Optimization Letters (2022) 16:1073–1091
https://doi.org/10.1007/s11590-021-01763-8

ORIGINAL PAPER

Minsum scheduling with acceptable lead-times and optional job rejection

Baruch Mor[1] · **Dana Shapira**[2]

Received: 10 February 2020 / Accepted: 3 June 2021 / Published online: 9 July 2021

Abstract
In our current fast-paced era, customers are often willing to pay extra premium for shorter lead times, which motivates further research on the scheduling problem with due-date assignment and customer-specified lead times. As part of this effort, we extend the classic minsum 'DIF' scheduling model to allow optional job-rejection, thus adding an important component of real-life applications, namely, the possibility that the scheduler decides to process only a subset of the jobs and outsource the disjoint set. The scheduler is penalised for rejecting certain jobs by setting job-dependent rejection costs, and he is limited by a given upper bound on the total rejection cost. The most general version of the minsum DIF problem includes job-dependent *cost parameters* and *lead-times*, and it is strongly NP-hard. Therefore, we study six variants of the problem, where either only the cost parameters or the lead-times are job dependent. All alternatives are extended by optional job-rejection that possibly bounds the constraints or the underlying cost functions. We establish that all studied problems are NP-hard in the ordinary sense and present pseudo-polynomial dynamic programming algorithms and extensive numerical studies for most solutions.

Keywords Single-machine scheduling · Due-date assignment · Minsum · Job-rejection · Dynamic programming

1 Introduction

The objective of scheduling with due-dates is to find the sequence of jobs that optimises performance under specific due-date constraints. In the DIF method for solving this scheduling problem, the due-dates are decision variables (i.e., an integral part of the solution, rather than part of the input), the assignment is unrestricted,

✉ Baruch Mor
baruchm@ariel.ac.il

[1] Department of Economics and Business Administration, Ariel University, Ariel, Israel

[2] Department of Computer Science, Ariel University, Ariel, Israel

and the scheduler is penalised if the job-dependent due-dates exceed given deadlines, known as lead-times. Thus, the scheduler must find the optimal sequence and due-dates that minimise the sum of three component costs: earliness, tardiness, and due-date tardiness. In the original DIF model [15], the scheduler is penalised if the due dates exceed the lead-times, reflecting the potential loss in sales; since delivering early has no cost in this model, the difference between the due date and lead time is represented by a linear function. The DIF method is highly appropriate in the rapidly increasing field of e-commerce, where retail sales are expected to exceed several trillion dollars in the next few years. Indeed, the commitment of the supplier to deliver the goods earlier than the customers' acceptable lead-time is pivotal in e-commerce, and late delivery may void the contract.

Several studies addressed the minsum DIF model, e.g., Shabtay and Steiner [18], who study job-dependent lead-times; Shabtay and Steiner [19, 20] and Leyvand et al. [11], who extend the model by considering controllable job processing times, and Shabtay [16], who addresses the model with batch delivery costs. Others address the minmax version of the DIF model, e.g., Mor et al. [12], who consider job-dependent cost parameters and lead-times; Gerstl and Mosheiov [5, 6], who extend the model to a time window for acceptable lead-times and solve both the minmax and minsum versions, respectively; and Gerstl et al. [3], who focus on the minmax version with either position-dependent processing times or optional job rejection. In the current study, we follow the approach of Gerstl et al. [3] and extend the *minsum* DIF problem to include possible job rejection. Shabtay et al. [17] claim that optional job rejection is justified in highly loaded production industries where the scheduler may prefer to reject or outsource some orders, rather than imposing a greater loss. Subsequently, the supplier may incur a considerable rejection cost that must be evaluated already at the scheduling phase. Numerous studies addressed the combination of due-date scheduling with optional job-rejection, including, among others, Zhao et al. [24], who study due-date assignments with job-rejection and position-dependent processing times; Gerstl and Mosheiov [4], who address scheduling with generalised due dates and job rejection; and Mosheiov and Pruwer [14], who focus on minmax common due-date problems with position-dependent processing times.

Shabtay and Steiner [18] prove that the DIF problem, with *job-dependent* acceptable lead-times and *job-dependent* cost parameters for earliness, tardiness, and due-date tardiness, is strongly NP-hard. Since this proof implies the lack of a pseudo polynomial-time solution unless P is equal to NP, we study several variations of the problem, where either the cost parameters or the lead-times are job-dependent. All alternatives are extended by an optional job-rejection, possibly bounding the constraints or the underlying cost functions. We establish that all studied problems are NP-hard in the ordinary sense and provide pseudo-polynomial DP algorithms. We present an extensive numerical study that proves that all the DP algorithms are extremely efficient for medium-size problems.

This paper is constructed as follows: Sect. 2 provides the formulation of the problem; Sect. 3 presents preliminary analyses and important known results in scheduling theory; Sect. 4 presents the solution to the problem with *job-dependent* cost parameters and a *common* (zero) lead-time, as well as its complementary

problem; Sect. 5 focuses on the variant with *common* cost parameters and *common* lead-time and its complementary problem; Sect. 6 addresses the case of *job-dependent* cost parameters and *common* lead-times; Sect. 7 discusses *common* cost parameters and *job-dependent* lead-times; and Sect. 8 presents our numerical study.

2 Notations and formulation

Given is a set $\mathcal{J}$ of n jobs that need to be processed on a single machine. The processing time of job j is denoted by p_j, $j = 1, \ldots, n$, the total processing time of the jobs in set $\mathcal{J}$ is denoted by $P\left(= \sum_{j\in\mathcal{J}} p_j\right)$, and the longest processing time among all jobs in set $\mathcal{J}$ is denoted by $p_{max}\ \left(= \max_{j\in\mathcal{J}}\left\{p_j\right\}\right)$. The due-date of job j is a decision variable denoted by d_j, $j = 1, \ldots, n$. For a given schedule, let C_j denote the completion time of job j, $j = 1, \ldots, n$. We consider a job-dependent lead-time, denoted by l_j, $j = 1, \ldots, n$; as common in the DIF model, if the due-date of job j is set to be later than its upper bound, the scheduler is penalised. For given schedule and due-dates, let $E_j = \max\left\{0, d_j - C_j\right\}$, $T_j = \max\left\{0, C_j - d_j\right\}$, and $A_j = \max\left\{0, d_j - l_j\right\}$ denote the earliness, tardiness, and due-date tardiness of job j, $j = 1, \ldots, n$, respectively. For each job j, $j = 1, \ldots, n$, three job-dependent unit costs are considered: the earliness unit cost, α_j, the tardiness unit cost, β_j, and the due-date tardiness unit cost, γ_j. Following Seidmann et al. [15] and Shabtay and Steiner [18], the objective function studied is of the minsum type. We extend the DIF model by allowing job rejection, i.e., we consider a *job-dependent* rejection cost, denoted by r_j, $j = 1, \ldots, n$, and we assume a given upper bound on the total rejection cost of all rejected jobs, denoted by R.

Let $\mathcal{J}_P$ and $\mathcal{J}_R$ denote the set of accepted (processed) jobs and the set of rejected (outsourced) jobs, respectively, such that $\mathcal{J} = \mathcal{J}_P \cup \mathcal{J}_R$ and $\mathcal{J}_P \cap \mathcal{J}_R = \emptyset$. The goal is to find the optimal schedule and due-dates that minimise the total (summed) cost of the earliness, tardiness, and due-date tardiness of set $\mathcal{J}_P$, i.e., $Z = \sum_{j\in\mathcal{J}_P}\left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right)$, under the constraint that the total rejection cost of set $\mathcal{J}_R$ is not larger than the rejection cost limit, i.e., $\sum_{j\in\mathcal{J}_R} r_j \le R$. We also consider two complementary problems where the goal is to minimise the total rejection cost, subject to the constraint that the total weighted completion time or the total weighted tardiness cannot exceed a given upper bound, denoted by Q.

We first extend the problem addressed by Shabtay and Steiner [18], wherein the cost parameters are considered to be *job dependent* and the lead-time is *common* to all jobs and is set to zero, i.e., $l_j = 0$, $j = 1, \ldots, n$. Thus, the objective function is $Z = \sum_{j\in\mathcal{J}_P}\left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right)$ and, using the three-field notation, the problem is

$$\boldsymbol{P}1.1 \ :\ 1|l_j = 0,\ \sum_{j\in\mathcal{J}_R} r_j \le R| \sum_{j\in\mathcal{J}_P}\left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right) \tag{1}$$

and its complementary problem,

$$\boldsymbol{P}1.2 \ : \ 1 \ | l_j = 0, \ \sum_{j \in \mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j \right) \leq Q | \sum_{j \in \mathcal{J}_R} r_j.$$

Next, we adjoin job rejection to the problem studied in Seidman et al. [15], i.e., *common* cost parameters and a *common* lead-time. Thus, for $j = 1, \ldots, n$, $\alpha_j = \alpha$, $\beta_j = \beta$, $\gamma_j = \gamma$ and $l_j = l$, and the second problem studied here is

$$\boldsymbol{P}2.1 \ : \ 1 | l_j = l, \ \sum_{j \in \mathcal{J}_R} r_j \leq R | \sum_{j \in \mathcal{J}_P} \left(\alpha E_j + \beta T_j + \gamma A_j \right) \tag{2}$$

and its complementary problem,

$$\boldsymbol{P}2.2 \ : \ 1 \ | l_j = l, \ \sum_{j \in \mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j \right) \leq Q | \sum_{j \in \mathcal{J}_R} r_j.$$

Third, we consider *job-dependent* cost parameters and a *common* lead-time with job-rejection:

$$\boldsymbol{P}3 \ : \ 1 | l_j = l, \ \sum_{j \in \mathcal{J}_R} r_j \leq R | \sum_{j \in \mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j \right) \tag{3}$$

Finally, we assume *common* cost parameters and *job-dependent* lead-times, implying that

$$\boldsymbol{P}4 \ : \ 1 | l_j, \sum_{j \in \mathcal{J}_R} r_j \leq R | \sum_{j \in \mathcal{J}_P} \left(\alpha E_j + \beta T_j + \gamma A_j \right). \tag{4}$$

3 Preliminary analysis

Shabtay and Steiner [18] studied the most general form of the DIF model with acceptable lead-times, i.e., where the unit cost parameters and lead-times are *job-dependent:*

$$1 \parallel \sum \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j \right). \tag{5}$$

The authors referred to the problem as Total Weighted Earliness and Tardiness with Due Date Assignment (*TWETD*) and proved the following essential properties and theorem:

Property 1 *In an optimal schedule, no job is early, i.e.,* $E_j = 0$, *for all* $j = 1, \ldots, n$.

Property 2 *In an optimal schedule, if* $\gamma_j \leq \beta_j$ *then the optimal due date of job* j *is* $d_j = C_j$. *Otherwise,* $d_j = \min \left\{ l_j, \ C_j \right\}$.

$$\text{Let:} \quad w_j = \min\left\{\beta_j, \gamma_j\right\}, \tag{6}$$

Following the above properties, the *TWETD* problem can be re-formulated as

$$1 \parallel \sum w_j \max\left\{0,\ C_j - l_j\right\}. \tag{7}$$

Thus, to solve the *TWETD* problem, one must find the sequence that minimises (7), which leads to the following important theorem:

Theorem 0 *The TWETD problem is equivalent to a* $1 \parallel \sum w_j T_j$ *problem. Therefore, TWETD is strongly NP-hard, even if* $\beta_j = \gamma_j (= w_j)$, $j = 1, \ldots, n$. *Furthermore, the TWETD problem with uniform penalties, i.e., with* $\alpha_j = \beta_j = \gamma_j = w$, $j = 1, \ldots, n$, *is ordinary NP-hard.*

The theorem follows from the facts that $1 \parallel \sum w_j T_j$ is NP-hard in the strong sense and that $1 \mid\mid \sum w T_j$ is NP-hard in the ordinary sense. Since TWETD is strongly NP-hard and a pseudo polynomial-time solution does not exist unless P is equal to NP, we focus here on the special cases in which either the unit cost parameters or the acceptable lead-times are job-dependent; these problems can be solved in polynomial time. Below, we establish that these problems, when extended to allow job rejection, are ordinary NP-hard and present DP solutions.

4 Problem *P*1.1 $1\left|l_j = 0, \sum_{j\in\mathcal{J}_R} r_j \le R\right| \sum_{j\in\mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right)$ and its complementary problem *P*1.2 $1\left|l_j = 0, \sum_{j\in\mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right) \le Q\right| \sum_{j\in\mathcal{J}_R} r_j$

Shabtay and Steiner [18] consider the DIF model with *job-dependent* cost parameters and a *common* lead-time, such that $l_j = 0$ for $j = 1, \ldots, n$. The authors claim that setting the upper bound of the lead-time to zero is realistic when the customer requests that the order is delivered as soon as possible and may even agree to pay for a faster delivery. Based on Properties 1 and 2 and formulation (7), $1\left|l_j = 0\right| \sum_{j\in\mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right)$ is equivalent to the elementary problem $1 \parallel \sum w_j C_j$, where $w_j = \min\left\{\beta_j, \gamma_j\right\}$. Thus, Problem ***P***1.1 is reduced to

$$1 \mid \sum r_j \le R \mid \sum w_j C_j. \tag{8}$$

and its complementary problem, ***P***1.2, is reduced to $1\left|\sum w_j C_j \le Q\right| \sum r_j$.

Cao et al. [1] proved that this problem is binary NP-hard and presented a pseudo polynomial-time DP algorithm and a fully polynomial-time approximation scheme (FPTAS). Their DP computational complexity is $O\left(n^3 p_{max} w_{max}\right)$, where $w_{\max} = \max_{1\le j\le n}\left\{w_j\right\}$. Next, we suggest a DP solution to ***P***1.1 with a faster processing time of $O(n \cdot P \cdot R)$.

Problem $1 \parallel \sum w_j C_j$ is known to be solved by sorting the jobs in a *Weighted Shortest Processing Time* first (*WSPT*) order, i.e., in a non-decreasing order of $\frac{p_j}{w_j}, j = 1, \ldots, n$; hence, we start our DP by sorting the jobs in a WSPT order.

Let *f(j, t, r)* denote the total weighted completion time for the partial schedule of jobs 1, …, *j*, with a completion time *t* and a maximum rejection cost *r*. At each iteration of the DP, one needs to decide whether to accept or reject job *j*. In the former case, the total weighted completion time cost is increased; in the latter, the rejection cost of job *j* does not exceed the current rejection limit. The formal DP (denoted by *DP*1.1) given by its recursion formula is:

Dynamic programming algorithm DP1:

$$f(j,t,r) = \begin{cases} \infty, & p_j > t \text{ and } r_j > r \\ f\left(j-1, t-p_j, r\right) + w_j t, & p_j \le t \text{ and } r_j > r \\ f\left(j-1, t, r-r_j\right), & p_j > t \text{ and } r_j \le r \\ \min\left(f\left(j-1, t-p_j, r\right) + w_j t, f\left(j-1, t, r-r_j\right)\right), & p_j \le t \text{ and } r_j \le r \end{cases} \tag{9}$$

In this equation, the first line reflects an unfeasible case, the second line reflects the option of processing job *j*, the third line represents the option of rejecting job *j*, and the fourth line reflects the case in which job *j* may be either processed or rejected, such that the option that achieves the minimal cost is selected.

The boundary conditions are: $f(0,0,r) = 0$ and $f(0,t,r) = \infty$ if $t \neq 0, t \le P$, for all $r \le R$. $f(j,t,0) = \sum_{k=1}^{j} w_k C_k$, and $t = \sum_{k=1}^{j} p_k$.

The optimal solution is given by $\min\{f(n,t,r) | 0 \le t \le P,\ 0 \le r \le R\}$. Note that, in the case of $R \ge \sum_{j=1}^{n} r_j$, the solution is trivial because all jobs may be rejected with a zero cost. Thus, the variable *r* is assumed to be bounded by *R*.

Theorem 1 *Algorithm DP1.1 solves Problem* ***P***1.1.

Proof By induction on *j*, *t* and *r*.

Base case: The case $j = 0$ indicates an empty set. Therefore, in this case, the total completion time, *t*, is also equal to 0 and $f(0,0,r) = 0$, whereas $f(0,t,r) = \infty$ for $0 \neq t \le P$ and $r \le R$. If $j \neq 0$ and the rejection upper bound is 0, all jobs should be processed. That is, the total completion time is $t = \sum_{k=1}^{j} p_k$ and $f(j,t,0) = \sum_{k=1}^{j} w_k C_k$ is the cost of the processed jobs when no job is rejected.

We assume the correctness of $f(i,s,u)$ in the case that at least one of $0 \le i < j$ or $0 \le s < t$ or $0 \le u < r$ holds, and we prove the correctness of $f(j,t,r)$. In the case that job *j* must be processed, that is, its rejection cost cannot be added to the total rejection cost without violating the rejection upper bound, i.e., $r_j > r$, the reference cost is $f\left(j-1, t-p_j, r\right)$, which is assumed to be correct by the induction hypothesis. This cost considers the first $j-1$ jobs with a total completion time $t - p_j$, which is applicable only when the processing time of job *j* can fit in the total completion time, i.e., $p_j \le t$. The cost is updated by $w_j t$, considering the additional cost for processing job *j* and leaving the total rejection cost unchanged, as seen on line

2 of Eq. (9). If the processing time of job j cannot fit in the total completion time, i.e., $p_j > t$, then the job should be rejected. Accordingly, the cost is equal to that of $f\left(j-1, t, r-r_j\right)$, which is known to be correct by the induction hypothesis. This is applied only in the case of $r_j \leq r$; otherwise, the case is impossible with infinity cost, as reflected by the first line of Eq. (9). In cases where the job may be either processed or rejected, its processing time is not higher than the total completion time. In these cases, the rejection cost can be added without violating the rejection upper bound, i.e., $p_j \leq t$ and $r_j \leq r$, and the minimum cost between case 2 and case 3 is chosen. By the induction hypothesis, as $f\left(j-1, t-p_j, r\right)+w_j t$ and $f\left(j-1, t, r-r_j\right)$ are optimal, the minimum is optimal for $f(j, t, r)$. □

Theorem 2 *The computational complexity of DP*1.1 *is* $O(n \cdot P \cdot R)$.

Proof Using the recursive formula in (9), the DP is calculated for every job j, $1 \leq j \leq n$, every possible processing time (which is bounded by P), and every rejection cost r (which is bounded by R), resulting in a processing time of $O(n \cdot P \cdot R)$. The calculation of the optimal solution, $f\left(n, t_0, r_0\right) = \min\{f(n,t,r) | 0 \leq t \leq P,\ 0 \leq r \leq R\}$, is done in $O(R)$. The reconstruction of the solution is done by backtracking, starting at $f\left(n, t_0, r_0\right)$ and ending at $f(0,0,0)$, for an addition of $O(n+P+R)$ operations. We conclude that the total processing time is still $O(n \cdot P \cdot R)$. □

Example 1 Consider the following six-job problem, in which the rejection cost limit is $R = 72$ and the common lead-time is $l = 0$. The processing times are $p_j = (18,\ 26,\ 35,\ 40,\ 30,\ 31)$, the rejection costs are $r_j = (35,\ 47,\ 43,\ 14,\ 32,\ 46)$, and the job-dependent cost parameters are $\alpha_j = (13,\ 12,\ 19,\ 23,\ 21,\ 1)$, $\beta_j = (3,\ 9,\ 8,\ 4,\ 3,\ 2)$, and $\gamma_j = (4,\ 4,\ 4,\ 23,\ 10,\ 20)$. Our solution starts by utilizing $w_j = \min\left\{\beta_j, \gamma_j\right\}$, resulting in $w_j = (3,\ 4,\ 4,\ 4,\ 3,\ 2)$. Subsequently, $\frac{p_j}{w_j} = (6.0,\ 6.5,\ 8.75,\ 10.0,\ 10.0,\ 15.5)$, implying that the jobs are sequenced in a WSPT order.

Executing DP1.1, the set of rejected jobs is $\mathcal{J}_R = \left\{J_3,\ J_4\right\}$ with a total rejection cost $\sum_{j \in \mathcal{J}_R} r_j = 57 \leq 72 = R$.

The set of accepted jobs is the complementary set $\mathcal{J}_P = \left(J_1,\ J_2,\ J_5,\ J_6\right)$, the jobs' due dates are $d_j = (0,\ 44,\ 0,\ 0)$, their tardiness values are $T_j = (18,\ 0, 74,\ 105)$, and their due-tardiness values are $A_j = (0,\ 44,\ 0,\ 0)$. Consequently, the total processing cost is $Z = 662$.

In the next subsection, we focus on Problem $\boldsymbol{P}$1.2. As mentioned above, this is the complemeary problem of $\boldsymbol{P}$1.1 and the goal is to minimize the total rejection costs, subject to the constraint that the total weighted completion cannot exceed a given upper bound (Q). Thus, our problem, denoted as $\boldsymbol{P}$1.2, is $1 \left|\sum w_j C_j \leq Q\right| \sum r_j$. Shabtay et al. [17] indicated that since the decision version of Problem $\boldsymbol{P}$1.1 is identical to that of $\boldsymbol{P}$1.2, then $\boldsymbol{P}$1.2 is also NP-hard. Hence, we turn our attention toward providing a DP solution for the problem at hand.

Let $g(j,t,r)$ denote the total rejection cost for the partial schedule of jobs $1,\ldots,j$, having a completion time t and a total rejection cost r. Similar to DP1.1, we first sort the jobs in a WSPT order. At each iteration of the DP, the scheduler then needs to decide whether to reject job j, implying that the total rejection cost is increased, or accept job j, which increases in the total weighted completion time. In the suggested DP, we must keep track of the scheduling measure to ensure that its value does not exceed the dictated upper bound. To this end, we utilize a variable, h_j, which represents the total weighted completion time of the accepted jobs of subset $\{1,\ldots,j\}$, defined as follows:

$$h_j = \begin{cases} h_{j-1} + w_j t, & \text{if job } j \text{ is accepted} \\ h_{j-1}, & \text{else} \end{cases}, \quad \text{where} \quad h_0 = 0.$$

Dynamic programming algorithm DP1.2:

$$g(j,t,r) = \begin{cases} \infty, & p_j > t \quad \text{and} \quad r_j > r \\ g\big(j-1, t-p_j, r\big), & p_j \le t \quad \text{and} \quad h_j \le Q \quad \text{and} \quad r_j > r \\ g\big(j-1, t, r-r_j\big) + r_j, & p_j > t \quad \text{and} \quad r_j \le r \\ \min \begin{cases} g\big(j-1, t-p_j, r\big) \\ g\big(j-1, t, r-r_j\big) + r_j, \end{cases} & p_j \le t \quad \text{and} \quad h_j \le Q \quad \text{and} \quad r_j \le r \end{cases} \tag{10}$$

In this equation, the first line reflects an unfeasible case, the second line reflects the option of processing job j, the third line represents the option of rejecting job j, the fourth line reflects the case in which job j may be either processed or rejected and the option that achieves the minimal cost is selected. The boundary conditions are $g(0,0,0) = 0$, $g(0,t,r) = \infty$ if $0 < t \le \sum p_j = P$, and $0 < r \le \sum r_j$. The optimal solution is given by $\min\left\{g(n,t,r) | 0 \le t \le P,\ 0 \le r \le \sum r_j\right\}$.

Theorem 3 *Algorithm DP1.2 solves Problem P1.2 and its computational complexity is* $O\big(n \cdot P \cdot \sum r_j\big)$.

Proof The proof is similar to those given for Theorems 1 and 2. Therefore, it is omitted here.

5 Problem $P2.1\ 1\left|l_j = l,\ \sum_{j\in\mathcal{J}_R} r_j \le R\right| \sum_{j\in\mathcal{J}_P} \left(\alpha E_j + \beta T_j + \gamma A_j\right)$ and its complementary problem $P2.2\ 1\left|l_j = l,\ \sum_{j\in\mathcal{J}_P} \left(\alpha E_j + \beta T_j + \gamma A_j\right) \le Q\right| \sum_{j\in\mathcal{J}_R} r_j$

In this section, we extend the classic DIF model introduced by Seidman et al. [15], which postulates *common* cost parameters and a *common* lead time, by considering job-rejection. Employing Property 2 and setting $w = \min\{\beta,\gamma\}$ and $l_j = l$, for $j = 1,\ldots,n$ in (7), we obtain

$$1 \left| l_j = l, \ \sum r_j \le R \right| w \sum \max\left(0, \ C_j - l\right). \tag{11}$$

Shabtay and Steiner [18] show that the *TWETD* problem is equivalent to the well-known problem $1\left|d_j = d\right| \sum T_j$. Replacing l_j by d_j, for $j = 1, \ldots, n$ and l by d, we conclude that Problem ***P***2.1 can be reformulated as:

$$1 \left| d_j = d, \ \sum r_j \le R \right| w \sum T_j, \tag{12}$$

and its complementary problem, ***P***2.2, is reduced to

$$1 \left| d_j = d, \ w \sum T_j \le Q \right| \sum r_j.$$

Zhang et al. [23] proved that (12) is binary NP-hard by reduction from *Knapsack*; subsequently, Problem ***P***2.1 is also NP-hard. Below, we provide a pseudo-polynomial time DP algorithm for Problem ***P***2.1, establishing that it remains ordinary NP-hard. Following Seidman et al. [15], we start the DP by sorting the jobs in a Shortest Processing Time first (SPT) order.

At each iteration of the DP, we compute the minimum due-date tardiness of jobs 1 to j that have a total processing time t and an upper bound r on the rejection cost. The computation is based on the results obtained for jobs 1 to $j-1$ that have:

(i) Completion time $t - p_j$ and an upper bound rejection cost r;
(ii) Completion time t and an upper bound rejection cost $r - r_j$.

At each stage, one needs to decide whether to accept or reject job j:

(i) Job j *must* be accepted if its rejection cost exceeds the current rejection limit r.
(ii) Job j *may* be accepted if its contribution minimises the total due-date tardiness.
(iii) Job j *may* be rejected if it minimises the objective function.

The recursive formula is given in the following dynamic programming algorithm DP2.1:

$$f(j,t,r) = \begin{cases} \infty, & p_j > t \quad \text{and} \quad r_j > r \\ f\left(j-1, t-p_j, \ r\right) + w \max\left(0, \ t-l\right), & p_j \le t \quad \text{and} \quad r_j > r \\ f\left(j-1, t, r-r_j\right), & p_j > t \quad \text{and} \quad r_j \le r \\ \min \begin{cases} f\left(j-1, t-p_j, \ r\right) + w \max\left(0, \ t-l\right) \\ f\left(j-1, t, r-r_j\right) \end{cases}, & p_j \le t \quad \text{and} \quad r_j \le r \end{cases} \tag{13}$$

The boundary conditions are $f(0,0,r) = 0 \quad \text{for } 0 \le r \le R, \ f(0,t,r) = \infty$ for $0 \le r \le R, \ 0 < t \le P$ and $f(j,t,0) = \sum_{k=1}^{j} w \ \max\left(0, t-l\right), \ \text{and} \ \ t = \sum_{k=1}^{j} p_k$. The optimal solution is given by $\min\left(f(n,t,r) | 0 \le t \le P, \ 0 \le r \le R\right)$.

Theorem 4 *Algorithm DP2 solves Problem* ***P2.1****.*

Proof The correctness proof is identical to the proof of Theorem 1 by substituting the additional cost incurred by job j $w_j t$ by $w \max(0,\ t-l)$. □

Theorem 5 *The computational complexity of* DP2.1 *is* $O(n \cdot P \cdot R)$.

Proof The proof is similar to the proof of Theorem 2. □

Example 2 Assume a six-job problem with $R=13$, $\alpha=16$, $\beta=5$, $\gamma=15$, and $l=15$. The processing times (sequenced in an SPT order and renumbered) are $p_j=(6,\ 7,\ 8,\ 9,\ 11,\ 16)$, and their rejection costs are $r_j=(11,\ 8,\ 2,\ 3,\ 9,\ 13)$. Since $\beta=5\le 15=\gamma$, all processed jobs are tardy, i.e., $A_j=0$, $\forall j\in\mathcal{J}_P$. Applying *DP*2, the following optimal solution is attained.

The set of rejected jobs is $\mathcal{J}_R=\{J_2,\ J_3,\ J_4\}$, with a total rejection cost of $\sum_{j\in\mathcal{J}_R} r_j=13\le 13=R$.

The set of accepted jobs is $\mathcal{J}_P=(J_1,\ J_5,\ J_6)$, the due dates of these jobs is $d_j=(6,\ 15,\ 15)$, and their tardiness is $T_j=(0,\ 2,\ 18)$. The total processing cost is, therefore, $Z=100$.

In what follows, we concentrate on the complementary problem of ***P***2.1. Our goal is to minimise the total rejection cost subject to the constraint that the total weighted tardiness cannot exceed a given upper bound (Q). Thus, our problem, denoted ***P***2.2, is $1\left|w\sum T_j\le Q\right|\sum r_j$. Let $g(j,t,r)$ denote the total rejection cost for the partial schedule of jobs $1,\ldots,j$, with a completion time t and a total rejection cost r. At each iteration of the DP, the scheduler must decide whether to reject job j, implying that the total rejection cost is increased, or to accept job j, and that the total weighted tardiness time is increased. Similar to the explanation of DP1.2, in the current DP, we need to keep track to the total weighted tardiness, which is defined as follows:

$$h_j=\begin{cases} h_{j-1}+w\max\{0,\ t-l\}, & \text{if job } j \text{ is accepted}\\ h_{j-1}, & \text{else}\end{cases},\quad \text{where}\quad h_0=0.$$

Dynamic programming algorithm DP2.2:

$$g(j,t,r)=\begin{cases}\infty, & p_j>t \quad\text{and}\quad r_j>r\\ g\left(j-1,t-p_j,r\right), & p_j\le t \quad\text{and}\quad h_j\le Q \quad\text{and}\quad r_j>r\\ g\left(j-1,t,r-r_j\right)+r_j, & p_j>t \quad\text{and}\quad r_j\le r\\ \min\begin{cases} g\left(j-1,t-p_j,r\right)\\ g\left(j-1,t,r-r_j\right)+r_j\end{cases}, & p_j\le t \quad\text{and}\quad h_j\le Q \quad\text{and}\quad r_j\le r\end{cases} \tag{14}$$

The boundary conditions are $g(0,0,0)=0$, $g(0,t,r)=\infty$ if $0<t\le\sum p_j=P$, and $0<r\le\sum r_j$. The optimal solution is given by $\min\left\{g(n,t,r)|0\le t\le P,\ 0\le r\le\sum r_j\right\}$.

Theorem 6 *Algorithm DP2.2 solves Problem* **P**2.2 *and its computational complexity is* $O\left(n \cdot P \cdot \sum r_j\right)$.

Proof The proof is similar to those given for Theorems 4 and 5.

6 Problem *P*3 $1\left|l_j = l, \sum_{j\in\mathcal{J}_R} r_j \le R\right| \sum_{j\in\mathcal{J}_P} \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right)$

Next, we consider *job-dependent* cost parameters and a *common* lead-time with optional job-rejection, formally given in (3) and rewritten as:

$$1 \left| \; l_j = l \,, \;\; \sum r_j \le R \; \right| \sum \left(\alpha_j E_j + \beta_j T_j + \gamma_j A_j\right). \tag{14}$$

Using Property 2 and setting $w_j = \min\left\{\beta_j, \gamma_j\right\}$ and $l_j = l$ for $j = 1, \ldots, n$ in (7), we obtain:

$$1\left| \; l_j = l, \;\; \sum r_j \le R \right| \sum w_j \max\left(0, \; C_j - l\right). \tag{15}$$

If job-rejection is not allowed, then the problem is equivalent to minimising the total weighted tardiness with a common due date, i.e., $1\left|d_j = d\right| \sum w_j T_j$. This problem was proved to be NP-hard by Yuan [22], hence, ***P***3 is also NP-hard. Below, we present an $O\left(n^2 \cdot l \cdot R\right)$ DP algorithm solution for ***P***3.

Kellerer and Strusevich [8] propose an $O\left(nP\left(W^{UB}\right)^2\right)$ time DP algorithm solution, where W^{UB} is the upper bound on the total weighted tardiness with a common due date on a single machine. Kianfar and Moslehi [9] suggest an $O\left(n^2 d\right)$ pseudo-polynomial DP algorithm time extending the algorithm of Kacem [7]. This running time also improves the complexity of the algorithm of Kellerer and Strusevich because $n < W^{UB}$ and $d < P$ for non-trivial cases. Our solution to ***P***3 also extends the solution of Kacem [7] by considering all cases and the option of job rejection [13]. For completeness of exposition, we report it briefly here. Early jobs are scheduled starting at time 0, while tardy jobs are scheduled so that they are completed at time P, going backwards. Let t denote the completion time of the last early job (scheduled before l), let $r \in \{0, 1, \ldots, R\}$ denote the allowed rejection costs, and let f denote the minsum DIF of the corresponding schedule. The DP solution, given in *Algorithm DP*3, generates a set of states, v_k, for each iteration k, $1 \le k \le n$. Each state in v_k is represented by an ordered triple (r, t, f), implying a completion time t, a rejection cost r, and a cost f for the first k jobs. The initial sorting of the jobs is in a Weighted Longest Processing Time first (WLPT) rule order; however, eventually, the tardy jobs are sequenced in accordance with the WSPT rule. In an optimal schedule, the early jobs are processed starting at time zero, and they may be followed by a *straddling job* that starts before time l and is completed, at least, at time l; the straddling job, in turn, is followed by the tardy jobs. The loop starting on line 6 of Algorithm *DP*3, below, addresses straddling jobs, as each job can potentially be the straddling job; the remaining $n - 1$ jobs can be either early (line 4) or late (line 5), but not straddling. When a job k is rejected, the cost of the tardy jobs scheduled

thus far—those scheduled to the right of k, i.e., jobs with $p_j/w_j \geq p_k/w_k$—must be updated. For a given schedule σ, let $q_T \subseteq \mathcal{J}_P$ be the set of tardy jobs of σ. The following Lemma is proved in Mor and Shapira [13].

Lemma 1 *The difference in the* minsum DIF *between accepting and rejecting job k is* $\sum_{\substack{j\in q_T \\ j<k}} w_j p_k$.

It follows that, for each state, we additionally store the summation of the weights of the tardy jobs that have already been scheduled. Thus, the value of the minsum DIF is updated in constant time, when a certain job k is rejected. In Algorithm *DP*3 the loop $r \in \{0, 1, \ldots, R\}$, which starts on line 2 of this algorithm, examines, on line 5(iii), all possible rejection costs.

Dynamic programming algorithm *DP*3:

Dynamic programming algorithm *DP*3:

1. For $j \in \{1, 2, \ldots, n\}$ // job j is examined as a straddling job
   $J = \{1, 2, \ldots, n\}$
   $P = \sum_{i\in J} p_i$ $S = J \backslash \{j\}$
2. For $r \in \{0,1, \ldots, R\}$
3. For $k \in S$
   i. Set $v_0^j = \{(0,0,0)\}$.
   ii. For each state (r, t, f) in v_{k-1}^j;
4. If $t + p_k \leq l$
   Put $(r, t + p_k, f)$ in v_k^j
5. Put $(r, t, f + w_k)$ in $v_{S\cap\{1,\ldots,k\}}^j$
   iii. If $r_k \leq r$
   For every state $(r - r_k, t, f,)$in v_{k-1}^j;
   Put $\left(r, t, f - p_k \sum_{\substack{i\in S\cap\{1,\ldots,k-1\} \\ i \text{ is tardy in } (r-r_k,t,f,)}} w_i\right)$ in v_k^j
6. For every state (r, t, f) in v_S^j
   If $((t < l)$ and $(t + p_j \geq l))$
   Put $\left(r, t, f + w_j\left(t + p_j - l\right)\right)$ in v_n^j
7. Return $\min_{(r,t,f)\in v_n^j} \{f\}$.

Theorem 7 *Algorithm DP3 solves Problem* ***P3*** *and its computational complexity is* $O\left(n^2 \cdot l \cdot R\right)$.

Proof The correctness of *DP3* follows from the reduction of $\boldsymbol{P}3$ to $1\left|\, l_j = l,\ \sum r_j \le R \right| \sum w_j \max\left(0,\ C_j - l\right)$ and from the correctness of Lemma 1. For the running time of *DP3*, the inner loop is applied $n-1$ times on each triplet in v_{k-1}^j at each iteration $1 \le k \le n$. Each triplet can be examined to see whether step (1) or step (3) should be applied, such that traversing the states in v_{k-1}^j is, in fact, done only once—in a linear order. As stated by Kacem [7], the complexity of the inner loop of *DP*3 is proportional to $\sum_{k=1}^{n} \left|v_k^j\right|$, where $\left|v_k^j\right|$ denotes the size of v_k^j. By choosing the state (r,t,f) with the smallest value f at each iteration k and for every t, the time is bounded by $O(n \cdot l)$, as d is the maximum value that t can reach. The next outer loop is then applied R times for each possible total rejection cost. Finally, the outer loop is applied n times for each potential straddling job, for a total of $O\left(n^2 \cdot l \cdot R\right)$. Returning the minimum cost on the last line requires a linear scan of all ordered triples in v_n^j, which does not increase the asymptotic running time. Therefore, the total running time is $O\left(n^2 \cdot l \cdot R\right)$. □

Example 3 Consider the following six-job problem, where the rejection cost limit is $R = 12$ and the common lead time is $l = 28$. The processing times, sequenced in WLPT order and renumbered, are $p_j = (27,\ 12,\ 32,\ 27,\ 32,\ 5)$, with rejection costs $r_j = (8,\ 11,\ 7,\ 9,\ 1,\ 6)$. The job-dependent cost parameters are $\alpha_j = (20,\ 16,\ 17,\ 5,\ 14,\ 8)$, $\beta_j = (1,\ 15,\ 7,\ 8,\ 11,\ 16)$, and $\gamma_j = (4,\ 1,\ 7,\ 18,\ 17,\ 12)$.

Calculating the weights using (6), we obtain $w_j = \min\left\{\beta_j, \gamma_j\right\} = (1,\ 1,\ 7,\ 8,\ 11,\ 12)$ and, subsequently, $\frac{p_j}{w_j} = (27.000,\ 12.000,\ 4.571,\ 3.375,\ 2.909,\ 0.417)$. Applying DP3, the set of rejected jobs is $\mathcal{J}_R = \left\{J_3,\ J_5\right\}$, implying that $\sum_{j \in \mathcal{J}_R} r_j = 8 \le 12 = R$. The set of accepted jobs is $\mathcal{J}_P = \left(J_6,\ J_4,\ J_2,\ J_1\right)$. Using Property 2, $d_j = (5,\ 28,\ 64,\ 28)$ and, therefore, $T_j = (0,\ 4,\ 0,\ 43)$, $A_j = (0,\ 0,\ 16,\ 0)$, and the total processing cost is $Z = 91$.

7 Problem *P*4 $1\left|l_j, \sum_{j \in \mathcal{J}_R} r_j \le R\right| \sum_{j \in \mathcal{J}_P} \left(\alpha E_j + \beta T_j + \gamma A_j\right)$

In this section, we study *common* cost parameters and *job-dependent* lead-times. Setting $w = \min\{\beta, \gamma\}$ in (6), $\boldsymbol{P}4$ can be reformulated as:

$$1\left|l_j,\ \sum r_j \le R\right| w \sum \max\left(0,\ C_j - l_j\right). \tag{16}$$

Denoting by $\boldsymbol{P}'4$ the problem $\boldsymbol{P}4$ without the option of job-rejection, it follows from (16) that $\boldsymbol{P}'4$ is equivalent to the fundamental problem, $1||\sum T_j$. This fundamental problem was studied by Lawler [10] and proved by Du and Leung [2] to be ordinary NP-hard. Assuming that the weights of the jobs are agreeable, i.e., that $p_i < p_j$ implies that $w_i \ge w_j$, Lawler showed that an optimal sequence can be found by a DP algorithm with a worst-case running time of $O\left(n^4 P\right)$ or $O\left(n^5 p_{\max}\right)$. Steiner

and Zhang [21] studied a problem related to $P'4$, namely, $1\left|l_j\right|\sum \max\left(0,\ d_j - l_j\right) + \sum \beta_j U_j$, where α is the delivery time quotation cost per extended time unit, β_j is the job-dependent tardiness cost (which can be treated as a rejection cost), and U_j is a binary variable, such that $U_j = \begin{cases} 1, & C_j > d_j \\ 0, & C_j \le 0 \end{cases}, j = 1, \ldots, n.$ They proved that this problem is equivalent to the problem of minimising the total tardiness with rejection, and they presented an $O\left(n^4P^3\right)$ time DP algorithm and an $O\left(n^4Z^3\right)$ time FPTAS algorithm, where Z is an upper bound on the optimal solution, based on Lawler [10]. Based on the DP provided in Lawler [10], we introduce a DP algorithm with a worst-case running time of either $O\left(n^4 \cdot P \cdot R\right)$ or $O\left(n^5 \cdot p_{\max} \cdot R\right)$ for Problem ***P***4, proving that it remains NP-hard in the ordinary sense.

First, we claim that, because $w_j = w,\ \ j = 1, \ldots, n$ in our problem, Lawler's agreeability assumption is clearly valid. We start by ordering the jobs in an Earliest Due Date first (*EDD*) order, $1, 2, \ldots, n$. Following Lawler, we denote by $S(i,j,k)$ the subset of jobs in $i, i+1, \ldots, j$ that have processing times shorter than those of job k, formally $S(i,j,k) = \left\{j' \middle| i \le j' \le j, p_{j'} < p_k\right\}$. We define $T(S(i,j,k),t,r)$ as the total weighted tardiness for an optimal schedule of the jobs in $S(i,j,k)$, starting at time t and with a total rejection cost R.

Lawler [10] proved that for some δ, $0 \le \delta \le n-k$ there exists an optimal schedule so that:

1. Jobs $1, 2, \ldots, k-1, k+1, \ \ldots, k+\delta$ in some sequence start at time t, followed by
2. Job k, with a completion time of $C_k(\delta) = t + \sum_{j \le k+\delta} p_j$, followed by
3. Jobs $k+\delta+1, k+\delta+2, \ldots, n$ in some sequence starting at time $C_k(\delta)$.

For simplicity, we present the DP algorithm in two stages: first for computing the total weighted tardiness, and then for allowing rejection. Based on Lawler's DP algorithm:

$T(S(i,j,k),t,r) = \min_\delta\left\{T\left(S\left(i,k'+\delta,k'\right),t,r\right) + w_{k'} \cdot \max\left(0,\ C_{k'}(\delta) - d_{k'}\right) + T\left(S\left(k'+\delta+1,j,k'\right),C_{k'}(\delta),r\right)\right\}$, where k' is the job with the maximal processing time within $S(i,j,k)$, and

$$C_{k'}(\delta) = t + \sum_{j' \le k'+\delta} p_{j'}.$$

$$T(\{1,2,\ldots,n\},t,r) = \min\left(T(\{1,2,\ldots,n\},t,r), \min_k\left\{T\left(S(i,j,k),t,r-r_k\right)\right\}\right). \tag{17}$$

The boundary conditions are $T\left(\emptyset,t,r\right) = 0$, $T(\{j\},t,0) = w_j \cdot \max\left(0,\ t - p_j - l_j\right)$ and if $r > 0$, $T(\{j\},t,r) = \begin{cases} 0 & r_j \le r \\ T(\{j\},t,0) & \text{else} \end{cases}$.

Theorem 8 *Algorithm DP4 solves Problem* ***P4*** *and its computational complexity is* $O\left(n^4 \cdot P \cdot R\right)$.

Proof By induction on r. The correctness of the base case of the induction follows directly from the correctness of $T(S(i,j,k),t,r)$ for $r = 0$ using the truth of the algorithm of Lawler. The correctness of $T(\{1,2,\ldots,n\},t,r)$ then follows from the induction hypothesis, which assumes correctness for all values lower than r and for all groups of cardinality lower than n in Eq. (17).

The correctness of the running time follows directly from the definition of $T(S(i,j,k),t,r)$. The group $S(i,j,k)$ is defined for each i,j,k, where $1 \le i,j,k \le n$, t is defined for each $1 \le t \le P$, and r is defined for each $1 \le r \le R$. Calculating $T(S(i,j,k),t,r)$ requires another factor of n for the minimum calculation running over all δ s, $1 \le \delta \le n$. □

Example 4 Assume a problem of five jobs with $R = 5$, $\alpha = 7$, $\beta = 6$, $\gamma = 11$, $w = \min\{\beta,\gamma\} = 6$, and $l_j = (24,\ 28,\ 35,\ 36,\ 67)$. The processing times, sequenced in their EDD order, are $p_j = (17,\ 16,\ 14,\ 19,\ 15)$, and their rejection costs are $r_j = (3,\ 3,\ 4,\ 3,\ 2)$. Since $\beta = 6 \le 11 = \gamma$, all processed jobs are tardy. By applying the solution shown above, the set of rejected jobs in the optimal solution is $\mathcal{J}_R = \{J_1,\ J_5\}$, with a total rejection cost of $5 \le 5 = R$. The set of processed jobs is $\mathcal{J}_P = (J_2,\ J_3,\ J_4)$, their due-dates are $d_j = (16,\ 30,\ 36)$, and their tardiness is $T_j = (0,\ 1,\ 13)$. The total processing cost is, therefore, $Z = 84$.

8 Numerical study

We performed numerical tests to measure the running time of algorithms *DP*1.1, *DP*2.1, and *DP*3. Throughout this experimental study, the job processing times (p_j) and rejection costs (r_j) were generated uniformly in the interval [1, 50], and the cost parameters ($\alpha_j, \beta_j, \gamma_j$) were generated uniformly in the interval [1, 20]. We denote by $\bar{p}$ and $\bar{r}$ the maximum possible value of jobs' processing times and rejection costs, respectively; thus, $\bar{p} = \bar{r} = 50$. The different C++ programs were executed on an Intel (R) Core™ i7-8650U CPU @ 1.90 GHz, 16.0 GB RAM platform. All numerical results are presented in tables constructed in the same format; the columns indicate the number of jobs, n, the range of the rejection cost, R, and the average and worst-case running times.

*DP*1.1: Instances with $n = 15$, 30, 45, and 60 jobs were generated, where the cost parameters are *job-dependent* and the *common* lead time set to zero. The upper bound on the total rejection cost, R, was generated uniformly in three intervals: $[0.0025,\ 0.0050]\bar{r}n$, $[0.0075,\ 0.010]\bar{r}n$, and $[0.015,\ 0.020]\bar{r}n$. Note that these intervals roughly reflect rejections of 5%, 10%, and 15% of the jobs, respectively. For each combination of n and R, 20 instances were generated and solved. The results in Table 1 demonstrate that *DP*1.1 is efficient for solving medium-size

Table 1 Average and worst-case running times of the ***DP*1** algorithm for Problem ***P*1**, with either 5%, 10%, or 15% rejected jobs (top, middle, and bottom tables, respectively)

n	R	Average running time (ms)	Worst-case running time (ms)
Approximately 5rejected jobs			
20	[2, 5]	<1	<1
30	[3, 7]	<1	<1
40	[5, 10]	1	1
50	[6, 12]	1	1
60	[7, 15]	2	3
Approximately 10% rejected jobs			
20	[7, 10]	<1	<1
30	[11, 15]	1	1
40	[15, 20]	1	2
50	[18, 25]	2	2
60	[22, 30]	4	4
Approximately 15% rejected jobs			
20	[15, 20]	<1	<1
30	[22, 30]	1	1
40	[30, 40]	3	5
50	[37, 50]	5	7
60	[44, 60]	9	12

Table 2 Average and worst-case running times of the ***DP*2** algorithm for Problem ***P*2**, with either 10%, 15%, or 20% rejected jobs (top, middle, and bottom tables, respectively)

n	R	Average running time (s)	Worst-case running time (s)
Approximately 10 rejected jobs			
50	[12, 25]	0.010	0.017
75	[18, 37]	0.031	0.044
100	[25, 50]	0.069	0.089
125	[31, 62]	0.142	0.186
150	[37, 75]	0.220	0.290
Approximately 15% rejected jobs			
50	[37, 50]	0.017	0.021
75	[56, 75]	0.059	0.079
100	[75, 100]	0.134	0.157
125	[93, 125]	0.267	0.306
150	[112, 150]	0.435	0.525
Approximately 20% rejected jobs			
50	[62, 75]	0.025	0.028
75	[93, 112]	0.084	0.094
100	[125, 150]	0.193	0.236
125	[156, 187]	0.392	0.438
150	[187, 225]	0.979	2.915

problems. The worst-case running time for problems of 60 jobs and 15% rejected jobs did not exceed 12 ms.

*DP*2.1: Instances with $n = 50,\ 75,\ 100,\ 125$, and 150 jobs were generated. The cost parameters are *common* to all jobs and the *common* lead time was generated uniformly in the interval $[0,\ 0.25]n\bar{p}$. The upper bound on the total rejection cost was generated uniformly in the intervals $[0.005,\ 0.010]n\bar{r}$, $[0.015,\ 0.020]n\bar{r}$, and $[0.025,\ 0.030]n\bar{r}$, reflecting approximately 10%, 15%, and 20% of rejected jobs. Table 2 presents the average and worst-case running times of Algorithm *DP*2.1 for solving 20 randomly generated instances of n and R. Note that the worst-case running time for problems of 150 jobs and 20% rejected jobs did not exceed 3 s, demonstrating that, similar to *DP*1, *DP*2.1 is exceptionally efficient in solving medium-size problems.

*DP*3: Instances were generated with $n = 10,\ 20,\ 30$, and 40 jobs. The cost parameters are *job dependent*. A tightness factor $\varepsilon = 0.20,\ 0.30,\ 0.40$ was used to generate the *common* lead time for each instance, as a factor of the total processing times, using $l = \varepsilon P$. The total rejection cost R was generated uniformly in the interval $[0.02,\ 0.04]n\bar{r}$, replicating approximately 20% of rejected jobs. For each set of n and l, 20 cases were constructed and solved. Table 3 presents the average and worst-case running times (in seconds) for $\varepsilon = 0.20,\ 0.30$, and 0.40. The results demonstrate that *DP*3 is efficient and can solve medium-size problems. In particular, the worst-case running times for problems with $n = 40$, $\varepsilon = 0.40$, and 20% of rejected jobs is less than 4.1 s.

Table 3 Average and worst-case running times of the ***DP*3** algorithm for Problem ***P*3**, with a tightness factor (ε) of either 0.2, 0.3, or 0.4 (top, middle, and bottom tables, respectively)

n	R	Average running time (s)	Worst-case running time (s)
$\varepsilon = 0.20$			
10	[10, 20]	0.001	0.003
20	[20, 40]	0.034	0.055
30	[30, 60]	0.325	0.474
40	[40, 80]	1.301	1.914
$\varepsilon = 0.30$			
10	[10, 20]	0.001	0.002
20	[20, 40]	0.052	0.079
30	[30, 60]	0.469	0.716
40	[40, 80]	1.940	3.005
$\varepsilon = 0.40$			
10	[10, 20]	0.001	0.004
20	[20, 40]	0.076	0.123
30	[30, 60]	0.590	1.074
40	[40, 80]	2.738	4.014

9 Conclusions

We studied a single-machine scheduling and due-date assignment problem, known as DIF, with the extension of optional job rejection that possibly bounds the constraints or the underlying cost functions. As DIF with job-dependent cost parameters and lead times is strongly NP-hard, we focused on several restricted versions of the problem, in which either the cost parameters or the lead times are job dependent. We established that all studied problems are NP-hard in the ordinary sense and introduced efficient pseudo-polynomial dynamic programming algorithms that are suitable for solving real-life, medium-sized instances. Solutions to larger instances may perhaps be addressed by FPTAS, metaheuristics, or machine learning. A challenging problem for future research is to provide a heuristic or branch and bound solution for the general problem of job-dependent cost parameters and lead times with an optional job-rejection, and to extend the setting to a more complex machine such as flowshops and parallel machines.

Funding The authors did not receive support from any organization for the submitted work.

Data availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Conflict of interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Cao, Z.G., Wang, Z., Zhang, Y.Z., Liu, S.P.: On several scheduling problems with rejection or discretely compressible processing times. Lect. Notes Comput. Sci. **3959**, 90–98 (2006)
2. Du, J., Leung, J.Y.T.: Minimizing total tardiness on one machine is NP-hard. Math. Oper. Res. **15**, 483–495 (1990)
3. Gerstl, E., Mor, B., Mosheiov, G.: Minmax scheduling with acceptable lead-times: extensions to position-dependent processing times, due-window and job rejection. Comput. Oper. Res. **83**, 150–156 (2017)
4. Gerstl, E., Mosheiov, G.: Single machine scheduling problems with generalised due-dates and job-rejection. Int. J. Prod. Res. **55**, 3164–3172 (2017)
5. Gerstl, E., Mosheiov, G.: Minmax due-date assignment with a time window for acceptable lead-times. Ann. Oper. Res. **211**(1), 167–177 (2013)
6. Gerstl, E., Mosheiov, G.: Scheduling with a due-window for acceptable lead-times. J. Oper. Res. Soc. **66**(9), 1578–1588 (2015)
7. Kacem, I.: Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date. Discret. Appl. Math. **158**, 1035–1040 (2010)
8. Kellerer, H., Strusevich, V.A.: A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date. Theoret. Comput. Sci. **369**, 230–238 (2006)
9. Kianfar, K., Moslehi, G.: A note on "Fully polynomial time approximation scheme for the total weighted tardiness minimization with a common due date." Discrete Appl. Math. **161**, 2205–2206 (2013)

10. Lawler, E.L.: A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. Ann. Discrete Math. **1**, 331–342 (1977)
11. Leyvand, Y., Shabtay, D., Steiner, G.: A unified approach for scheduling with convex resource consumption functions using positional penalties. Eur. J. Oper. Res. **206**, 301–312 (2010)
12. Mor, B., Mosheiov, G., Shabtay, D.: A note: minmax due-date assignment problem with lead-time cost. Comput. Oper. Res. **40**, 2161–2164 (2013)
13. Mor, B., Shapira, D.: Scheduling with regular performance measures and optional job rejection on a single machine. J. Oper. Res. Soc. **71**(8), 1315–1325 (2020)
14. Mosheiov, G., Pruwer, S.: On the minmax common-due-date problem: extensions to position-dependent processing times, job rejection, learning effect, uniform machines and flowshops. Eng. Optim. **53**(3), 408–424 (2021)
15. Seidmann, A., Panwalkar, S.S., Smith, M.L.: Optimal assignment of due-dates for a single processor scheduling problem. Int. J. Prod. Res. **19**, 393–399 (1981)
16. Shabtay, D.: Scheduling and due date assignment to minimize earliness, tardiness, holding, due date assignment and batch delivery costs. Int. J. Prod. Econ. **123**, 235–242 (2010)
17. Shabtay, D., Gaspar, N., Kaspi, M.: A survey on offline scheduling with rejection. J. Sched. **16**, 3–28 (2013)
18. Shabtay, D., Steiner, G.: Two due-date assignment problems in scheduling a single machine. Oper. Res. Lett. **34**, 683–691 (2006)
19. Shabtay, D., Steiner, G.: The single-machine earliness-tardiness scheduling problem with due-date assignment and resource-dependent processing times. Ann. Oper. Res. **159**, 25–40 (2008)
20. Shabtay, D., Steiner, G.: Optimal due date assignment in multi-machine scheduling environments. J. Sched. **11**, 217–228 (2008)
21. Steiner, G., Zhang, R.: Revised delivery-time quotation in scheduling with tardiness penalties. Oper. Res. **59**, 1504–1511 (2011)
22. Yuan, J.: The NP-hardness of the single machine common due date weighted tardiness problem. Tabriz Univ. Ser. **5**, 328–333 (1992)
23. Zhang, L., Lu, L., Yuan, J.: Single-machine scheduling under the job rejection constraint. Theoret. Comput. Sci. **411**, 1877–1882 (2010)
24. Zhao, C., Yin, Y., Cheng, T.C.E., Wu, C.C.: Single-machine scheduling and due date assignment with rejection and position-dependent processing times. J. Indus. Manag. Optim. **10**(3), 691–700 (2014)