J Sched (2015) 18:3–13
DOI 10.1007/s10951-014-0396-7

Optimal delivery time quotation in supply chains to minimize tardiness and delivery costs

Sorina Dumitrescu · George Steiner · Rui Zhang

Received: 19 June 2013 / Accepted: 5 September 2014 / Published online: 21 October 2014

Abstract There are many situations when, due to unexpected delays, the supplier may not be able to deliver some orders by the promised due dates. We present a model for quoting attainable delivery times to minimize tardiness penalties and delivery costs, when deliveries take place in batches. We show that the general problem is strongly $\mathcal{NP}$-hard, but when all orders have the same per-unit due-date-assignment cost, it is $\mathcal{NP}$-hard only in the ordinary sense. For the latter case, we present a pseudo-polynomial algorithm, which is converted into a fully polynomial-time approximation scheme. If the tardiness penalties are also identical, we show that the problem can be solved in polynomial time.

Keywords Scheduling · Tardiness · Delivery times · Approximation algorithm

1 Introduction

Supply chain management has been one of the most important topics in manufacturing research. Most of the supply chain literature focuses on issues on the strategic level, using stochastic models. According to the survey paper by Thomas and Griffin (1996), over 11 % of the U.S. Gross National Product is spent on logistics, and for many products, logistics expenses may exceed 30 % of the cost of goods sold. This underlines the need for research dealing with scheduling and distribution issues in supply chains. The recently emerging research area of supply chain scheduling studies such problems using deterministic models Hall and Potts (2003), Chen and Vairaktarakis (2005). In these models, a set of jobs (customer orders) must be processed in a facility, usually represented by a single "machine," and then delivered to the customer without any delay. The problem is to find an optimal schedule of production and distribution with a composite objective that measures the customer service level and the total distribution cost. Customer service level usually is expressed as a function of the times when the orders are delivered. The distribution costs depend on the configuration of the delivery system used. A classification and overview of these models are in Chen (2010).

One set of these models measures the customer service component of the schedule cost by considering how closely the schedule follows the contracted delivery dates (due dates). Customers often demand that suppliers meet contracted delivery dates or face large penalties. For example, Slotnick and Sobel (2005) cite contracts from the aerospace industry, which may impose tardiness penalties of millions of dollars on subcontractors for aircraft components. In order to avoid tardiness penalties, including the possibility of losing customers, companies are under increasing pressure to quote *attainable* delivery dates for customer orders. Naturally, longer due dates are easier to meet, but promising delivery dates too far into the future may not be acceptable to the customer, who may even be lured away by a competitor offering shorter delivery times. In such situations, a company may be forced to offer price discounts in order to retain the business. At the same time, shorter due dates increase the probability that the order will be delivered late. Thus there

S. Dumitrescu
Department of Electrical and Computer Engineering,
McMaster University, Hamilton, ON, Canada
e-mail: sorina@mail.ece.mcmaster.ca

G. Steiner (✉) · R. Zhang
Operations Management Area, DeGroote School of Business,
McMaster University, Hamilton, ON, Canada
e-mail: george.steiner4@gmail.com

R. Zhang
e-mail: zhangr6@mcmaster.ca

is an important trade-off between assigning relatively short due dates to customer orders and avoiding tardiness penalties, which creates the need for methodology that allows firms to quote attainable delivery dates and obtain efficient schedules at the same time.

In traditional scheduling models, due dates are considered as given by decisions exogenous to scheduling. Because of the strong pressure on managers of manufacturing or service organizations to quote attainable delivery dates to clients, a large number of recent scheduling studies include due-date assignment, i.e., treat the due dates as variables with a certain cost. Many of these due-date assignment models are classified in Gordon et al. (2002a, b). More recent surveys can be found in Kaminsky and Hochbaum (2004) and Gordon et al. (2004). All these models assign a certain cost to the assigned due dates. Typically, the longer is the due date, the higher is its assignment cost to reflect the trade-off discussed above.

The ability to integrate due-date assignment and scheduling can be an important factor in improving performance in supply chains. Among the earlier papers relevant to our current model, Shabtay and Steiner (2006) study a single-machine scheduling problem, in which each job has a previously contracted due date that can be changed to a new assigned due date at a certain cost. Their goal is to find a schedule which minimizes the sum of due-date-assignment costs and the tardy-job penalties with respect to the assigned due dates, but their model does not include distribution (batching or delivery) costs. Hall and Potts (2003) consider delivery costs together with the scheduling objective of minimizing the weighted number of tardy jobs (with fixed due dates) and assume that deliveries occur in batches and without any further delay. This problem has strong connections to the classical scheduling problem of minimizing the weighted number of tardy jobs on a single machine, which has been studied with varying assumptions since the 1960s. For details, we refer to the papers: Moore (1968), Karp (1972), Gens and Levner (1981), Hochbaum and Landy (1994), and Brucker and Kovalyov (1996). Hall and Potts (2003) show that their problem with batch deliveries is ordinary $\mathcal{NP}$-hard by presenting a pseudo-polynomial algorithm for it, but they make the simplifying assumption that the tardy jobs do not need to be delivered. Steiner and Zhang (2007) include batch-setup time and tardy deliveries in the model and present a pseudo-polynomial algorithm and a fully polynomial-time approximation scheme (FPTAS).

Our problem is essentially the combination of the above two problems studied by Steiner and Zhang (2007) and Shabtay and Steiner (2006). The former paper did not consider assignable due dates. The latter paper considered due date assignments but did not consider distribution costs and provided algorithms for two special cases only. Our current model allows arbitrary assigned due dates, and the objective is to minimize the sum of the weighted number of tardy jobs, the due-date-assignment costs, and the batch-delivery costs in a supply chain.

The paper is organized as follows. Section 2 contains preliminaries: a formal problem definition, important propositions, and a strong $\mathcal{NP}$-hardness proof. In Section 3, a special case with equal per-unit due-date-assignment costs is studied. It is proven to be $\mathcal{NP}$-hard, and a pseudo-polynomial algorithm and a multi-stage FPTAS are proposed. In Section 4, the special case with equal per-unit due-date-assignment costs and equal tardiness penalties is shown to be polynomially solvable. Section 5 includes our conclusions and a brief discussion of future research.

2 Problem definition, due-date assignments, and computational complexity

Given a job set $J = \{1, \ldots, n\}$, all the jobs have to be scheduled on a single machine and delivered to the customer in batches. For each job $j \in J$, *processing time* and *tardiness penalty* (*weight*) are denoted by p_j and w_j, respectively. Let s be the *batch-setup time* before processing the first job in each batch and q be the *batch-delivery cost* for each batch. In contrast with classical machine scheduling, delivering jobs in batches saves costs. However, an early finished job has to wait for delivery until the completion of the last job in the same batch. This time is called the *batch-completion time*, denoted by C_j for $j \in J$. This completion time is shared by all the jobs in the same batch. Our model analyzes the trade-offs between the savings in batch-delivery costs and the effect of the longer batch-completion times on the schedule quality.

Let A denote the common *lead time* normally used for all the jobs. Let D_j denote the *assigned due date* of job j, which is a decision made or negotiated by the supplier. Let $R_j = \max\{D_j - A, 0\}$ represent the *extended time units* from A to D_j and $\alpha_j R_j$ be the corresponding *due-date-assignment cost*, where α_j is the *due-date-assignment cost per extended time unit*. Since D_j becomes the effective due date of job j, it is used to determine the *tardiness indicator* U_j such that if $C_j > D_j$, $U_j = 1$; otherwise, $U_j = 0$. The goal is to find a schedule which minimizes the sum of the weighted number of tardy jobs, the due-date-assignment costs, and the batch-delivery costs, denoted by $1|s, A, \text{DIF}| \sum \alpha_j R_j + \sum w_j U_j + bq$, where DIF means that a different due date can be assigned to each job (Gordon et al. 2002b), and b is the number of batches. In addition to the non-preemption assumption, all jobs are available for processing at time zero, and we assume w.l.o.g. that all data are non-negative integers.

There are three decisions to be made in the $1|s, A, \text{DIF}$ $|\sum \alpha_j R_j + \sum w_j U_j + bq$ problem: determining a job sequence; grouping the sequence into batches; and assigning the due dates. Let σ be a given schedule, where the first two decisions have been made, but no due date has been assigned to any job yet. Let $b(\sigma)$ be the number of batches in σ and $D_j(\sigma)$ the due date to be assigned to each job $j \in J$ in σ. Thus, the batch-delivery cost of σ is $b(\sigma)q$. In order to minimize the total cost of σ, $D_j(\sigma)$ needs to be determined for each job such that $\sum \alpha_j R_j(\sigma) + \sum w_j U_j(\sigma)$ is minimized, where $\alpha_j R_j(\sigma) = \alpha_j \max\{D_j(\sigma) - A, 0\}$ is the due-date-assignment cost of $D_j(\sigma)$, and $w_j U_j(\sigma)$ is the tardiness penalty if job j is tardy with respect to $D_j(\sigma)$. The following important proposition is proved by Shabtay and Steiner (2006).

Proposition 2.1 *For any given schedule σ, the optimal due-date assignment is*

$$D_j(\sigma) = \begin{cases} A, & \text{if } C_j(\sigma) \le A \text{ or } C_j(\sigma) > A + \frac{w_j}{\alpha_j} \\ C_j(\sigma), & \text{if } A < C_j(\sigma) \le A + \frac{w_j}{\alpha_j} \end{cases}. \tag{1}$$

This proposition further implies the following.

Proposition 2.2 *For the $1|s, A, DIF|\sum \alpha_j R_j + \sum w_j U_j + bq$ problem, there is an optimal solution in which $A + \frac{w_j}{\alpha_j}$ is an upper bound for the assigned due date of each job $j \in J$. Furthermore, job j is tardy, with an assigned due date $D_j(\sigma) = A$, in an optimal schedule σ if and only if $C_j(\sigma) > A + \frac{w_j}{\alpha_j}$.*

The above propositions analyze the trade-off between due-date extension and tardiness costs. Regarding the computational complexity, we show that even a simplified special case of the problem is strongly $\mathcal{NP}$-hard: Consider an instance of the $1|s, A, \text{DIF}|\sum \alpha_j R_j + \sum w_j U_j + bq$ problem, where $s = 0$, $A = 0$ and $w_j >> \alpha_j \sum_{k=1}^{n} p_k, \forall j \in J$. By Eq. (1), $D_j(\sigma) = C_j(\sigma)$ and $U_j(\sigma) = 0$, for any σ and $\forall j \in J$. Then, the objective value can be written as

$$\sum \alpha_j R_j(\sigma) + \sum w_j U_j(\sigma) + bq = \sum \alpha_j C_j(\sigma) + bq,$$

and therefore, the above instance is equivalent to the problem of minimizing the sum of total weighted completion times and batch-delivery costs, denoted by $1||\sum \alpha_j C_j + bq$. Since this problem is strongly $\mathcal{NP}$-hard (Hall and Potts 2003), Corollary 2.1 follows directly.

Corollary 2.1 *The $1|s, A, DIF|\sum \alpha_j R_j + \sum w_j U_j + bq$ problem is strongly $\mathcal{NP}$-hard.*

3 The problem with equal per-unit due-date-assignment costs

The problem with equal per-unit due-date-assignment costs, i.e., $\alpha_j = \alpha > 0, \forall j \in J$, is denoted by $1|s, A, \text{DIF}|\sum \alpha R_j + \sum w_j U_j + bq$. Consider an instance where $s = 0$, $q = 0$ and $\alpha >> w_j, \forall j \in J$. Then it can be easily seen that this instance is equivalent to the single-machine scheduling problem of minimizing the weighted number of tardy jobs with a common due date A, denoted by $1|A|\sum w_j U_j$. The latter problem is equivalent to the knapsack problem, which is well known to be $\mathcal{NP}$-hard. Thus Corollary 3.1 follows.

Corollary 3.1 *The $1|s, A, DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem is $\mathcal{NP}$-hard.*

Since the single-machine scheduling problem of minimizing the weighted number of tardy jobs, denoted by $1||\sum w_j U_j$, is a classical $\mathcal{NP}$-hard problem, considerable literature exists on approximation algorithms for it. We mention here the well-known pseudo-polynomial algorithm and FPTAS of Lawler (1982) and an FPTAS by Brucker and Kovalyov (1996) for the batching version of the problem with no delivery costs or due-date assignment. Our problem is a further generalization, which includes batch-delivery costs and individual due-date assignment for the jobs. As we will see later, this makes it much harder to design efficient approximation algorithms for the problem. In the remainder of this section, a pseudo-polynomial algorithm and an FPTAS are proposed for this problem.

3.1 Pseudo-polynomial algorithm

Hall and Potts (2003) present a pseudo-polynomial algorithm for the supply chain scheduling problem $1||\sum w_j U_j + bq$ with batch deliveries, but they make the simplifying assumption that the tardy jobs do not need to be delivered. Steiner and Zhang (2007) include batch-setup time and tardy deliveries in the model and present a pseudo-polynomial algorithm and an FPTAS. We start with two important observations for our $1|s, A, \text{DIF}|\sum \alpha R_j + \sum w_j U_j + bq$ problem, which will be used in developing a dynamic-programming pseudo-polynomial algorithm, in which only the schedules with the observed structures will be searched.

Proposition 3.1 *There is an optimal schedule for the $1|s, A, DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem in which all the tardy jobs are delivered at the end in a single batch either by themselves or together with some early jobs.*

Proof Scheduling the tardy jobs at the end in the last batch always leaves more room to schedule the early jobs better. □

Proposition 3.2 *For the $1|s, A, DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem, there is an optimal schedule σ, in which the early jobs are processed in the shortest processing time* (SPT) *order.*

Proof Suppose that two jobs i and k with $p_i > p_k$ are early in two consecutive batches in σ with batch-completion times $C_i(\sigma) < C_k(\sigma)$. By Propositions 2.1 and 2.2, we can assume that $D_i(\sigma) \in \{A, C_i(\sigma)\}$ and $C_i(\sigma) \le \frac{w_i}{\alpha} + A$, and $D_k(\sigma) \in$

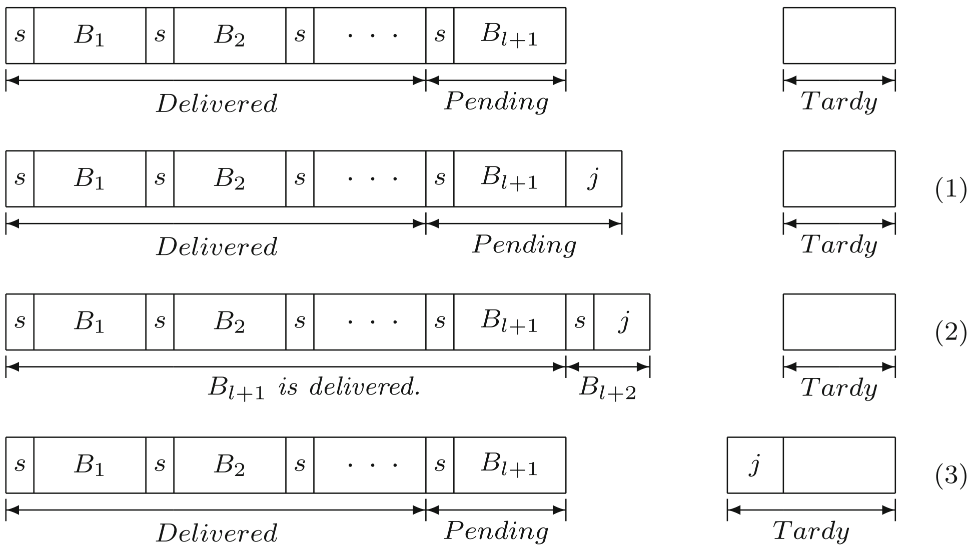

Fig. 1 Scheduling alternatives for adding j to the partial schedule $(j-1, l, k, t, x)$

$\{A, C_k(\sigma)\}$ and $C_k(\sigma) \le \frac{w_k}{\alpha} + A$. Let σ' be the schedule in which jobs i and k exchange their positions. Then $C_i(\sigma') = C_k(\sigma)$ and $C_k(\sigma') = C_i(\sigma) - p_i + p_k < C_i(\sigma)$.

It is clear that all the early jobs (excluding job i) will stay early after the exchange, and the number of jobs in each batch would stay the same after the exchange. If $C_i(\sigma') \le \frac{w_i}{\alpha} + A$, then job i could be scheduled early. In this situation, the total due-date-assignment cost will not increase because job i and job k have the same α. If $C_i(\sigma') > \frac{w_i}{\alpha} + A$, then it is best to set $D_i(\sigma') = A$ and let job i be tardy. In this situation, the cost will be reduced by at least

$$\begin{aligned}
&\alpha[D_i(\sigma) + D_k(\sigma)] - \alpha[D_i(\sigma') + D_k(\sigma')] - w_i \\
&\quad = \alpha[\max\{A, C_i(\sigma)\} - \max\{A, C_k(\sigma')\}] \\
&\qquad + [\alpha \max\{A, C_k(\sigma)\} - \alpha A - w_i] \\
&\quad \ge \alpha[\max\{A, C_i(\sigma)\} - \max\{A, C_k(\sigma')\}] \\
&\qquad + [\alpha \max\{A, C_k(\sigma)\} - \alpha C_i(\sigma')] \\
&\quad \ge 0.
\end{aligned}$$

After performing the exchange for all such pairs in consecutive batches, a desired SPT sequence will be obtained for all the early jobs. □

For the rest of this subsection, let all jobs be indexed so that $p_1 \le p_2 \le \cdots \le p_n$. A batch is called *delivered* (or finalized) if due dates have been assigned to the jobs in the batch. On the other hand, a batch is called *pending*, if no due date has been assigned yet to any job in the batch (since we do not know the batch-completion time before finalizing).

We present our DP algorithm (A1) using state-space generation: Let $(j-1, l, k, t, x)$ be the state for a partial schedule on job set $\{1, 2, \ldots, j-1\}$, where

- l is the number of delivered batches, which are denoted by $B_1, B_2, \ldots, B_l$;
- k is the number of jobs in the pending batch, denoted by B_{l+1};
- t is the total processing time of the early jobs in $\{B_1, B_2, \ldots, B_{l+1}\}$;
- x is the objective value for the jobs in the delivered batches plus the tardiness penalties for jobs scheduled tardy in the partial schedule. (Note that x does not include the delivery cost of B_{l+1} or the due-date-assignment costs of the jobs in B_{l+1}.)

We explain the salient features of our approach here and refer the reader to Algorithm A1 for their implementation: Consider the state $(j-1, l, k, t, x)$, for $j \le n$. Then the next unscheduled job j can be scheduled using one of the following three alternatives as shown in Fig. 1:

(1) Add job j to the pending batch B_{l+1} and keep it pending; (2) Close and deliver B_{l+1} and open a new pending batch (B_{l+2}) starting with job j in it; (3) Schedule job j in the last batch as a tardy job, denoted by the rectangle at the end. It is important to note that Alternative (2) can only be applied to a partial schedule with a *non-empty* pending batch, i.e., when $k > 0$. Thus, only a non-empty pending batch can be delivered. Additionally, notice that the pending batch can be empty only before scheduling the first early job. Finally, we make the observation that Alternative (1) is the only operation which can be used to schedule the first early job.

In Alternative (1), x stays the same, since no delivery or tardiness cost occurs. In Alternative (2), closing B_{l+1} results in batch-completion time (makespan) $t + (l+1)s$ for the batch. Therefore, a common due date $\max\{A, t + (l+1)s\}$ is assigned to the jobs in B_{l+1}, which makes the jobs early. The due-date-assignment cost $\alpha k \max\{t + (l+1)s - A, 0\}$ and delivery cost q are added to x. In Alternative (3), the tardiness penalty w_j is added to x. The delivery cost for the tardy jobs will be added only at the end when their batch is finalized.

Now let us consider how to finish computing the cost of a state (n, l, k, t, x) representing a "partial" schedule with n scheduled jobs. There are three cases: (1) there are no early jobs, i.e., $t = 0$; (2) there are no tardy jobs, i.e., $t = P$; and (3) the schedule with $0 < t < P$ has a non-empty pending batch and at least one tardy job (since $t < P$). In the following algorithm A1, procedure [FinalCompletion] computes the costs for these cases. In case (1), the delivery cost q of the tardy jobs is added to x. (Note that this case is identical to our initial base case defined below and thus it is not necessary to compute it in [FinalCompletion].) For case (2), a common due date $\max\{A, t + (l + 1)s\}$ is assigned to the jobs in the pending batch, which makes the jobs early. The due-date-assignment cost $\alpha k \max\{t + (l + 1)s - A, 0\}$ and the delivery cost q are added to x. For case (3) there are two options: (a) deliver separately the pending batch and the batch with the tardy jobs; (b) deliver the jobs from the pending batch together with the tardy jobs in a single batch. The option yielding the lower cost is selected. Note that for option (a) a common due date $\max\{A, t+(l+1)s\}$ is assigned to the jobs in the pending batch, which makes them early. The due-date-assignment cost $\alpha k \max\{t + (l + 1)s - A, 0\}$ and delivery cost $2q$ are added to x, where the second delivery cost q is for delivering the tardy jobs in a separate batch. In option (b) the jobs from the pending batch are made early by assigning them the due date $\max\{A, P + (l + 1)s\}$. The due-date-assignment cost $\alpha k \max\{P + (l + 1)s - A, 0\}$ and the delivery cost q are added to x.

In order to reduce the state space, for any two states (j, l, k, t, x_1) and (j, l, k, t, x_2) with $x_1 < x_2$, we can eliminate the second one, since any later states generated from it cannot lead to a smaller x value than the value of similar states generated from the first one.

Remark 3.1 For all states with the same entries: $(j, l, k, t, \cdot)$, we only need to keep the one with the smallest x value.

To initialize, we also consider the base case when all jobs are scheduled as tardy jobs and delivered in a single batch:

$$\bar{x} = \sum_{j=1}^{n} w_j + q. \tag{2}$$

Clearly, the cost of the optimal schedule cannot be larger than $\bar{x}$. (Note that by initializing x to $\bar{x}$ it is no longer needed to consider case (1) in [FinalCompletion] for any state $(n, l, k, t = 0, x)$ and thus this calculation is omitted.)

Let v^* be the cost of the optimal schedule with initial value $\bar{x}$, computed in Eq. (2), which represents the cost of scheduling all the jobs in a single tardy batch. Algorithm A1 starts from an empty schedule, (0, 0, 0, 0, 0). Then v^* will be repeatedly updated whenever a smaller value x is obtained when completing a partial schedule into a full schedule. States for partial schedules for the first j jobs $\{1, 2, \cdots, j\}$, $j = 1, 2, \ldots, n$, are included in set $\mathcal{S}^{(j)}$. Initially, $\mathcal{S}^{(0)} = \{(0, 0, 0, 0, 0)\}$.

Algorithm A1

```
[Initialization] Set $v^* = \bar{x}$, $\mathcal{S}^{(0)} = \{(0,0,0,0,0)\}$ and
  $\mathcal{S}^{(j)} = \emptyset$, for $j = 1, \cdots, n$.
[Generation] Generate set $\mathcal{S}^{(j)}$ from $\mathcal{S}^{(j-1)}$.
For $j = 1$ to $n + 1$
    [Setup] Set $\mathcal{T} = \emptyset$.
    For each state $(j - 1, l, k, t, x)$ in $\mathcal{S}^{(j-1)}$
      Alternative (1): If $j \le n$, then set $\mathcal{T} \leftarrow \mathcal{T} \cup (j, l, k+1, t + p_j, x)$.
        /*Schedule job j in the current pending batch or schedule job j as the first early job.
      Alternative (2): If $j \le n$ and $k > 0$, then set $\mathcal{T} \leftarrow
        \mathcal{T} \cup (j, l+1, 1, t + p_j, x + \alpha k \max\{t + (l+1)s - A, 0\} + q)$.
        /*Deliver the current pending batch and start a new pending batch with job j in it.
      Alternative (3): If $j \le n$, then set $\mathcal{T} \leftarrow \mathcal{T} \cup (j, l, k, t, x + w_j)$.
        /*Schedule job j tardy in the last batch and charge its penalty.
      [FinalCompletion] Do the following only when $j = n + 1$:
        1. If $t = P$ and $v^* > x + x'$, where $x' = \alpha k \max\{t + (l+1)s - A, 0\} + q$,
           then set $v^* = x + x'$. /*Delivering the current pending batch (case (2)).
        2. If $0 < t < P$ and $v^* > x + x'$, where
           $x' = \min\{\alpha k \max\{t + (l+1)s - A, 0\} + 2q, \alpha k \max\{P + (l+1)s - A, 0\} + q\}$
           then set $v^* = x + x'$. /*Delivering the current pending batch and tardy jobs in two
           separate batches or together in a single batch if the latter yields a lower cost (case (3)).
    Endfor
    [Elimination] If $j \le n$, then for any two states $(j, l, k, t, x)$ and $(j, l, k, t, x')$
      with $x < x'$, eliminate from $\mathcal{T}$ the one with $x'$ and set $\mathcal{S}^{(j)} = \mathcal{T}$.
      /*by Remark 3.1.
Endfor
[Optimization] Trace back $v^*$ to obtain an optimal schedule
  and the corresponding assigned due dates.
```

Theorem 3.1 *For the* $1|s, A, DIF|\sum \alpha R_j + \sum w_j U_j + bq$ *problem, Algorithm A1 finds an optimal schedule in* $O(n^3 P)$ *time and space. Thus the problem is* $\mathcal{NP}$*-hard only in the ordinary sense.*

Proof The correctness follows directly from the above discussion. Regarding the complexity, each state $(j-1, l, k, t, x)$ in $\mathcal{S}^{(j-1)}$ gives rise to at most three states containing j and one calculation for updating v^*. The upper bound for the number of triplets $\{j, l, k\}$ is n^3. For each $\{j, l, k\}$, there are at most $P+1$ pairs $\{t, x\}$, because of [Elimination]. Therefore, the overall complexity is $O(n^3 P)$. □

Corollary 3.2 *For the* $1|s, A,DIF|\sum \alpha R_j + \sum w_j U_j + bq$ *problem, if all processing times are equal, i.e.,* $p_j = p > 0$, $\forall j \in J$, *Algorithm A1 finds an optimal schedule in* $O(n^4)$ *time and space.*

Proof Since $p_j = p$, for each $\{j, l, k\}$, there are at most n possible values for t and therefore there are at most n pairs, $\{t, x\}$. By the proof of Theorem 3.1, the run time is $O(n^4)$. □

3.2 Fully polynomial-time approximation scheme

As a classical topic in scheduling, converting a dynamic-programming algorithm into an FPTAS has a long history. Our FPTAS is based on the technique of trimming the execution of the pseudo-polynomial algorithms by partitioning the range of potential solution values into equal-size sub-intervals and keeping only one solution value and schedule for each sub-interval. This technique was introduced by Ibarra and Kim (1975) and was applied to some scheduling problems by Sahni (1976). The application of this technique requires computing a lower and an upper bound for the optimal solution value that are *related* by a preferably small polynomial factor. For the majority of applications these related bounds are easily obtainable. Gens and Levner (1981), Hochbaum and Landy (1994), and Brucker and Kovalyov (1996) presented FPTAS-s for various versions of scheduling to minimize the weighted number of tardy jobs on a single machine. Our $1|s, A,$ DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem substantially extends these models by including due-date assignment, batch-setup times, and batch deliveries (including the delivery of tardy jobs.) The three different components of the schedule cost are only loosely related, but the schedule structure introduces complex trade-off relationships between them. This makes the development of good related bounds for the overall objective very difficult. In fact, the development of such related bounds for the unknown optimum in polynomial time is the most challenging and innovative part of the paper. We also mention that an FPTAS with much higher complexity was presented by Steiner and Zhang (2011) for a somewhat related problem with individual leadtimes but without batch deliveries.

In order to obtain the related bounds, we break down the original problem into $O(n)$ restricted problems, which will yield bounds for the $\sum w_j U_j$ component of the objective. To bound the $\sum \alpha R_j$ component, we introduce an auxiliary problem for each restricted problem. The auxiliary problems are solved by a dynamic-programming algorithm in polynomial time. We are able to prove the somewhat surprising fact that these separately obtained bounds for the different objective components can be *combined* into a pair of related bounds. Discretizing the interval defined by these bounds produces a polynomial number of sub-intervals. Since the overall efficiency of the resulting FPTAS depends on how close the related bounds are to each other, we also employ a bound tightening procedure. The objective value of every partial feasible schedule, computed by the pseudo-polynomial algorithms, falls into one of the sub-intervals. Then, the execution of the algorithms is accelerated to become polynomial by trimming the partial schedules that are produced (when an unscheduled job is considered) keeping only one schedule for each sub-interval. Naturally, eliminating all but one schedule from each sub-interval introduces some errors, but the overall cumulative error does not exceed ε times the optimal solution value, where $\varepsilon > 0$ is an arbitrary error bound.

3.2.1 Bounds analysis

We start by defining a hierarchy of *restricted* and *auxiliary* problems, whose solutions will be used in determining the initial related bounds for our $1|s, A,$DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem.

- *The restricted problems*: The ith restricted problem is denoted by $1|s, A, i,$DIF$|\sum \alpha R_j + \sum w_j U_j + bq$, $i = 0, 1, \cdots, n,$, and it is designed in such a way that any feasible schedule for it is also a feasible schedule for the original $1|s, A,$DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem. Among all the optimal solution values of the restricted problems, the smallest one is the optimal solution value of the $1|s, A,$DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem. The restricted problems have to satisfy two feasibility conditions listed below.

Condition 3.1 *In any feasible schedule for the* $1|s, A, i,$ *DIF*$|\sum \alpha R_j + \sum w_j U_j + bq$ *problem, job i is a tardy job and has the largest tardiness penalty among the tardy jobs. (In particular, if* $i = 0$*, then no job is tardy in any feasible schedule.)*

Condition 3.2 *In any feasible schedule for the* $1|s, A, i,$ *DIF*$|\sum \alpha R_j + \sum w_j U_j + bq$ *problem, all the early jobs are in SPT order in order to satisfy Proposition 3.2.*

Condition 3.1 allows us to have the non-zero lower bound w_i for the $\sum w_j U_j$ component of the optimal solution value of $1|s, A, i,$DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ for any $i > 0$. It also forces every job in $J(i) = \{j | w_j > w_i, j \in J \backslash \{i\}\}$ to be early. To account for this restriction in the $\sum \alpha R_j$ component of the cost, we define an auxiliary problem for each restricted problem.

- *The auxiliary problems*: The ith auxiliary problem is denoted by $1|s, A, J(i), \text{DIF}|\sum \alpha R_j$, $i = 0, 1, \cdots, n$, and it is defined on job set $J(i) = \{j|w_j > w_i, j \in J\backslash\{i\}\}$. Additionally, a feasible schedule of the auxiliary problem has to satisfy Condition 3.2.

We will present polynomial-time solutions for the auxiliary problems. Furthermore, somewhat surprisingly, the solutions for the auxiliary problems and the bounds for $\sum w_j U_j$ can be *combined* to develop *related* bounds for the optimal solution values of the corresponding restricted problems.

Let σ_i and v_i^* be the optimal schedule and the corresponding objective value, respectively, for the ith restricted problem $1|s, A, i, \text{DIF}|\sum \alpha R_j + \sum w_j U_j + bq$. Then, by definition

$$v_i^* = \sum_{j=1}^{n} \alpha R_j(\sigma_i) + \sum_{j=1}^{n} w_j U_j(\sigma_i) + b(\sigma_i)q, \quad (3)$$

where $1 \le b(\sigma_i) \le n$ yields the following bounds for the third term

$$q \le b(\sigma_i)q \le nq. \quad (4)$$

Next, consider bounding the first two terms in Eq. (3). Let $J_E(\sigma_i)$ and $J_T(\sigma_i)$ be the set of early and tardy jobs in σ_i, respectively. By Condition 3.1 job i is tardy in σ_i, and any tardy job j has $w_j \le w_i$ and $J(i) \subseteq J_E(\sigma_i)$. Thus the second component can be bounded by

$$w_i \le \sum_{j=1}^{n} w_j U_j(\sigma_i) \le |J_T(\sigma_i)| w_i, \quad (5)$$

where $|J_T(\sigma_i)|$ denotes the number of jobs in $J_T(\sigma_i)$. Let π_i^* be the optimal solution value for problem $1|s, A, J(i), \text{DIF}|\sum \alpha R_j$. Since $J(i) \subseteq J_E(\sigma_i)$ and σ_i schedules the jobs in $J_E(\sigma_i)$ early, even though it would be feasible to schedule the jobs in $J_E(\sigma_i)\backslash J(i)$ tardy,

$$\pi_i^* \le \sum_{j=1}^{n} \alpha R_j(\sigma_i) \le \pi_i^* + \sum_{j \in J_E(\sigma_i)\backslash J(i)} w_j$$
$$\le \pi_i^* + |J_E(\sigma_i)\backslash J(i)| w_i, \quad (6)$$

where $|J_E(\sigma_i)\backslash J(i)|$ is the number of jobs in $J_E(\sigma_i)\backslash J(i)$. (The right-hand side in the preceding bounds is the cost of a schedule that starts with the schedule corresponding to π_i^* and schedules all the remaining jobs tardy.) Combining Eqs. (4), (5) and (6) and using the fact that $|J_E(\sigma_i)\backslash J(i)| + |J_T(\sigma_i)| \le n$ gives a pair of bounds for v_i^*

$$\pi_i^* + w_i + q \le v_i^* \le \pi_i^* + |J_E(\sigma_i)\backslash J(i)| w_i$$
$$+ |J_T(\sigma_i)| w_i + nq \le \pi_i^* + n w_i + nq. \quad (7)$$

Note that if $J(i) = \emptyset$, then $\pi_i^* = 0$ on this empty set. If $J(i) = J$, which occurs when $i = 0$, then all of the jobs must be early and $w_0 = 0$. We proved the following.

Lemma 3.1 *Let v_i^* be the optimal solution value for the ith restricted problem $1|s, A, i, DIF|\sum \alpha R_j + \sum w_j U_j + bq$ and let π_i^* be the optimal solution value for the ith auxiliary problem $1|s, A, J(i), DIF|\sum \alpha R_j$. A pair of related bounds for the optimal solution value v_i^* of the ith restricted problem can be determined by $L_i' \le v_i^* \le nL_i'$, where $L_i' = \frac{\pi_i^*}{n} + w_i + q$, $i = 0, 1, \ldots, n$.*

3.2.2 Initial bounds determination

To apply the related bounds of Lemma 3.1, we need a solution algorithm for the auxiliary problems. In order to obtain π_i^* for the ith auxiliary problem, a dynamic- programming algorithm (Algorithm A2(i)) is developed.

Assume w.l.o.g. that the jobs in $J(i)$ are re-indexed so that $p_1 \le \cdots \le p_r$, where $r = |J(i)|$. Let $(j-1, l, k, y)$ represent a partial schedule on job set $\{1, 2, \ldots, j-1\} \subseteq J(i)$, where

- y is the due-date-assignment cost for the delivered batches so far;
- l is the number of delivered batches, which are denoted by $B_1, B_2, \ldots, B_l$;
- k is the number of jobs in the current batch, B_{l+1}, which is undelivered/pending (B_{l+1} always consists of the *last* k jobs in the sequence $(1, 2, ..., j-1)$);

There are two ways to add job j to $(j-1, l, k, y)$: (W1) schedule job j into B_{l+1} and (W2) deliver B_{l+1} and start a new pending batch, B_{l+2}, with job j as the first scheduled job in it. Let

$$t(\bar{j}, \bar{l}) = \sum_{h=1}^{\bar{j}} p_h + \bar{l}s \quad (8)$$

denote the makespan of a partial schedule on $\{1, 2, \cdots, \bar{j}\}$ with $\bar{l}$ batches. For case (W1) the generated partial schedule is represented by the state $(j, l, k+1, y)$, where B_{l+1} is pending. In case (W2), B_{l+1} is delivered at $t(j-1, l+1)$, and the generated partial schedule with a delivered B_{l+1} and a pending B_{l+2} is represented by $(j, l+1, 1, y + y(j-1, l+1, k))$, where $y(j-1, l+1, k)$ is the sum of the due-date-assignment costs of the jobs in B_{l+1}. Since B_{l+1} includes k jobs and is delivered at $t(j-1, l+1)$, we can compute

$$y(j-1, l+1, k) = \alpha k \max\{t(j-1, l+1) - A, 0\}$$
$$= \alpha k \max\left\{\sum_{h=1}^{j-1} p_h + (l+1)s - A, 0\right\}. \quad (9)$$

Consider (r, l, k, y), where all the r jobs have been scheduled, and the current batch B_{l+1} has not been delivered yet. Simply delivering B_{l+1} at $t(r, l+1)$ gives a full schedule $(r, l+1, y + y(r, l+1, k))$, where the value of $y(r, l+1, k)$ can be obtained by replacing $j-1$ with r in Eq. (9). This case is denoted by (W3) in Algorithm A2(i). The algorithm starts from an empty schedule $(0, 0, 0, 0)$. Partial schedules for the first $j-1$ jobs $\{1, 2, \ldots, j-1\}$ are included in set $\mathcal{S}^{(j-1)}$, $j = 1, \ldots, r$. In particular, $\mathcal{S}^{(0)} = \{(0, 0, 0, 0)\}$. The following state-reduction remark will also be used.

Remark 3.2 For any two states (j, l, k, y_1) and (j, l, k, y_2) with $y_1 < y_2$, we can eliminate the second one, because any later states generated from it cannot lead to a smaller y value than the value of similar states generated from the first one.

Algorithm A2(i)

```
[Initialization] Determine $J(i) = \{j | w_j > w_i, j \in J \backslash \{i\}\}$ and $r = |J(i)|$, renumber the jobs in $J(i)$ so that $p_1 \le \cdots \le p_r$, set $\mathcal{S}^{(0)} = \{(0,0,0,0)\}$ and $\mathcal{S}^{(j)} = \emptyset$, $j = 1, \ldots, r$.
    Compute and store the values $\sum_{h=1}^{j} p_h$, for all $j = 1, \ldots, r$. /* These values are needed in order to compute each $t(j, l+1)$ using Eq. (8) and each $y(j-1, l+1, k)$ by Eq. (9) in constant time.
[Generation] Generate set $\mathcal{S}^{(j)}$ from $\mathcal{S}^{(j-1)}$.
For $j = 1$ to $r + 1$
    [Setup] Set $\mathcal{T} = \emptyset$.
    For each $(j-1, l, k, y) \in \mathcal{S}^{(j-1)}$
        (W1): If $j \le r$, then set $\mathcal{T} = \mathcal{T} \cup (j, l, k+1, y)$. /*Schedule job $j$ in the current batch, $B_{l+1}$.
        (W2): If $j \le r$, then set $\mathcal{T} = \mathcal{T} \cup (j, l+1, 1, y + y(j-1, l+1, k))$. /*Deliver the jobs in $B_{l+1}$ and schedule job $j$ as the first scheduled job in $B_{l+2}$.
        (W3): If $j = r + 1$, then set $\mathcal{T} = \mathcal{T} \cup (r, l+1, y + y(r, l+1, k))$. /*Delivering the jobs in $B_{l+1}$ completes the partial schedule into a full schedule.
    Endfor
    [Elimination] If $j < r$, then for any two states $(j, l, k, y)$ and $(j, l, k, y')$ with $y < y'$, eliminate from $\mathcal{T}$ the one with $y'$. Set $\mathcal{S}^{(j)} = \mathcal{T}$. /* By Remark 3.2.
Endfor
[Result] Set the optimal solution value $\pi_i^*$ equal to the smallest $y$ among all states in $\mathcal{T}$. /* Note that now $\mathcal{T}$ contains the final schedules.
```

Theorem 3.2 *For the* $1|s, A, J(i), DIF|\sum \alpha R_j$ *auxiliary problem, Algorithm A2(i) finds an optimal solution in $O(n^3)$ time and space.*

Proof The correctness of the algorithm follows from the discussion preceding it. Regarding the complexity, for each $(j-1, l, k, y)$, there are at most two operations. Since $(j, l, k, \cdot)$ always holds the smallest possible y value, there are at most $O(n^2)$ states in each $\mathcal{S}^{(j)}$ (k is a work variable recording the number of jobs in the current pending batch). Since there are at most $n+1$ iterations, the complexity is $O(n^3)$. □

Corollary 3.3 *Let v_i^* be the optimal solution value for the i-th restricted problem. In $O(n^3)$ time and space, Algorithm A2(i) finds a pair of related bounds such that $L'_i \le v_i^* \le nL'_i$, where $L'_i = \frac{\pi_i^*}{n} + w_i + q$.*

Proof The corollary directly follows from Lemma 3.1 and Theorem 3.2. □

Let v^* be the optimal solution value for the original $1|s, A,$ DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem. Since the feasible schedules for all the restricted problems cover all the feasible schedules for the original problem, we know that v^* must fall into one of the non-empty intervals: $[L'_i, nL'_i]$, $i = 0, \ldots, n$. Let $L' = \min_{i=0,\ldots,n} \{L'_i\}$. Then this implies $v^* \ge \min_{i=0,\ldots,n} \{L'_i\} = L'$. Since the optimal solution v_i^* for the ith restricted problem represents a schedule that is also feasible for the original problem, we have $v^* \le \min_{i=0,\ldots,n} \{v_i^*\}$ by the optimality of v^*. Thus using Corollary 3.3, we also have $v^* \le \min_{i=0,\ldots,n} \{nL'_i\} = nL'$. Thus we proved our crucial result for obtaining initial related bounds for our original problem.

Corollary 3.4 *In $O(n^4)$ time, we can determine a pair of initial related bounds for optimal solution value v^* of our original problem such that $v^* \in [L', nL']$, where $L' = \min_{i=0,\ldots,n} \{L'_i\}$.*

3.2.3 Tight bounds determination

The bounds obtained in the previous section are related by a factor of n. This would allow a polynomial size interval partitioning procedure. This size and the time complexity of the resulting FPTAS, however, depend on how close these bounds are to each other. The tighter are the bounds, the better is the complexity. In this section, we introduce a *range approximation algorithm* (Algorithm A3 (u, ε)), which will be called a number of times by a search algorithm (Algorithm A4). This will yield tighter bounds which are related by a small constant factor and thus allow a faster implementation of our FPTAS.

Algorithm A3(u, ε) is a modified, approximation version of Algorithm A1. Instead of finding the optimal solution value (v^*), given a target value $u > 0$ for the unknown v^* and an arbitrary small $\varepsilon > 0$, Algorithm A3(u, ε) reports either that $v^* > (1-\varepsilon)u$ or $v^* \le u$. Algorithm A3(u,ε) applies to the objective the interval partitioning technique. It uses the same state representation (j,l,k,t,x) as Algorithm A1 described there. Initially $v = \infty$ and v is used to keep the objective value of the best full schedule found. The following easy-to-prove remark is similar to Remark 3.1.

Remark 3.3 For all states with the same entries: $(j,l,k,\ldots,x)$, we only need to keep the one which has the smallest t value.

Algorithm A3(u,ε)

```
[Initialization] Set $v = \bar{x} = \sum_{j=1}^{n} w_j + q$, $\mathcal{S}^{(0)} = \{(0,0,0,0,0)\}$ and $\mathcal{S}^{(j)} = \emptyset$, $j = 1,\ldots,n$.
[Partitioning] Partition the interval $[0,u]$ into $\lceil n/\varepsilon \rceil$ equal sub-intervals of size $\varepsilon u/n$, with the last one possibly smaller.
[Generation] Generate set $\mathcal{S}^{(j)}$ from $\mathcal{S}^{(j-1)}$.
For $j = 1$ to $n+1$
    [Setup] Set $\mathcal{T} = \emptyset$.
    For each state $(j-1,l,k,t,x)$ in $\mathcal{S}^{(j-1)}$
        Do the same state generation steps as in Algorithm A1 but use $v$, the best value obtained so far, instead of the optimal $v^*$ everywhere. /In order to save space, we do not repeat the Generation procedure here.
    Endfor
    [Elimination] If $j \le n$, do the following:
        1. If $x > u$, then eliminate from $\mathcal{T}$ any newly generated state $(j,l,k,t,x)$.
        2. For any two states $(j,l,k,t,x)$ and $(j,l,k,t',x)$ with $t \le t'$, eliminate the one with $t'$ from set $\mathcal{T}$. /*by Remark 3.3.
        3. For states $(j,l,k,t,x)$ with values $x$ falling into the same sub-interval, keep only the one with the smallest $t$ value for each sub-interval. /*This guarantees that the x value of the representative state kept for the sub-interval can be larger than the x value of any discarded state by at most $\varepsilon u/n$.
        4. Set $\mathcal{S}^{(j)} = \mathcal{T}$.
Endfor
[Report] If $v > u$, then report $v^* > (1-\varepsilon)u$; otherwise, report $v^* \le u$.
```

Theorem 3.3 *In $O(n^4/\varepsilon)$ time, Algorithm A3(u,ε) either establishes that $v^* > (1-\varepsilon)\,u$ or demonstrates that $v^* \le u$ (by finding a solution with $x \le u$), where v^* is the optimal solution value for the $1|s,A,DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem.*

Proof The correctness of the state generations follows from the same arguments as in the case of Algorithm A1. If $v < u$, then $v^* < u$ because $v^* \le v$. Let us consider now the case $v > u$. Since the algorithm can generate at least one full schedule where all jobs are tardy, we have $v < \infty$. Since v represents the best full schedule found, and the total error introduced is at most εu (in one iteration it is at most $\varepsilon u/n$), $v > u$ implies that any feasible full schedule that could have been generated from a state discarded in Step 3 would have an objective value greater than $u - \varepsilon u = (1-\varepsilon)u$. Thus $v^* > (1-\varepsilon)u$ in this case indeed.

Because of [Partitioning] and [Elimination], for each triplet $\{j,l,k\}$, there are at most $O(\lceil n/\varepsilon \rceil)$ pairs $\{t,x\}$. By the proof of Theorem 3.1, the running time is then $O(n^4/\varepsilon)$. □

In order to tighten the previously obtained related bounds $[L', nL']$, we apply the "*Bound Improvement Procedure*" introduced by Chubanov et al. (2006): Algorithm A4 repeatedly calls Algorithm A3$(u, 1/3)$ (with $\varepsilon = 1/3$) in a binary search on the exponential form $n = 2^{\log n}$.

Algorithm A4

```
1. Set $L = 2^{\lceil \log n \rceil} L'/3$, $l_1 = 0$ and $l_2 = \lceil \log n \rceil$.
2. Set $k = \lceil (l_1 + l_2)/2 \rceil$, $u = 2^{k-1} L'$.
3. Run Algorithm A3$(u, 1/3)$ for the $1|s,A,\mathrm{DIF}|\sum \alpha R_j + \sum w_j U_j + bq$ problem:
    (a) If Algorithm A3$(u, 1/3)$ reports $v^* > (1-\varepsilon)u = 2u/3$, and if $l_2 = k$, then stop; otherwise set $l_1 = k$ and go to step 2. /*$l_1$ gets updated only in this step.
    (b) If Algorithm A3$(u, 1/3)$ reports a solution with value $v \le u$, and if $l_2 = k$, then set $L = u/3$ and stop; otherwise set $l_2 = k$, $L = u/3$ and go to step 2. /*$l_2$ gets updated only in this step.
```

Theorem 3.4 *Algorithm A4 determines L in $O(n^4 \log\log n)$ time, which yields the tight bounds $L \le v^* \le 3L$, where v^* is the optimal solution value for the $1|s,A,DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem.*

Proof We refer the reader for the proof of correctness to the Appendix in the paper by Chubanov et al. (2006). □

3.2.4 Approximation

The FPTAS for the $1|s,A,\mathrm{DIF}|\sum \alpha R_j + \sum w_j U_j + bq$ problem can be obtained by combining the algorithms introduced above: For any given $\varepsilon > 0$, using the bounds $L \le v^* \le 3L$ obtained by Algorithm A4, we run a slightly modified version of Algorithm A3(u,ε), called Algorithm

A5, where $u = (1+\varepsilon/3)3L$. The only difference is that in the [Partitioning] step $[0,u] = [0,(1+\varepsilon/3)3L]$ is partitioned into $n\lceil 3/\varepsilon + 1\rceil$ intervals of size at most $\varepsilon L/n$, so that the cumulative error over n iterations will be no more than εL. (In order to save space, we do not repeat the algorithm here.) Since we know that the problem has an optimal solution value $v^* \le 3L$, Algorithm A5 cannot return the answer $v^* > u = (1+\varepsilon/3)3L$. Thus, it returns a solution v such that $v \le 3L + \varepsilon L = u$. Furthermore, whichever sub-interval of $[L, 3L]$ the optimal v^* falls into, the algorithm will generate an approximate solution v with an error at most εL, i.e., $v \le v^* + \varepsilon L \le (1+\varepsilon)v^*$.

Corollary 3.5 *For any given $\varepsilon > 0$, Algorithm A5 finds a $(1+\varepsilon)$-approximate solution for the $1|s, A,DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem in $O(n^4/\varepsilon)$ time.*

Finally, we summarize the whole FPTAS in Algorithm A6, which calls the previously defined algorithms as components:

Algorithm A6

```
[InitialBoundsPreparation]: For $i = 0$ to $n$
    1. If $i = 0$, i.e., $J(i) = J$, then run Algorithm A2($i$) for the $1|s, A, J(i)$,DIF$|\sum \alpha R_j$ problem and if the answer is $\pi_i^* < \infty$, then obtain initial bounds: $L_i' \le v_i^* \le nL_i'$, where $L_i' = \pi_i^*/n + q$. /*There is no tardy job.
    2. If $i > 0$ and $J(i) \ne \emptyset$, then run Algorithm A2($i$) for the $1|s, A, J(i)$,DIF$|\sum \alpha R_j$ problem and if the answer is $\pi_i^* < \infty$, then obtain initial bounds: $L_i' \le v_i^* \le nL_i'$, where $L_i' = \pi_i^*/n + q + w_i$.
    3. If $i > 0$ and $J(i) = \emptyset$, then set $L_i' = q + w_i$. /*There is no early job.
Endfor
[InitialBounds]: Set $L' = \min_{i=0,\cdots,n}\{L_i'\}$, which yields initial bounds: $L' \le v^* \le nL'$.
[TightBounds]: Run Algorithm A4 for the $1|s, A$,DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem. The answer $L$ yields tight bounds: $L \le v^* \le 3L$.
[Approximation]: Run Algorithm A5 for the $1|s, A$,DIF$|\sum \alpha R_j + \sum w_j U_j + bq$ problem and based on the answer $v$, we obtain an approximation schedule.
```

Theorem 3.5 *For the $1|s, A,DIF|\sum \alpha R_j + \sum w_j U_j + bq$ problem, Algorithm A6 is an FPTAS and finds a $(1+\varepsilon)$-approximate solution in $O(n^4/\varepsilon + n^4 \log\log n)$ time.*

Proof The complexity and correctness follow directly from the properties of the component algorithms. □

4 The problem with equal per-unit due-date-assignment costs and tardiness penalties

In this section, we present a polynomial-time algorithm for $1|s, A$,DIF$|\sum \alpha R_j + \sum w U_j + bq$, i.e., when $w_j = w > 0$ and $\alpha_j = \alpha > 0$. Note that Propositions 2.1, 2.2, and 3.1 can be applied to this problem. By Proposition 3.2 all early jobs are sequenced in SPT order, and we assume w.l.o.g. that $p_1 \le \cdots \le p_n$. The following two propositions are important for this case.

Proposition 4.1 *There is an optimal schedule for the $1|s, A,DIF|\sum \alpha R_j + \sum w U_j + bq$ problem, in which all the early jobs are scheduled before or at $\frac{w}{\alpha} + A$.*

Proof By Proposition 2.2, $\frac{w}{\alpha} + A$ is an upper bound for any assigned due date. □

Proposition 4.2 *There is an optimal schedule for the $1|s, A,DIF|\sum \alpha R_j + \sum w U_j + bq$ problem, where the early jobs are exactly $\{1, 2, \ldots, h\}$, for some h, $0 \le h \le n$, with the convention that $\{1, \ldots, 0\} = \emptyset$. Moreover, all the tardy jobs, if any, are scheduled by themselves in the last batch.*

Proof By Proposition 4.1, there is an optimal schedule where the early jobs are delivered in batches before or at $\frac{w}{\alpha} + A$, and the tardy jobs (if any) are delivered after $\frac{w}{\alpha} + A$ in a single batch. Suppose that there is an early job i and a tardy job k with $p_i > p_k$ in a schedule. Then exchanging jobs i and k may reduce the schedule cost by $\alpha(p_i - p_k)$ and cannot lead to an increase in the other cost components. Exchanging all such pairs will generate an early job set as claimed. □

Proposition 4.2 allows a simple modeling of the $1|s, A$, DIF$|\sum \alpha R_j + \sum w U_j + bq$ problem as a minimum weight path problem in a weighted directed acyclic graph (WDAG) as follows. Consider the WDAG $G = (V, E)$, where the set of vertices is $V = \{(j,l)|0 \le l \le j \le n\}$, and the set of edges is $E = \{(j,l|k,l+1)|0 \le l \le j < k \le n\}$. Any edge $(j,l|k,l+1)$ represents the fact that the $(l+1)$-th batch consists of jobs $\{j+1, \ldots, k\}$. Further, let the source node of the graph be $(0,0)$ and the set of final nodes be $F = \{(n,l)|1 \le l \le n\}$. Then any path π in the graph starting at the source node and ending at some final node has the form $\pi = (j_0, 0), (j_1, 1), \ldots, (j_\ell, \ell)$, where $1 \le \ell \le n$ and $0 = j_0 < j_1 < j_2 < \cdots < j_\ell = n$. Clearly, these paths are in a one-to-one correspondence with all possible schedules obeying the SPT order, (i.e., a batching of the SPT job sequence). Specifically, the l-th batch in this schedule contains jobs $\{j_{l-1}+1, \ldots, j_l\}$. Now let us assign weights to edges. Recall that an edge $(j,l|k,l+1)$ represents the fact that the $(l+1)$-th batch consists of jobs $\{j+1, \ldots, k\}$. Then the completion time of all jobs in the batch is $C_{k,l+1} = \sum_{i=1}^{k} p_i + s(l+1)$. Thus, the weight $w(j,l|k,l+1)$ assigned to edge $(j,l|k,l+1)$ is the sum of the batch-delivery cost q

and the cost of the optimal due-date assignments according to (1):

$$w(j,l|k,l+1) = \begin{cases} q, & \text{if } C_{k,l+1} \le A \\ \alpha(k-j)(C_{k,l+1}-A)+q, & \text{if } A < C_{k,l+1} \le A + \frac{w}{\alpha} \\ w(k-j)+q, & \text{if } C_{k,l+1} > A + \frac{w}{\alpha} \end{cases} . \quad (10)$$

Therefore, the weight of any path π equals the cost of the associated schedule assuming optimal due-date assignments, and we need to find a minimum weight path from the source to some final node in G.

Theorem 4.1 *The $1|s, A, DIF|\sum \alpha R_j + \sum wU_j + bq$ problem can be solved in $O(n^3)$ time.*

Proof Since $|V| = O(n^2)$ and $|E| = O(n^3)$ in the graph G, the minimum weight path can be found in $O(|V| + |E|) = O(n^3)$ time by standard algorithms. □

5 Conclusions and future research

We presented a model for quoting attainable delivery times to minimize delivery costs and tardiness penalties for the supplier. The model can be applied to situations, where orders cannot be delivered to the customer within the original lead times. The general problem is shown to be strongly $\mathcal{NP}$-hard. When all the jobs have the same per-unit due-date-assignment costs, we proved that the problem is $\mathcal{NP}$-hard only in the ordinary sense by providing a pseudo-polynomial algorithm. This algorithm was also converted into an FPTAS. If the problem also has an identical tardiness penalty for all the jobs, the problem can be solved in polynomial time. In summary, we have shown that all the known results for the classical scheduling problem $1||\sum w_jU_j$, and its special cases are extendable to include due-date assignment and batch-delivery costs.

Further research may consider including controllable job processing times or variable batch-delivery costs.

Acknowledgments This research was supported in part by NSERC Discovery Grant 1708-09.

References

Brucker, P., & Kovalyov, M. Y. (1996). Single machine batch scheduling to minimize the weighted number of late jobs. *Mathematical Methods of Operations Research*, *43*, 1–8.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: Review and extensions. *Operations Research*, *58*, 130–148.

Chen, Z.-L., & Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science*, *51*, 614–628.

Chubanov, S., Kovalyov, M., & Pesch, E. (2006). An FPTAS for a single-item capacity economic lot-sizing problem with monotone cost structure. *Mathematical Programming Series A*, *106*, 453–466.

Gens, G. V., & Levner, E. V. (1981). Fast approximation algorithm for job sequencing with deadlines. *Discrete Applied Mathematics*, *3*(4), 313–318.

Gordon, V. S., Proth, J.-M., & Chu, C. (2002a). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, *139*(3), 1–25.

Gordon, V. S., Proth, J.-M., & Chu, C. (2002b). Due date assignment and scheduling: SLK, TWK and other due date assignment models. *Production Planning and Control*, *13*(2), 117–132.

Gordon, V. S., Proth, J.-M., & Strusevich, V. A. (2004). Scheduling with due date assignment. In J. Y. T. Leung (Ed.), *Handbook of scheduling: Algorithms, models and performance analysis* (Vol. 21, pp. 1–22). Boca Raton, FL: CRC Press.

Hall, N. G., & Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, *51*(4), 566–584.

Hochbaum, D. S., & Landy, D. (1994). Scheduling with batching: Minimizing the weighted number of tardy jobs. *Operations Research Letters*, *16*, 79–86.

Ibarra, O., & Kim, C. E. (1975). Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the Associahon for Computing Machinery*, *22*, 463–468.

Kaminsky, P., & Hochbaum, D. S. (2004). Due date quotation models and algorithms. In J. Y.-T. Leung (Ed.), *Handbook of scheduling: Algorithms, models and performance analysis* (pp. 20-1–20-22). Boca Raton, FL: CRC Press.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller & J. W. Thatcher (Eds.), *Complexity of computer computations*. New York: Plenum Press.

Lawler, E. L. (1982). A fully polynomial time approximation scheme for the total tardiness problem. *Operations Research Letters*, *1*(6), 207–208.

Moore, J. M. (1968). An n job, one machine sequencing algorithm for minimzing the number of late jobs. *Management Science*, *15*, 102–109.

Sahni, S. K. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM*, *23*(1), 116–127.

Shabtay, D., & Steiner, G. (2006). Two due date assignment problems in scheduling a single machine. *Operations Research Letters*, *34*(6), 683–691.

Slotnick, S. A., & Sobel, M. J. (2005). Manufacturing lead-time rules: Customer retention versus tardiness costs. *European Journal of Operational Research*, *169*, 825–856.

Steiner, G., & Zhang, R. (2007). Minimizing the weighted number of late jobs with batch setup times and delivery costs on a single machine. In E. V. Levner (Ed.), *Multiprocessor scheduling, theory and applications* (pp. 85–98). Vienna: Itech Education and Publishing.

Steiner, G., & Zhang, R. (2011). Revised delivery-time quotation in scheduling with tardiness penalties. *Operations Research*, *59*(6), 1504–1511.

Thomas, D. J., & Griffin, P. M. (1996). Coordinated supply chain management. *European Journal of Operational Research*, *94*, 1–15.