

$$(\delta_t) \quad D_t + \sum_{k=t-p+1}^{d} \big(a_E + (d-k)m_E\big)$$

$$+ \sum_{k=d+1}^{t} \big(a_T + (k-d-1)m_T\big) = \pi(t-d),$$

$$t = d+1, \ldots, d+p-1, \tag{3.8}$$

$$(\mu_t) \quad D_t + \sum_{k=t-p+1}^{t} \big(a_T + (k-d-1)m_T\big) = \pi(t-d),$$

$$t = d+p, \ldots, t_{\max}, \tag{3.9}$$

$$a_E, a_T \text{ unrestr.}, \tag{3.10}$$

$$m_E, m_T \geq 0, \tag{3.11}$$

$$D_t \geq 0, \quad t = t_{\min}, \ldots, t_{\max}. \tag{3.12}$$

Constraints (3.7) through (3.9) define the difference in period t, D_t, between the cost functions of TR and P1 for different completion times of job j. By (3.12), $TC_j(S_{\mathrm{P1}}) - TC_j(S_{\mathrm{TR}})$ is guaranteed to be nonnegative for all possible completion times of job j. Thus, $TC(S^*_{\mathrm{TR}})$ yields a lower bound on $TC(S^*_{\mathrm{P1}})$ for all objective function coefficients c_{jk}, generated from any feasible solution of CP. The Greek letters on the left are the dual variables associated with the constraints.

Now we show that the optimal solution to CP corresponds to the cost coefficients (2.11) when $t_{\min} \leq d - p$ and $t_{\max} \geq d + 2p$. In practice, these conditions are automatically satisfied for the great majority of problem instances. From the definition of the planning horizon H, it follows that all we require is $r_j + p_j \leq d_j - p_j$ and $d_j + P \geq d_j + 2p_j$, i.e., $d_j - r_j \geq 2p_j$ and $P \geq 2p_j$.

Theorem 3.2 *If $t_{\min} \leq d - p$ and $t_{\max} \geq d + 2p$, then the optimal solution to CP is given by $m_E = \frac{\epsilon}{p}$, $m_T = \frac{\pi}{p}$, $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, and $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$, which correspond to the cost coefficients*

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

given in (2.11). The optimal objective function value is $\frac{1}{12}(\epsilon + \pi)(p^2 - 1)$.

Proof Our strategy is to find a feasible dual solution with the same objective value given above. It follows directly from Theorem 2.2 that $m_E = \frac{\epsilon}{p} \geq 0$, $m_T = \frac{\pi}{p} \geq 0$, $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$, and the corresponding D_t values are feasible for CP. The resulting objective function value is $\sum_{t=t_{\min}}^{t_{\max}} D_t = \sum_{x=1}^{p-1}(\pi x - f(x)) = \frac{1}{12}(\epsilon + \pi)(p^2 - 1)$, using (2.15) and the identities $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$ and $\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$. The dual of CP is:

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

$$+ \sum_{t=d+p}^{t_{\max}} \pi(t-d)\mu_t, \tag{3.13}$$

$$(D_t) \qquad \gamma_t \le 1, \quad t = t_{\min}, \ldots, d, \tag{3.14}$$

$$(D_t) \qquad \delta_t \le 1, \quad t = d+1, \ldots, d+p-1, \tag{3.15}$$

$$(D_t) \qquad \mu_t \le 1, \quad t = d+p, \ldots, t_{\max}, \tag{3.16}$$

$$(a_E) \qquad \sum_{t=t_{\min}}^{d} p\gamma_t + \sum_{t=1}^{p-1}(p-t)\delta_{d+t} = 0, \tag{3.17}$$

$$(a_T) \qquad \sum_{t=1}^{p-1} t\delta_{d+t} + \sum_{t=d+p}^{t_{\max}} p\mu_t = 0, \tag{3.18}$$

$$(m_E) \qquad \sum_{t=t_{\min}}^{d} \sum_{k=t-p+1}^{t} (d-k)\gamma_t + \sum_{t=d+1}^{d+p-1} \sum_{k=t-p+1}^{d} (d-k)\delta_t \le 0, \tag{3.19}$$

$$(m_T) \qquad \sum_{t=d+1}^{d+p-1} \sum_{k=d+1}^{t} (k-d-1)\delta_t + \sum_{t=d+p}^{t_{\max}} \sum_{k=t-p+1}^{t} (k-d-1)\mu_t \le 0, \tag{3.20}$$

$$\gamma_t \text{ unrestr.}, \quad t = t_{\min}, \ldots, d, \tag{3.21}$$

$$\delta_t \text{ unrestr.}, \quad t = d+1, \ldots, d+p-1, \tag{3.22}$$

$$\mu_t \text{ unrestr.}, \quad t = d+p, \ldots, t_{\max}. \tag{3.23}$$

The corresponding primal variable for each constraint is indicated in parentheses on the left. A proof of the following lemma is presented by Bülbül (2002).

Lemma 3.3 *If $t_{\min} \le d-p$ and $t_{\max} \ge d+2p$, then the following solution is feasible for DCP.*

$$\gamma_{t_{\min}} = -\gamma_d - \frac{p-1}{2} - K + 1, \quad \text{where } K = d - t_{\min}, \tag{3.24}$$

$$\gamma_t = 1, \quad t_{\min}+1 \le t \le d-1, \tag{3.25}$$

$$\gamma_d = \frac{-(p+1)(p-1)}{12K} - \frac{K+p}{2} + 1, \tag{3.26}$$

$$\delta_t = 1, \quad d+1 \le t \le d+p-1, \tag{3.27}$$

$$\mu_{d+p} = \frac{\frac{N^2}{2} + N(\frac{-p}{2} - 1) + (p-1)(\frac{p+13}{12}) + 1}{-N+p}, \quad \text{where } N = t_{\max} - d, \tag{3.28}$$

$$\mu_t = 1, \quad d+p+1 \le t \le t_{\max} - 1, \tag{3.29}$$

$$\mu_{t_{\max}} = -\mu_{d+p} - \frac{p-1}{2} - N + p + 1. \tag{3.30}$$

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

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

$$\sum_{t=d+1}^{d+p-1} (t-d)\delta_t = \sum_{t=1}^{p-1} t = \frac{p(p-1)}{2},$$

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

Therefore, the objective function value for the dual feasible solution given in (3.24–3.30) is:

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

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

Algorithm 1 The Switch Heuristic

```
Require: Given an optimal transportation schedule $S^*_{TR}$, apply this algorithm to each block $B$.
1:  Initialize:
2:  For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^{TR^*}_{j_i}$.
3:  Let $Q_B$ be the set of split jobs in $B$.
4:  Main routine:
5:  while $Q_B \neq \emptyset$ do
6:      $k = \arg\min_{j \in Q_B}\{C_{j_{[1]}}\}$.
7:      SHIFT-EARLY($B$, $k$) and let the resulting schedule be $S^E_{TR}$.      {Try shifting early.}
8:      SHIFT-LATE($B$, $k$) and let the resulting schedule be $S^L_{TR}$.      {Try shifting late.}
9:      $\Delta^E = TC(S^E_{TR}) - TC(S^*_{TR})$ and $\Delta^L = TC(S^L_{TR}) - TC(S^*_{TR})$      {Compute cost increases.}
10:     if $\Delta^E \leq \Delta^L$ then
11:         For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^E_{j_i}$.
12:     else
13:         For each job $j \in B$ and for each segment $j_i \in j$, $C_{j_i} = C^L_{j_i}$.
14:     Update $Q_B$.      {A single shift can bring together several split jobs.}
```

Note that the conditions $t_{\min} \leq d - p$ and $t_{\max} \geq d + 2p$ are sufficient, but not necessary, for the coefficients in Theorem 3.2 to be optimal. For instance, assume $d = 10$, $p = 4$, $\epsilon = 1$, $\pi = 1$. Choose $t_{\min} = 8 > d - p$ and $t_{\max} = 14 < d + 2p$. Then the optimal solution to CP is $m_E = 0.25$, $m_T = 0$, $a_E = -0.375$, and $a_T = 1$. These values satisfy $m_E = \frac{\epsilon}{p}$ and $a_E = \frac{\epsilon}{p}(\frac{-p}{2} + \frac{1}{2})$, but not $m_T = \frac{\pi}{p}$ and $a_T = \frac{\pi}{p}(\frac{p}{2} + \frac{1}{2})$, i.e., the cost coefficients characterized in Theorem 3.2 are not optimal. The planning horizon extends for such a short duration after the due date that the optimal cost coefficients are constant in the interval [11, 14]. However, if $t_{\max}$ is increased to $d + 2p = 18$, then the cost coefficients in Theorem 3.2 are optimal, even if $d - t_{\min} < p$.

4 The switch heuristic (SW)

As discussed in Sect. 1, the quality of feasible solutions for P1 are mainly determined by the job sequencing decisions. Therefore, in this section we develop an algorithm to construct a feasible solution for P1, where the processing sequence is derived from the characteristics of the optimal solution for TR. As in the original non-preemptive problem P1, any solution for TR can be characterized by "blocks" of processing with idle time between adjacent blocks. In the context of TR, we define a block $\mathcal{B}$ as a sequence of *unit* jobs processed contiguously. Clearly, in an optimal solution S^*_{TR} for TR, all unit jobs of job j must belong to the same block $\mathcal{B}$ that starts no later than d_j. Similarly, the last unit job in $\mathcal{B}$ cannot finish earlier than d_j. Therefore, as in the Switch Heuristic presented here, one can construct a feasible solution for P1 by re-arranging the unit jobs within each block independently of the other blocks and without changing the start and completion times of the blocks.

For many instances of this problem we observed that, within a block, those jobs that contribute most to the overall cost are scheduled very close to the same time in both S^*_{P1} and S^*_{TR}, while less expensive jobs are frequently spread out in S^*_{TR}. Typically, in S^*_{TR} the unit jobs of expensive jobs are grouped tightly together, while those of less expensive jobs are distributed around them. In S^*_{P1} these less expensive jobs either precede or succeed such a group of expensive jobs. This observation motivates the Switch Heuristic (see Algorithm 1, as well as the subroutine Algorithm 2). In particular, this heuristic chooses a job (called a *split* job) that is preempted at least once by another job in the current schedule, and rearranges its unit jobs while attempting to minimize the increase in the total cost of the block from its original value in S^*_{TR}. For each split job, all of its unit jobs are moved either next to its first or last unit job completed. This strategy aims to minimize the impact on the more expensive jobs in the middle by shifting split jobs alternately early and late. As a result, less expensive jobs that are more spread out in S^*_{TR} are shuffled around, while in the final non-preemptive schedule, more expensive jobs remain close to their original locations in S^*_{TR}.

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

The two subroutines of the Switch Heuristic, SHIFT-EARLY and SHIFT-LATE, shift a job adjacent to its earliest unit job, or its latest unit job, respectively. We define a

Algorithm 2 Subroutine SHIFT-EARLY(B, k)

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

temporary queue K, for which the sorting discipline in the queue is first in, first out. Note that $Enqueue(K, j_i)$ adds unit job j_i to the queue, and $Dequeue(K)$ removes the first unit job in the queue. In addition, let $B(t)$ be the unit job j_i that is processed during the time interval $[t-1, t]$ in the current schedule, and let $J(t)$ be the associated job j. Only the pseudo-code of the subroutine SHIFT-EARLY is given here because conceptually the only difference between SHIFT-EARLY and SHIFT-LATE is that in the latter case we need to check whether the schedules of the other jobs remain feasible while moving the unit jobs of the split job later.

The algorithm terminates in at most $|Q_B|$ iterations for each block because each call to the shift subroutines brings at least one split job together and never splits a job that has already been scheduled non-preemptively. Finally, when the Switch Heuristic terminates, we recover the job processing sequence and insert idle time optimally in order to obtain the final non-preemptive schedule.

5 Computational experiments

We performed a variety of computational experiments in order to evaluate the performance of both the lower bound and a set of heuristics for P1. The set of heuristics consists of the Switch Heuristic, described in Sect. 4, as well as the following simple heuristics, each of which constructs a sequence based on the elements of the optimal solution for TR, which, in general, is a preemptive schedule. We apply an optimal timing algorithm to each heuristic sequence in order to determine the final schedule (see Sect. 1).

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

The LCT and MCT heuristics are related to the α-point concept introduced by Phillips et al. (1998). An α-point of job j in a preemptive schedule is defined as the earliest point in time at which α-fraction of job j is completed. Note that Sourd and Kedad-Sidhoum (2003) obtain an initial upper bound at the root node of their B&B tree by scheduling jobs in non-decreasing order of their α-points, where $\alpha = 0.5$.

We designed our computational experiments with two main objectives. First of all, we would like to demonstrate that our algorithms find good solutions and find them quickly. Our other major concern is the robustness of our algorithms. In particular, we use the information from a preemptive relaxation to generate non-preemptive schedules, and among the factors that affect preemption in the transportation problem are the processing times and the number of jobs. Thus, we are interested in the behavior of our lower bound and the heuristics as processing times and the number of jobs increase.

In Sect. 3 we identified a theoretical drawback of our approach using a single job example where the earliness cost is strictly greater than the tardiness cost. However, our computational results indicate that such cases do not happen in practice when there are many jobs competing for the same time periods. To observe the effect of the cost structure, we use the data sets from Mazzini and Armentano (2001), and we also benchmark our results against theirs.

Finally, we compare our lower bound to another lower bound that is obtained from the linear programming relaxation of a time-indexed integer linear programming formulation for P1 which we call TIP1. We argue in Sect. 5.1 that the bound obtained from the LP relaxation of TIP1 dominates our lower bound in terms of objective value, but the improved bound comes at a significant additional computational cost.

5.1 Another lower bound

It is well known that the LP relaxations of time-indexed integer linear programming formulations of single machine scheduling problems, first introduced by Dyer and Wolsey (1990), provide very tight bounds. In this section, we argue that the lower bound obtained from the LP relaxation of TIP1 dominates the lower bound from TR. However, the tightness of the LP relaxation comes at a significant additional computational cost, as we demonstrate in the computational experiments. The lower bound obtained from the LP relaxation of TIP1 is useful for instances that we could not solve to optimality. It allows us to compute a tight upper bound on the optimality gaps of our heuristic solutions in such instances.

From Sects. 2 and 3, recall that $t_{\min} = \min_j r_j + 1$, $t_{\max} = \max_j \max(r_j, d_j) + P$, and the planning horizon is defined as $H = \{k \mid k \in \mathbb{Z}, k \in [t_{\min}, t_{\max}]\}$. In this section, let $t^j_{\min} = r_j + p_j$. Also, define z_{jk} as a binary variable that takes the value 1 if job j finishes processing at time k, and zero, otherwise. Then the time-indexed formulation of P1 is:

$$\textbf{(TIP1)}\quad \min \sum_j \sum_{\substack{k\in H\\ k\ge t^j_{\min}}} z_{jk}(\epsilon_j(d_j-k)^+ + \pi_j(k-d_j)^+), \tag{5.1}$$

$$\sum_j \sum_{t=\max(k,t^j_{\min})}^{\min(t_{\max},k+p_j-1)} z_{jt} \le 1, \quad \forall k \in H, \tag{5.2}$$

$$\sum_{\substack{k\in H\\ k\ge t^j_{\min}}} z_{jk} = 1, \quad \forall j, \tag{5.3}$$

$$z_{jk} \in \{0,1\}, \quad \forall j,\ k \in H, k \ge t^j_{\min}. \tag{5.4}$$

The constraints (5.2) and (5.4) together prescribe that at most one job is processed by the machine at any point in time and that processing is non-preemptive. Constraints (5.3) ensure that all jobs are processed.

Now consider the linear programming relaxation LP(TIP1) of TIP1. A fractional value α for z_{jk} implies that job j is processed on the machine in *each* of the p_j unit-length time periods $[k-p_j, k-p_j+1], \dots, [k-1, k]$, but only for a fraction α in any such period. This implies that LP(TIP1) is a more constrained version of TR that imposes contiguity constraints on X_{jk}'s that belong to the same job. By using this observation and Theorem 2.2, one can show that for each feasible solution of LP(TIP1) there exists a feasible solution for TR with no greater objective function value. Thus, LP(TIP1) provides a tighter bound than TR. Refer to Bülbül (2002) for a formal proof of Theorem 5.1.

Theorem 5.1 $TC(S^*_{\mathrm{TR}}) \le TC(S^*_{\mathrm{LP(TIP1)}}) \le TC(S^*_{\mathrm{P1}})$.

5.2 Data generation

To generate the processing times and the due dates of jobs for our computational study, we follow the popular scheme of Potts and van Wassenhove (1982). This method is motivated by the observation that the mean and the range of the due date distribution determines the level of difficulty of scheduling problems with due dates (also, see Hall and Posner 2001).

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

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

5.3 Summary of results

The solution procedure for TR and the heuristics are implemented in C using `CPLEX 9.0` callable libraries. Whenever possible, TIP1 and/or its linear programming relaxation are solved by `ILOG AMPL 9.0` and the solver `CPLEX 9.0`. All computations are performed on a Sun Blade 1000 workstation with 512 MB of memory. In the following sections, we examine the effects of the number of jobs, processing times and the relationship between the earliness and tardiness costs on the performance of our algorithms. We also benchmark our algorithms against results published in the literature.

5.3.1 Number of jobs

In order to test whether increasing the number of jobs affects the performance of the lower bound and the heuristics, we

Table 1 Effect of the number of jobs: problem parameters

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

Table 2 Effect of the number of jobs: CPU times, number of optimal solutions, and average and worst case gaps of heuristics and bounds

n	CPU time (sec.)					# Opt.[†]		Percentage gap (%)						
	B&B	TL	LP	BH	LP/BH	BH	LP	LP[†]	TR[†]	LCT	ACT	MCT	SW	BH
20	**0.59**	**0**	**0.09**	**0.03**	**2.53**	**6**	**10**	**−0.86**	**−5.08(−5.49)**	**12.27**	**8.64**	**5.22**	**4.53**	**2.45(3.18)**
	2.08		0.21	0.05	5.25			−4.74	−14.75(−12.92)	65.36	37.89	25.77	18.13	12.76(15.35)
40	**10.17**	**0**	**0.89**	**0.10**	**8.42**	**0**	**1**	**−0.69**	**−2.86(−3.40)**	**8.87**	**7.64**	**4.19**	**3.47**	**2.36(3.01)**
	69.00		2.48	0.15	22.55			−2.15	−7.90(−8.60)	32.11	28.35	18.59	15.25	10.15(10.90)
60	**170.62**	**3**	**3.54**	**0.18**	**19.82**	**0**	**0**	**−0.59**	**−2.10(−2.56)**	**7.81**	**7.23**	**3.77**	**2.92**	**2.27(2.77)**
	1801.37		10.09	0.24	48.11			−2.40	−6.76(−7.34)	37.63	34.16	20.40	17.30	9.39(7.70)
80	**528.60**	**15**	**9.27**	**0.30**	**30.24**	**0**	**0**	**−0.45**	**−1.68(−2.06)**	**7.71**	**6.85**	**3.56**	**2.58**	**2.15(2.67)**
	1802.03		23.91	0.39	69.11			−1.65	−6.02(−5.68)	48.16	34.18	16.98	8.28	7.18(8.62)

[†]For $n = 60$ (80), 97 (85) instances out of 100 solved to optimality are included

designed experiments with the parameters given in Table 1. For each combination of parameters, 5 instances are generated, resulting in a total of 100 problems for each value of n. Whenever possible, the problems are solved optimally using the time-indexed formulation TIP1. For the branch and bound (B&B) algorithm, `CPLEX` is called with the default options except that the next node to be explored is the one with the best estimated integer objective value instead of the node with the best objective value for the associated LP relaxation. Thus, we emphasize finding a good feasible solution for P1 as soon as possible, so as to provide a benchmark in case the B&B algorithm does not terminate with an optimal solution within a maximum CPU time of 30 minutes.

Table 2 shows how the performance of our lower bound and heuristics change, both in terms of solution quality and time, as the number of jobs increases. In columns 2–6, we report performance measures related to CPU times. The computation time for obtaining the optimal solution of TIP1 and the number of times the B&B algorithm terminated due to the time limit are reported in columns B&B and TL ("Time Limit"), respectively. The computing time to solve the relaxation at the root node for TIP1 and the total time required to compute the best solution from four different heuristic sequences are given in columns LP ("LP Relaxation") and BH ("Best Heuristic"), respectively. The ratio of these two quantities is computed in column LP/BH. The number of times the best heuristic solution and the LP relaxation of TIP1 matched the optimal solution are indicated under # Opt ("Number Optimal") in columns 7 and 8, respectively. The gaps for the bounds and heuristics are reported in columns 9–15. The columns LP and TR ("Transportation Problem") indicate the tightness of the two lower bounds obtained from the LP relaxation of TIP1 and the transportation problem, respectively. The remaining columns report the optimality gaps of our heuristics, where the percentage gap for any heuristic H is computed as $\frac{TC(H)-TC(\text{opt sol'n})}{TC(\text{opt sol'n})}$, when the optimal solution is available. Otherwise, we use $TC(S^*_{\text{LP(TIP1)}})$ instead of the optimal objective value. In columns TR and BH, the numbers in parentheses refer to the percentage gaps when the cost coefficients of Sourd and Kedad-Sidhoum (2003), given in (2.16), are used instead of our cost coefficients, given in (2.11). For each value of n, the first row indicates the average and the second row indicates the worst case performance.

The results indicate that the performance of our lower bound and heuristics improves as the number of jobs increases. One intuitive explanation is that, as the number of jobs increases, more jobs conflict, and there are fewer jobs that finish soon after their due dates, so that the performance of the lower bound improves. Recall that, when job j is scheduled non-preemptively and it finishes in the time interval $[d_j + 1, d_j + p_j - 1]$, then its contribution to the objective function of TR is strictly less than to that of P1. As the gap between the lower bound and the optimal solution diminishes, the sequences extracted from the lower bound improve. Generally, the switch heuristic sequence is the best, followed by the median completion time, average completion time and the last completion time sequences. Overall, in these 400 problems BH has very good average and worst case performances of 2.31% and 12.76%, respectively. Observe that the solution time of the LP relaxation of TIP1 grows very rapidly. On the other hand, TR can be solved much more quickly and its solution time is less sensitive to increases in the number of jobs. Moreover, it provides bounds that are nearly as good as those from the LP relaxation of TIP1. We also note that, on average, the lower bound is not as tight and the quality of the feasible solutions for the original problem declines (see column BH in Table 2), when the objective coefficients (2.16) of Sourd and Kedad-Sidhoum (2003) are used in the transportation problem. In these cases, we observe that in the optimal solution of TR, unit jobs are scheduled in periods k such that $c'_{jk} < c_{jk}$ (see Sect. 2).

In Table 3, we explore the effect of the tardiness (*TF*) and due date range (*RDD*) factors on the percentage gaps of

Table 3 Effect of the tardiness and due date range factors: average and worst case gaps of heuristics and bounds for $n = 80$

TF	Percentage gap (%) for TR[†]				Percentage gap(%) for BH			
	RDD				*RDD*			
	0.4	0.7	1.0	1.3	0.4	0.7	1.0	1.3
0.2	**−2.33(−3.78)**	**−2.97(−3.84)**	**−3.75(−3.72)**	**−2.51 − (2.50)**	**2.18(3.94)**	**4.14(5.16)**	**4.99(4.72)**	**2.51(2.78)**
	−3.30(−4.60)	−4.01(−5.07)	−6.02(−5.68)	−3.27(−3.36)	2.91(5.85)	5.28(8.62)	7.18(7.64)	3.83(3.63)
0.4	**−1.53(−2.25)**	**−1.80(−2.33)**	**−2.02(−2.34)**	**−1.68(−1.78)**	**2.67(2.78)**	**2.51(3.50)**	**3.28(3.60)**	**1.25(2.58)**
	−1.98(−2.69)	−2.02(−2.69)	−2.40(−2.69)	−2.16(−2.14)	5.27(3.64)	4.14(5.81)	5.19(4.98)	1.63(3.91)
0.5	**−1.18(−1.79)**	**−1.27(−1.63)**	**−1.15(−1.47)**	**−1.62(−1.82)**	**2.28(2.62)**	**2.42(2.74)**	**1.66(1.88)**	**1.79(2.38)**
	−1.27(−1.08)	−1.28(−1.77)	−1.38(−1.86)	−2.21(−2.39)	4.57(3.88)	3.81(4.51)	2.36(2.34)	2.85(3.75)
0.6	**−1.29(−1.74)**	**−0.97(−1.26)**	**−1.36(−1.61)**	**−1.16(−1.45)**	**1.56(2.40)**	**1.45(1.52)**	**2.09(2.25)**	**1.52(2.26)**
	−1.85(−2.44)	−1.10(−1.41)	−1.77(−2.02)	−1.84(−1.86)	2.14(2.95)	2.32(2.52)	2.93(2.77)	2.24(3.09)
0.8	**−0.92(−1.17)**	**−0.98(−1.18)**	**−0.98(−1.33)**	**−1.21(−1.47)**	**0.86(1.17)**	**0.99(1.56)**	**1.54(2.21)**	**1.34(1.43)**
	−1.32(−1.57)	−1.10(−1.38)	−1.11(−1.55)	−1.61(−1.89)	2.06(2.65)	1.88(2.25)	2.19(3.73)	2.00(2.05)

[†]Only instances solved to optimality are included

Table 4 Effect of processing times: problem parameters

n	p_j	r_j	*TF*	*RDD*	ϵ_j, π_j
{20, 40, 60, 80}	$U[1,10]$, $U[1,30]$, $U[1,50]$	$U[0,P]$	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	$U[0,100]$
{100, 130, 170, 200}	$U[1,50]$, $U[1,75]$, $U[1,100]$	$U[0,P]$	{0.2, 0.4, 0.5, 0.6, 0.8}	{0.4, 0.7, 1.0, 1.3}	$U[0,100]$

TR and BH for problem instances with $n = 80$ in Table 2. Results for 5 problem instances are reported for each combination of *TF* and *RDD*. The performance measures that appear in parentheses are based on the cost coefficients of Sourd and Kedad-Sidhoum (2003).

Table 3 clearly indicates that for a given value of *RDD*, the percentage gaps for both TR and BH decrease as *TF* increases, which supports our intuition above that our algorithms perform better when the contention among jobs increases. Furthermore, we observe that our algorithms provide solutions of higher quality when our cost coefficients in (2.11) are used in the transportation problem, regardless of the tardiness and due date range factors. The cost coefficients of Sourd and Kedad-Sidhoum (2003) in (2.16) yield slightly better solutions on average only if $TF = 0.2$ and $RDD = 1.0$ or 1.3 in Table 3. We also note that based on the results in Table 3 it is not possible to make a general statement about the effect of increasing the due date range factor for a given tardiness factor.

5.3.2 *Processing times*

We also investigate whether our lower bound and heuristics are sensitive to increases in processing times, which may lead to increases in the number of preemptions. The problem parameters are given in Table 4. For each combination of parameters, 5 problem instances are generated, resulting in a total of 300 problems for each n. The problems with $n = 20, 40, 60, 80$ and $p_j \sim U[1,10]$ are the same problems as in the previous section. In this section we compare the feasible solutions from our algorithms to either the LP relaxation of TIP1 or to our lower bound in order to reduce the total computation time.

The results for $n \leq 80$ appear in Table 5. Here we report the difference between the objective values from our heuristics and the lower bound provided by the optimal objective of LP(TIP1), which yields an *upper* bound on the optimality gap. Nevertheless, the results in Table 5 are excellent. For these 1200 problems, the average gap of the best heuristic with respect to the LP relaxation of TIP1 is only 3.25%, and except for $n = 20$ and $p_j \sim U[1,50]$, the worst case is within 16% of the LP lower bound. Note that the column TR indicates the gap between the optimal solutions of TR and the LP relaxation. In general, there is a very slight degradation in performance as the processing times increase, although this effect is less pronounced than the effect of an increase in the number of jobs. The combined effect of increasing both the number of jobs and the processing times is a decrease in the percentage gaps.

In Sects. 1 and 2 we mentioned that obtaining a lower bound for P1 from TR can be accomplished in pseudo-polynomial time. From Table 5 we observe that for $n = 80$ the increase in average CPU times is about an order of magnitude when the processing time distribution changes from

Table 5 Effect of processing times: CPU times, contiguity indices, average and worst case gaps of heuristics and bounds

n	p_j	CPU time (sec.)			CI	Percentage gap (%)					
		LP	BH	LP/BH		TR	LCT	ACT	MCT	SW	BH
20	$U[1,10]$	**0.09**	**0.03**	**2.53**	**1.03(1.56)**	**−4.27(−4.68)**	**13.25**	**9.61**	**6.14**	**5.44**	**3.35(4.08)**
		0.21	0.05	5.25	3.84(4.26)	−13.95(−12.11)	65.85	40.43	25.77	18.59	12.94(15.35)
	$U[1,30]$	**1.10**	**0.09**	**12.13**	**1.09(1.64)**	**−4.52(−5.13)**	**16.48**	**10.99**	**5.73**	**7.02**	**3.99(4.79)**
		2.89	0.12	26.00	6.64(4.10)	−15.39(−13.73)	153.31	50.04	22.85	37.65	15.87(26.02)
	$U[1,50]$	**3.03**	**0.14**	**20.65**	**1.05(1.65)**	**−4.79(−5.36)**	**16.36**	**12.22**	**6.15**	**8.15**	**4.24(4.58)**
		8.88	0.23	43.88	2.93(3.01)	−20.90(−19.22)	56.42	73.81	35.52	47.78	35.52(23.38)
40	$U[1,10]$	**0.89**	**0.10**	**8.42**	**1.31(2.62)**	**−2.19(−2.73)**	**9.65**	**8.40**	**4.93**	**4.20**	**3.08(3.74)**
		2.48	0.15	22.55	5.32(4.30)	−6.70(−6.89)	33.21	29.43	20.65	15.97	12.06(12.67)
	$U[1,30]$	**10.27**	**0.31**	**32.22**	**1.30(2.07)**	**−2.37(−2.94)**	**11.11**	**9.87**	**4.84**	**4.97**	**3.47(4.38)**
		28.05	0.46	87.66	6.02(7.00)	−6.55(−6.55)	49.57	61.00	14.72	15.64	14.72(17.25)
	$U[1,50]$	**30.15**	**0.68**	**43.21**	**1.39(2.52)**	**−2.40(−2.95)**	**12.10**	**9.21**	**4.96**	**4.87**	**3.46(4.27)**
		86.10	1.04	103.73	11.92(4.32)	−6.54(−6.77)	80.88	38.84	26.20	22.84	15.52(14.21)
60	$U[1,10]$	**3.54**	**0.18**	**19.82**	**1.57(2.25)**	**−1.52(−1.98)**	**8.43**	**7.85**	**4.36**	**3.51**	**2.85(3.36)**
		10.09	0.24	48.11	6.48(5.33)	−5.15(−5.06)	39.07	35.17	20.40	17.95	10.21(8.91)
	$U[1,30]$	**46.45**	**0.73**	**63.82**	**1.51(2.31)**	**−1.71(−2.17)**	**9.43**	**8.62**	**5.00**	**3.77**	**3.32(4.16)**
		124.98	1.11	159.13	7.66(7.34)	−5.73(−6.25)	31.22	51.34	17.95	11.23	10.38(15.03)
	$U[1,50]$	**181.59**	**1.44**	**128.41**	**1.41(2.33)**	**−1.67(−2.15)**	**9.68**	**8.52**	**4.62**	**3.79**	**3.17(4.01)**
		749.08	2.40	416.16	3.72(4.53)	−4.46(−4.97)	33.80	32.97	18.79	19.45	12.47(11.82)
80	$U[1,10]$	**9.27**	**0.30**	**30.24**	**1.58(2.38)**	**−1.20(−1.59)**	**8.14**	**7.27**	**3.96**	**2.98**	**2.55(3.07)**
		23.91	0.39	69.11	4.15(5.27)	−4.44(−4.10)	49.36	34.82	17.62	9.05	8.68(9.45)
	$U[1,30]$	**165.56**	**1.19**	**140.56**	**1.57(2.59)**	**−1.30(−1.66)**	**8.79**	**7.13**	**4.15**	**2.98**	**2.65(3.16)**
		422.83	3.22	330.34	3.51(4.95)	−5.95(−5.48)	31.89	34.47	20.79	11.82	11.82(13.15)
	$U[1,50]$	**688.12**	**3.51**	**188.78**	**1.67(2.53)**	**−1.28(−1.69)**	**8.25**	**7.59**	**4.17**	**3.30**	**2.86(3.45)**
		2490.89	8.06	448.00	3.86(4.67)	−3.72(−3.80)	33.06	38.91	19.45	16.45	9.74(9.79)

$U[1,10]$ to $U[1,50]$. (Note that the increase is much more rapid for the LP relaxation of TIP1.) Therefore, it is important to assess the applicability of our heuristics to larger instances. In Table 6 results on problems with up to 200 jobs and processing times up to 100 units are reported. The percentage gaps of our heuristics are computed with respect to our lower bound because solving the LP relaxation of TIP1 for these instances is very time consuming. We observe that problems with $n = 100$ or 130, and $p_j \sim U[1,100]$ are solved within 1 and 2 minutes on average, respectively. Problems with $n = 170, 200$ and $p_j \sim U[1,100]$ take 6 and 11 minutes of CPU time on average, respectively. Note that doubling the average processing time, i.e., changing the processing time distribution from $U[1,50]$ to $U[1,100]$, results approximately in a five-fold increase in CPU times for all n. Fortunately, this increase in CPU times is accompanied by a decrease in average and maximum percentage gaps. For 300 instances with $n = 200$, BH is on average within 2.03% and in the worst case within 5.35% of our lower bound, respectively. In other words, the results in Table 6 support our earlier conclusion that increasing both the number of jobs and the processing times decreases the optimality gaps. These results demonstrate that our algorithms find very good solutions in reasonable computation times even for very large instances. In Sect. 6 we indicate a possible extension of our approaches that may be implemented to reduce computation times by further relaxing the problem.

Intuitively, we would expect the quality of the feasible solutions for P1 to improve as the optimal solution of TR becomes closer to a non-preemptive schedule. Now let $q_j = \frac{1}{p_j-1}\sum_{k=1}^{p_j-1} g_{j[k]}$ be the *contiguity index* of job j, where $g_{j[k]}$ is the number of time periods elapsed between the processing of unit jobs k and $k+1$ of job j in the optimal transportation solution S^*_{TR}. This is, therefore, a measure of spread for job j in S^*_{TR}, and we note that $q_j = 0$ if the unit jobs of job j are processed in p_j adjacent intervals. Then we define the contiguity index of S^*_{TR} as $q = \frac{1}{n}\sum_{j=1}^{n} q_j$. The average (upper row) and worst case (lower row) values are reported in column CI ("Contiguity Index") in Tables 5 and 6. The average contiguity index q increases as n increases, but, somewhat surprisingly, not necessarily as the processing times increases. The cost coefficients of Sourd and Kedad-Sidhoum (2003) in (2.16) yield higher q values in general (shown in parentheses in Tables 5–6), i.e., in this

Table 6 Large instances: CPU times, contiguity indices, average and worst case gaps of heuristics

n	p_j	CPU time (sec.)	CI	Percentage gap (%) LCT	ACT	MCT	SW	BH
100	$U[1,50]$	**10.66**	**1.70(2.62)**	**8.89**	**7.58**	**4.30**	**3.66**	**3.38(4.32)**
		21.25	5.49(4.24)	30.04	24.41	15.23	8.05	8.05(10.92)
	$U[1,75]$	**29.00**	**1.88(2.81)**	**9.01**	**7.25**	**4.45**	**3.66**	**3.33(4.45)**
		51.19	15.40(5.83)	28.78	24.41	19.44	9.27	9.26(11.36)
	$U[1,100]$	**60.87**	**1.80(2.84)**	**8.87**	**7.90**	**4.70**	**3.86**	**3.56(4.53)**
		117.80	4.57(5.38)	31.60	53.91	21.39	11.78	11.78(13.77)
130	$U[1,50]$	**26.34**	**1.87(2.99)**	**7.32**	**5.79**	**3.86**	**2.92**	**2.69(3.52)**
		44.93	3.87(7.81)	20.81	14.29	13.90	8.68	8.45(8.07)
	$U[1,75]$	**64.59**	**2.04(3.20)**	**8.16**	**6.85**	**3.90**	**3.18**	**2.95(3.81)**
		103.71	5.89(10.31)	27.68	24.90	12.92	9.82	9.70(10.27)
	$U[1,100]$	**126.44**	**2.34(3.39)**	**7.98**	**6.39**	**3.86**	**3.02**	**2.89(3.80)**
		250.78	20.51(11.89)	27.82	20.27	10.55	8.01	7.16(9.09)
170	$U[1,50]$	**69.32**	**2.37(3.43)**	**6.24**	**5.60**	**3.24**	**2.46**	**2.34(3.17)**
		125.18	16.68(5.41)	18.60	20.06	9.08	5.67	5.67(8.63)
	$U[1,75]$	**176.53**	**2.21(3.52)**	**6.31**	**5.57**	**3.07**	**2.43**	**2.30(2.99)**
		342.66	7.09(8.31)	20.71	18.45	7.51	7.75	6.73(7.60)
	$U[1,100]$	**340.31**	**2.29(3.47)**	**6.78**	**5.78**	**3.47**	**2.49**	**2.41(3.17)**
		577.63	11.67(9.44)	31.83	25.44	22.91	8.27	8.27(8.82)
200	$U[1,50]$	**123.25**	**2.48(3.84)**	**5.83**	**4.90**	**2.93**	**2.10**	**2.06(2.76)**
		218.25	5.44(8.39)	19.71	13.85	10.58	5.61	5.32(6.48)
	$U[1,75]$	**324.23**	**2.51(3.78)**	**5.51**	**4.96**	**3.03**	**2.12**	**2.05(2.76)**
		659.01	9.68(6.95)	18.49	17.07	11.25	5.35	5.35(6.33)
	$U[1,100]$	**643.77**	**2.99(4.06)**	**5.68**	**4.97**	**2.93**	**2.05**	**1.99(2.69)**
		1258.54	19.07(16.70)	17.33	17.28	9.30	4.91	4.91(7.71)

case the unit jobs that belong to the same job are grouped together less tightly in S^*_{TR}, which may affect the quality of the resulting feasible solutions for P1 adversely. Our cost coefficients may produce high q values when there exists a job j with $\epsilon_j > 0$ and $\pi_j = 0$ that has some unit jobs scheduled immediately before its due date in periods k such that $c_{jk} < 0$ and then is delayed for many time periods because $\pi_j = 0$. In such instances the cost coefficients in (2.16) may yield a lower q value by delaying all unit jobs of job j because $c'_{jk} = 0$ for all time periods $k \geq d_j - p_j + 1$.

In addition, the cost coefficients in (2.16), which are constant for p_j consecutive time periods, tend to produce alternate optima for TR that may yield different feasible solutions for P1 under the same sequencing rule. Consider the example in Fig. 1, where $p_1 = 5$, $p_2 = 3$, $d_1 = d_2 = 2$, $\epsilon_1 = \epsilon_2 = 1$, $\pi_1 = 3$, and $\pi_2 = 1$. The cost coefficients (for each job in each period) are indicated in the figure for reference. Both of the schedules in Fig. 1 are optimal for TR with an objective of 13. However, for instance, the MCT (median completion time) sequences extracted from these alternate optima on the left and on the right are $2 \to 1$ and $1 \to 2$, respectively. (The optimal sequence for P1 is $1 \to 2$.) We note that our cost coefficients in (2.11) provide an incentive to schedule the unit jobs of the same job as tightly together as possible and would tend to avoid ties such as in Fig. 1. Thus, for similar lower bound values we expect (2.11) to yield better sequence information for constructing feasible solutions for P1. In this example, the unique optimal solution for TR with the cost coefficients in (2.11) is to schedule $1 \to 2$ non-preemptively starting at time zero with an optimal objective of 12.6. This schedule is non-optimal with an objective of 15 if the cost coefficients in (2.16) are used for TR.

5.3.3 Cost structure

In this section we explore the sensitivity of our algorithms to the relative costs of earliness and tardiness. We use the data sets from Mazzini and Armentano (2001), who present both optimal and heuristic solutions for these problems. The values of the parameters are given in Table 7. Five problem instances for each combination of parameters result in a total of 180 problems for each n.

For $n \leq 20$, we compute the performance measures with respect to the optimal solution when it is available, or with

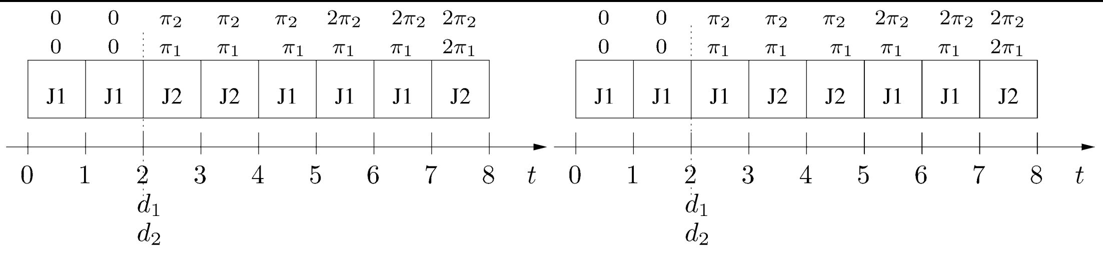

Fig. 1 Alternate optima with cost coefficients given in (2.16)

Table 7 Parameters for the data sets from Mazzini and Armentano (2001)

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

respect to $TC(S^*_{\text{LP(TIP1)}})$ when we are not able to find the optimal solution within the CPU time limit of 30 minutes. We do not use the solutions of Mazzini and Armentano (2001), as in several cases we discovered that they report erroneous optimal solutions. Hence, we also re-evaluate the performance of the heuristics proposed by Mazzini and Armentano (2001) based on the new best solutions obtained. For $n > 20$, it becomes very time consuming to even solve the LP relaxation of TIP1. Therefore, for these problems we report all performance measures with respect to our lower bound that yields an upper bound on the percentage gaps.

In Tables 8 and 9, for each combination of the number of jobs and the cost parameters, the first row indicates the average and the second row indicates the worst case performance. From Section 3, recall that one can create single-job instances with $\epsilon > \pi$ for which the gap between the optimal solution for P1 and the lower bound is large. However, Tables 8 and 9 indicate that the performance of our algorithms is not sensitive to the cost structure. In particular, for cases in which the earliness costs are twice as large as the tardiness costs, they perform as well as in the other cases.

In Tables 8 and 9, the last completion time sequence is the worst in most instances, and for small n the median completion time sequence is superior to all others. As n increases, the median completion time and switch heuristic sequences dominate the other two. Particularly for problems with more than 20 jobs in Table 9, both the average and maximum percentage gaps decrease as the number of jobs increases. The solution times increase rapidly with the number of jobs.

Overall, our algorithms perform very well. For $n = 8$, 10, 12, 20, we find the optimal solution for 135, 74, 47, 22 out of 180 problems in each case, respectively. For these 720 problems, the average and worst case optimality gaps of BH are 1.51% and 29.50%, respectively. For $n = 40$, 60, 80, we give upper bounds on the performance measures for a total of 540 problems, and the average and worst case gaps of BH are 1.77% and 10.53%, respectively.

Finally, Table 10 presents a comparison of the results of Mazzini and Armentano (2001) to our results. For each n, the first row indicates the average and the second row indicates the worst case performance across 180 problems. The label MA denotes the heuristic presented in Mazzini and Armentano (2001). In columns 4 and 5 ("MA" and "BH"), the heuristics are compared to either the optimal objective value of the original problem or the optimal objective value of the LP relaxation of TIP1 for $n \le 20$, and to our lower bound for $n > 20$, as discussed at the beginning of this section. In all cases our average gap is less than half of that of MA, and the worst cases indicate that our algorithms are more robust. In the last two columns, we report the percentage gap $(TC(MA) - TC(BH))/TC(BH)$ when $TC(BH) < TC(MA)$ and $TC(MA) < TC(BH)$, respectively. For $n = 40, 60, 80$, we observe that even when MA yields a smaller objective value than BH, the difference is small both on average and in the worst case. Mazzini and Armentano (2001) report that their heuristics take less than 0.5 seconds for all of their problem instances. Our heuristics are more time consuming; however, the additional effort is well justified. For $n \ge 40$, we provide better results for two-thirds of the problem instances, and the improvement can be significant.

6 Concluding remarks

We developed algorithms to solve a difficult single machine E/T scheduling problem in its most general form. This research was motivated by the appearance of this problem as a subproblem in more complex scheduling environments. The algorithms in this paper were successfully implemented for

Table 8 Effect of the cost structure: CPU times, number of optimal solutions, average and worst case gaps of heuristics and bounds

	Costs		CPU time	# Opt.[†]	Percentage gap (%)					
n	ϵ_j	π_j	(sec.)		TR[†]	LCT	ACT	MCT	SW	BH
8	$U[0, 50]$	$2\epsilon_j$	**0.06**	**33**	**−2.44**	**7.22**	**1.90**	**0.95**	**3.05**	**0.15**
			0.16		−7.08	40.22	24.69	7.94	43.48	3.87
	$U[0, 100]$	ϵ_j	**0.07**	**37**	**−3.82**	**7.21**	**3.30**	**0.89**	**3.62**	**0.15**
			0.14		−20.01	35.51	21.87	10.41	27.65	3.04
	$U[0, 100]$	$0.5\epsilon_j$	**0.06**	**34**	**−5.01**	**9.52**	**3.44**	**1.83**	**2.62**	**0.39**
			0.12		−48.51	136.36	44.17	25.86	24.60	5.23
	$U[0, 100]$	100	**0.07**	**31**	**−4.20**	**9.53**	**3.32**	**2.22**	**4.58**	**0.71**
			0.14		−30.25	50.27	29.59	18.96	31.65	12.52
10	$U[0, 50]$	$2\epsilon_j$	**0.11**	**17**	**−7.40**	**16.40**	**10.30**	**5.88**	**14.65**	**1.90**
			0.19		−21.85	77.14	80.87	80.87	115.13	15.09
	$U[0, 100]$	ϵ_j	**0.11**	**20**	**−8.27**	**18.39**	**8.81**	**4.56**	**9.62**	**2.00**
			0.21		−32.01	87.02	31.42	20.40	58.90	13.92
	$U[0, 100]$	$0.5\epsilon_j$	**0.10**	**19**	**−8.31**	**13.57**	**8.26**	**6.34**	**8.95**	**1.99**
			0.15		−27.96	50.56	75.32	75.32	61.48	9.45
	$U[0, 100]$	100	**0.11**	**18**	**−8.33**	**20.10**	**15.43**	**9.95**	**8.66**	**2.97**
			0.22		−31.28	95.59	187.62	149.17	54.46	29.50
12	$U[0, 50]$	$2\epsilon_j$	**0.13**	**12**	**−7.80**	**19.00**	**11.63**	**5.72**	**7.03**	**1.86**
			0.22		−31.16	98.81	80.69	39.95	22.67	13.47
	$U[0, 100]$	ϵ_j	**0.15**	**12**	**−7.79**	**16.71**	**11.41**	**7.25**	**11.27**	**2.80**
			0.26		−33.51	68.29	76.26	62.94	98.53	14.74
	$U[0, 100]$	$0.5\epsilon_j$	**0.14**	**14**	**−7.36**	**19.61**	**8.29**	**4.04**	**6.34**	**2.06**
			0.23		−35.67	105.39	43.12	25.06	28.52	15.64
	$U[0, 100]$	100	**0.15**	**8**	**−6.83**	**16.76**	**9.39**	**7.20**	**7.73**	**3.00**
			0.20		−20.09	91.09	41.44	88.99	44.57	19.02
20	$U[0, 50]$	$2\epsilon_j$	**0.29**	**8**	**−1.68**	**5.07**	**3.73**	**2.07**	**2.44**	**0.82**
			0.46		−5.63	21.36	30.46	9.53	21.36	6.38
	$U[0, 100]$	ϵ_j	**0.32**	**5**	**−1.64**	**4.56**	**2.85**	**2.15**	**2.02**	**1.04**
			0.53		−5.37	16.37	20.37	8.98	6.34	4.47
	$U[0, 100]$	$0.5\epsilon_j$	**0.30**	**7**	**−2.54**	**6.10**	**3.51**	**1.60**	**2.34**	**1.07**
			0.55		−9.78	24.26	19.87	10.00	12.98	10.00
	$U[0, 100]$	100	**0.33**	**2**	**−2.11**	**6.16**	**4.11**	**2.10**	**2.93**	**1.18**
			0.60		−6.24	31.66	18.48	10.21	12.73	5.50

[†]For $n = 12$ (20), 178 (171) instances out of 180 solved to optimality are included

solving the pricing subproblems of a column generation algorithm for an m-machine flow shop E/T scheduling problem, which demonstrates both their effectiveness and speed (see Bülbül et al. 2004). We investigated a relatively unexplored path by considering a preemptive structure that differs from the preemption implicit in the LP relaxation of the integer programming formulation of the original non-preemptive problem. We constructed a set of coefficients for the preemptive problem that has two attractive features. First, it is, in some sense, the best set of coefficients with a piecewise linear form with two segments. Second, the coefficients provide better discrimination among similar alternatives than the coefficients proposed by other researchers for similar preemptive relaxations. Thus, our procedure is less likely to provide alternate optima, which in turn leads to better heuristic solutions that are based upon information from the preemptive schedule. Our computational experiments demonstrated that this approach yields excellent results.

We also note that a possible extension of our research could consider "composite unit jobs" in order to reduce solution times for problems with long processing times. The idea is to divide each job into segments whose duration is greater than one time unit, and to treat these segments

Table 9 Effect of the cost structure: CPU times, average and worst case gaps of heuristics

n	Costs		CPU time	Percentage gap (%)				
	ϵ_j	π_j	(sec.)	LCT	ACT	MCT	SW	BH
40	$U[0,50]$	$2\epsilon_j$	**1.61**	**6.35**	**4.66**	**2.71**	**2.85**	**2.03**
			2.44	23.89	43.07	9.36	11.40	9.36
	$U[0,100]$	ϵ_j	**1.57**	**4.63**	**3.46**	**2.10**	**2.29**	**1.71**
			2.48	15.17	11.81	7.53	7.78	3.76
	$U[0,100]$	$0.5\epsilon_j$	**1.64**	**6.18**	**4.81**	**2.92**	**3.13**	**2.49**
			3.66	27.87	20.58	14.18	10.53	10.53
	$U[0,100]$	100	**1.90**	**6.03**	**4.67**	**3.12**	**3.03**	**2.42**
			3.19	19.48	16.35	11.91	13.12	6.55
60	$U[0,50]$	$2\epsilon_j$	**4.92**	**3.98**	**2.89**	**1.90**	**1.94**	**1.58**
			8.70	16.60	9.60	5.81	5.02	4.20
	$U[0,100]$	ϵ_j	**4.68**	**4.59**	**2.97**	**2.08**	**1.97**	**1.57**
			11.44	16.78	11.39	9.13	6.11	4.46
	$U[0,100]$	$0.5\epsilon_j$	**4.90**	**4.38**	**3.18**	**2.01**	**2.13**	**1.65**
			14.01	13.01	17.11	7.31	10.31	6.09
	$U[0,100]$	100	**5.56**	**4.27**	**3.37**	**2.21**	**2.15**	**1.82**
			10.55	15.11	9.07	6.65	8.59	6.65
80	$U[0,50]$	$2\epsilon_j$	**15.72**	**3.21**	**2.94**	**1.57**	**1.48**	**1.27**
			30.00	12.21	13.33	6.97	5.91	4.63
	$U[0,100]$	ϵ_j	**15.97**	**3.30**	**3.09**	**1.95**	**1.68**	**1.45**
			30.54	13.79	13.00	6.10	6.44	4.74
	$U[0,100]$	$0.5\epsilon_j$	**15.89**	**3.87**	**3.20**	**1.91**	**1.85**	**1.62**
			38.03	14.90	12.45	8.12	5.65	5.42
	$U[0,100]$	100	**21.61**	**4.38**	**3.38**	**2.28**	**1.77**	**1.67**
			57.63	11.03	15.06	8.91	4.27	4.27

Table 10 Comparison with Mazzini and Armentano (2001)

n	# Opt.		Gap (%)[†]		Number of times			Gap (%)	
	MA	BH	MA	BH	BH < MA	MA < BH	BH = MA	BH < MA	MA < BH
8	**127**	**135**	**0.89**	**0.35**	**44**	**28**	**108**	**3.06**	**−1.29**
			30.03	12.52				25.19	−9.11
10	**64**	**74**	**5.47**	**2.22**	**82**	**60**	**38**	**9.67**	**−3.48**
			85.12	29.50				85.12	−18.50
12	**40**	**46**	**7.08**	**2.43**	**104**	**58**	**18**	**9.31**	**−2.49**
			59.04	19.02				52.50	−12.27
20	**40**	**22**	**2.66**	**1.03**	**80**	**90**	**10**	**4.48**	**−0.77**
			36.85	10.00				30.23	−7.27
40	**N/A**[*]	**N/A**[*]	**4.35**	**2.16**	**116**	**64**	**0**	**3.52**	**−0.43**
			38.52	10.53				26.67	−2.25
60	**N/A**[*]	**N/A**[*]	**3.79**	**1.65**	**115**	**65**	**0**	**3.41**	**−0.26**
			23.64	6.65				21.43	−1.71
80	**N/A**[*]	**N/A**[*]	**3.50**	**1.50**	**125**	**55**	**0**	**2.91**	**−0.25**
			44.30	5.42				39.30	−1.17

[†]These percentage gaps are computed differently for $n \leq 20$ and $n > 20$. See text for details

[*]Optimal solution not available

in the same way as unit jobs are treated in our methodology. This type of approximation can be implemented in many different ways, but by experimenting with different values of the basic segment duration, one could trade off factors such as ease of constructing feasible solutions from the solution of TR, solution quality and computing time.

Our model is flexible and can be used to solve a variety of problems by making appropriate adjustments to the problem parameters. For instance, the weighted tardiness problem can be solved by setting all earliness costs equal to zero. In addition, raw material inventory holding costs can easily be incorporated for jobs that are ready, but waiting for processing by setting $\epsilon'_j = \epsilon_j - h_j$ and $\pi'_j = \pi_j + h_j$ for all jobs, where h_j is the raw material inventory holding cost of job j, and ϵ'_j and π'_j are the new earliness and tardiness costs, respectively. We have already exploited this property to solve a flow shop scheduling problem with intermediate inventory holding costs (Bülbül et al. 2004), and hope to extend this research to job shop scheduling with inventory holding costs in the future.

Acknowledgements We would like to thank Professor Vinicius Armentano and Dr. Renata Mazzini for providing the data in Sect. 5.3.3. This research was partially supported by NSF Grant DMI-0092854.

References

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

Bülbül, K. (2002). *Just-in-time scheduling with inventory holding costs*. PhD thesis, University of California at Berkeley.

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

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

Dyer, M., & 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.

Fry, T., & Keong Leong, G. (1987). A bi-criterion approach to minimizing inventory costs on a single machine when early shipments are forbidden. *Computers and Operations Research*, *14*(5), 363–368.

Fry, T., Armstrong, R., & Blackstone, J. (1987). Minimizing weighted absolute deviation in single machine scheduling. *IIE Transactions*, *19*(4), 445–450.

Fry, T., Armstrong, R., & Rosen, L. (1990). Single machine scheduling to minimize mean absolute lateness: a heuristic solution. *Computers and Operations Research*, *17*(1), 105–112.

Fry, T., Armstrong, R., Darby-Dowman, K., & Philipoom, P. (1996). A branch and bound procedure to minimize mean absolute lateness on a single processor. *Computers and Operations Research*, *23*(2), 171–182.

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

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

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

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

Hoogeveen, J., & van de Velde, S. (1996). A branch-and-bound algorithm for single-machine earliness-tardiness scheduling with idle time. *INFORMS Journal on Computing*, *8*(4), 402–412.

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

Kim, Y.-D., & Yano, C. A. (1994). Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. *Naval Research Logistics*, *41*(7), 913–933.

Lee, C., & Choi, J. (1995). A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. *Computers and Operations Research*, *22*(8), 857–869.

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

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

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

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

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

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

Phillips, C., Stein, C., & Wein, J. (1998). Minimizing average completion time in the presence of release dates. *Mathematical Programming*, *82*, 199–223.

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

Sourd, F. (2004). The continuous assignment problem and its application to preemptive and non-preemptive scheduling with irregular cost functions. *INFORMS Journal on Computing*, *16*(2), 198–208.

Sourd, F., & Kedad-Sidhoum, S. (2003). The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, *6*(6), 533–549.

Sridharan, V., & Zhou, Z. (1996). A decision theory based scheduling procedure for single-machine weighted earliness and tardiness problems. *European Journal of Operational Research*, *94*(2), 292–301.

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

Szwarc, W. (1993). Adjacent orderings in single-machine scheduling with earliness and tardiness penalties. *Naval Research Logistics*, *40*(2), 229–243.

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

Ventura, J., & Radhakrishnan, S. (2003). Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research*, *144*(3), 598–612.

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

Wan, G., & Yen, B. (2002). Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, *142*(2), 271–281.

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