Single-machine scheduling with deadlines to minimize the total weighted late work

Rubing Chen[1], Jinjiang Yuan[1*], C.T. Ng[2], T.C.E. Cheng[2]

[1]School of Mathematics and Statistics, Zhengzhou University,

Zhengzhou, Henan 450001, People's Republic of China

[2]Logistics Research Centre, Department of Logistics and Maritime Studies,

The Hong Kong Polytechnic University, Hong Kong, People's Republic of China

Abstract: We consider scheduling a set of jobs with deadlines to minimize the total weighted late work on a single machine, where the late work of a job is the amount of processing of the job that is scheduled after its due date and before its deadline. This is the first study on scheduling with the late work criterion under the deadline restriction. In this paper, we show that (i) the problem is unary NP-hard even if all the jobs have a unit weight, (ii) the problem is binary NP-hard and admits a pseudo-polynomial-time algorithm and a fully polynomial-time approximation scheme if all the jobs have a common due date, and (iii) some special cases of the problem are polynomially solvable.

Key words: due dates; deadlines; late work; NP-hardness; approximation algorithm

1 Introduction

Background: The scheduling problem with the late work criterion under the deadline restriction may arise in agricultural production. For instance, there are different fields that need to be harvested by a single harvester. Each crop can be regarded as a job and the total amount of each crop planted is regarded as its processing time. Due to different requirements from different buyers, any part of the crop not gathered by a given date, which depends on the type, can no longer be sold. This date can be regarded as the due

*Corresponding author. Email address: yuanjj@zzu.edu.cn

date for a kind of crop. Moreover, each kind of crop has its own final harvest date, which can be regarded as its deadline. Minimizing the total late work in this example corresponds to minimizing the quantity of the crop that has not been sold but has been harvested before its final harvest date. The example described above shows that scheduling in the setting of the late work criterion under the deadline restriction has applications in practice and is worth studying.

Motivated by applications in the field of information processing, Blazewicz (1984) introduced the scheduling model to minimize the total weighted late work. Other applications of the model can be found in manufacturing (Sterna, 2006; 2007) and agriculture (Alminana et al., 2010). While there has been plentiful research on scheduling to minimize the total weighted late work in the literature, no work has considered the case where the jobs have deadlines. The constraint of job deadlines is important for scheduling research and practice. Applications of job deadlines can be found in energy efficient packet transmission (Zafer and Modiano, 2009; Shan Feng et al., 2014), workflow scheduling (Abrishami and Naghibzadeh, 2012; Abrishami et al., 2013), and integrated production and distribution scheduling of perishable products (Moons et al., 2017). To the best of our knowledge, the scheduling problem with the late work criterion under the deadline restriction has not been discussed in the literature.

Problem Formulation: Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of jobs to be processed on a single machine. Each job J_j has an integer processing time $p_j > 0$, a weight $w_j \geq 0$, a due date $d_j \geq 0$, and a deadline $\bar{d}_j \geq \max\{p_j, d_j\}$. A *feasible schedule* is a schedule that processes the jobs of $\mathcal{J}$ on the single machine subject to deadline constraints. Given a feasible schedule σ, $J_{\sigma(j)}$ denotes the j-th job in σ. Let $C_j(\sigma)$ be the completion time of job J_j in σ. Then the deadline constraints require that $C_j(\sigma) \leq \bar{d}_j$. The late work of job J_j in schedule σ, denoted by $Y_j(\sigma)$, is the amount of processing of job J_j that is scheduled after its due date d_j in σ. Since preemption is not allowed, we have

$$Y_j(\sigma) = \begin{cases} 0, & \text{if } C_j(\sigma) \leq d_j, \\ C_j(\sigma) - d_j, & \text{if } d_j < C_j(\sigma) \leq \min\{d_j + p_j, \bar{d}_j\}, \\ p_j, & \text{if } d_j + p_j < C_j(\sigma) \leq \bar{d}_j. \end{cases}$$

If $Y_j(\sigma) = 0$, job J_j is called *early*. If $0 < Y_j(\sigma) < p_j$, job J_j is called *partially early*. If $Y_j(\sigma) = p_j$, job J_j is called *late*. The objective is to find a feasible schedule σ such that the total weighted late work, denoted by $\sum_{j=1}^{n} w_j Y_j(\sigma)$, is minimized. Using the three-field notation for describing scheduling problems introduced by Graham et al. (1979), we denote the scheduling problem under study by $1|\bar{d}_j|\sum w_j Y_j$. Clearly, we only need to consider the schedules in which the n jobs are processed consecutively from time 0 to

$\sum_{j=1}^{n} p_j$ without idle times.

Feasibility Checking: Under the deadline constraints, not every instance is feasible. Clearly, an instance with a job set $\mathcal{J}$ is feasible (subject to the deadline constraints) if and only if the feasibility problem $1|\bar{d}_j|-$ on the instance with a job set $\mathcal{J}$ has a feasible schedule. Jackson (1955) showed that the EDD (earliest due date first) rule solves problem $1||L_{\max}$ optimally. Then the problem $1|\bar{d}_j|-$ on the instance with a job set $\mathcal{J}$ can be solved by the following way: Generate the schedule σ_0 in which the jobs in $\mathcal{J}$ are scheduled in the nondecreasing order of their deadlines, and then check whether σ_0 is feasible subject to the deadlines. Consequently, the feasibility checking process of problem $1|\bar{d}_j|-$ could be done in $O(n \log n)$ time.

For each scheduling problem studied in this paper, the $O(n \log n)$-time complexity used in the feasibility checking is dominated by the time complexity for solving this problem. Henceforth, we assume that a feasible schedule always exists when we design an algorithm for solving problem $1|\bar{d}_j| \sum w_j Y_j$.

Literature Review: For problem $1|| \sum Y_j$, Potts and Van Wassenhove (1992a) showed that it is binary NP-hard, and provided a pseudo-polynomial dynamic programming solution algorithm that runs in $O(n\text{UB})$ time, where UB is an upper bound on the minimum total late work. They also presented an $O(n \log n)$ time algorithm to solve problem $1|p_j = p| \sum Y_j$. Moreover, for problem $1|d_j = d| \sum Y_j$, since the value of the total late work equals $\max\{\sum p_j - d, 0\}$ in any schedule, any schedule is optimal and the problem is solved in $O(n)$ time. Potts and Van Wassenhove (1992b) presented $(1+\frac{1}{k})$-approximation algorithms ($1 \leq k \leq n$) with time and space requirements of $O(n^{(k+1)})$ and $O(n)$, respectively, and two fully polynomial approximation schemes for problem $1|| \sum Y_j$. Lin and Hsu (2005) provided a branch-and-bound algorithm for problem $1|r_j| \sum Y_j$, and presented $O(n \log n)$ time algorithms to solve problems $1|r_j, d_j = d| \sum Y_j$ and $1|r_j, \text{pmtn}| \sum Y_j$.

For problem $1|| \sum w_j Y_j$, its binary NP-hardness follows from Potts and Van Wassenhove (1992a). Kovalyov et al. (1994) provided a dynamic programming solution algorithm and applied the rounding technique to design a fully polynomial-time approximation scheme (FPTAS) for problem $1|| \sum w_j Y_j$. Hariri et al. (1995) presented a pseudo-polynomial dynamic programming solution algorithm for problem $1|| \sum w_j Y_j$ that runs in $O(n^2 \sum p_j)$ time. They also presented an $O(n)$ time algorithm to solve problem $1|d_j = d| \sum w_j Y_j$ and an $O(n^3)$ time algorithm to solve problem $1|p_j = p| \sum w_j Y_j$. Leung et al. (1994) provided an $O(n \log n + kn)$ time solution algorithm for problem $1|r_j, \text{pmtn}| \sum w_j Y_j$, where k is the number of distinct weights.

In the parallel-machine setting, Blazewicz and Finke (1987) reduced problems $P|r_j, \text{pmtn}| \sum w_j Y_j$

and $Q|r_j, \text{pmtn}|\sum w_jY_j$ to the minimum cost flow problems, respectively. The corresponding networks contain only two groups of vertices, corresponding to the jobs and time intervals. They provided an $O(n^7 \log n)$ time algorithm to solve problem $P|r_j, \text{pmtn}|\sum w_jY_j$ and an $O(m^3n^7 \log mn)$ time algorithm to solve problem $Q|r_j, \text{pmtn}|\sum w_jY_j$, where m is the number of machines. By using the same reduction in Orlin (1993) and a more efficient algorithm for solving the minimum cost flow problem, Leung (2004) showed that problems $P|r_j, \text{pmtn}|\sum w_jY_j$ and $Q|r_j, \text{pmtn}|\sum w_jY_j$ can be solved in $O(n^2 \log^3 n)$ time and $O(m^2n^4 \log mn)$ time, respectively, and problems $P|r_j, \text{pmtn}|\sum Y_j$ and $Q|r_j, \text{pmtn}|\sum Y_j$ can be solved in $O(n^2 \log^2 n)$ time and $O(m^2n^3 \log mn)$ time, respectively. Xu et al. (2015) first applied meta-heuristic algorithms to treat problem $P|d_j = d|\sum w_jY_j$, in which the jobs have a common due date. Sterna (2011) provided a comprehensive survey of research on the late work scheduling.

There is also extensive research on scheduling with the late work criterion in other machine environments. Kethley and Alidaee (2002) considered the single-machine scheduling to minimize the modified total weighted late work. The modified weighted late work of J_j is 0 if $C_j \leq d_j$, $w_j(C_j - d_j)$ if $d_j < C_j \leq \bar{d}_j$, and $w_j(\bar{d}_j - d_j)$ otherwise. They provided some heuristic algorithms and conducted computational experiments to assess their performance. Ren et al. (2009) showed that scheduling on an unbounded parallel batch machine to minimize the total late work is binary NP-hard. Ren et al. (2013) showed that scheduling in an assembling manufacturing system where several suppliers provide component parts to a manufacturer to minimize the total late work is unary NP-hard. Wu et al. (2016) studied single-machine scheduling to minimize the total late work with a position-based learning effect. They provided a branch-and-bound algorithm to solve the problem and three genetic algorithms to obtain near-optimal solutions. Chen et al. (2016) considered both offline and online versions of scheduling on parallel identical machines to minimize the total late work with a common due date. Wang et al. (2017) studied the two-agent problem $1||\sum Y_j^{(A)} : L_{\max}^{(B)} \leq U$. They presented two pseudo-polynomial dynamic programming solution algorithms for small-scale instances and a branch-and-bound algorithm for medium-to large-scale instances. Zhang and Wang (2017) and Zhang and Yuan (2019) studied the two-agent problem $1||\sum w_jY_j^{(A)} : f_{\max}^{(B)} \leq U$ and presented pseudo-polynomial-time solution algorithms. Zhang and Yuan (2019) also showed that problem $1|d_j^{(A)} = d^{(A)}|\sum Y_j^{(A)} : C_{\max}^{(B)} \leq U$ is binary NP-hard and presented an $O(n\sum p_j^{(A)})$ time solution algorithm.

Scheduling with job deadlines is also an active topic in scheduling research. Problem $1|\bar{d}_j|\sum C_j$ is solvable in $O(n \log n)$ time by Smith's deadline rule in Smith (1956). Lenstra et al. (1977) showed that problem $1|\bar{d}_j|\sum w_jC_j$ is unary NP-hard. A number of branch and bound algorithms can be found in the literature for this problem, such as Potts and

Van Wassenhove (1983), Posner (1985), Wenrner (1993) and Pan (2003). For problem $1|\bar{d}_j|T_{\max}$, Koulamas and Kyparisis (2001) presented an $O(n \log n)$ time algorithm. They also showed that problem $1|(d_j, \bar{d}_j)|\sum T_j$ is solvable in $O(n^4 \sum p_j)$ or $O(n^5 \max p_j)$ time, where "$(d_j, \bar{d}_j)$" means that if $d_i \leq d_j$ then $\bar{d}_i \leq \bar{d}_j$. Yuan (2017) showed that problem $1|\bar{d}_j|\sum U_j$ is unary NP-harness. Recently, Chen and Yuan (2018a) showed that problem $1|\bar{d}_j|\sum T_j$ is unary NP-hard. When the release times and preemption are taken into consideration, Du and Leung (1993) and Wan et al. (2015) showed that problem $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$ is NP-hard. He et al. (2014) presented an $O(n^2)$ time algorithm for a special case of problem $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$, where release times and processing times are agreeable. Recently, Chen and Yuan (2018b) showed that problem $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$ is unary NP-hard.

For problem $P|r_j, \bar{d}_j, \text{pmtn}|-$, Horn (1974) showed that this feasibility problem can be converted to a network flow problem which can be solved in $O(n^3)$ time. Labetoulle et al. (1984) showed that problem $P|r_j, \bar{d}_j, \text{pmtn}|L_{\max}$ is solvable in $O(n^3 \min\{n^2, \log n + \log p_{\max}\})$ time, where $p_{\max}$ is the maximum processing time. For problem $Q|r_j, \bar{d}_j, \text{pmtn}|-$, Federgruen and Groenevelt (1986) showed that this feasibility problem can be solvable as a maximum flow problem in $O(tn^3)$ time, where t is the number of different machine speeds. They also showed that problem $Q|r_j, \bar{d}_j, \text{pmtn}|L_{\max}$ is solvable in $O(tn^3(\log n + \log p_{\max} + \log s_1))$ time, where $p_{\max}$ is the maximum processing time, t is the number of different machine speeds, and s_1 is the largest machine speed. Mokotoff (2001) provided a comprehensive survey of research on the parallel-machine scheduling.

Our Contributions: The main contributions of this paper are as follows:

In Section 2 we show that problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$ is binary NP-hard. Then we develop a dynamic programming algorithm that runs in $O(n^2 d)$ time and an FPTAS that runs $O(n^5/\epsilon)$ time for problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$.

In Section 3 we show that problem $1|\bar{d}_j|\sum Y_j$ is unary NP-hard. Our proof imitates the unary NP-harness proof for problem $1|\bar{d}_j|\sum U_j$ in Yuan (2017).

In Section 4 we consider two special cases of problem $1|\bar{d}_j|\sum w_j Y_j$ in which the jobs either have a common due date and a unit weight or a common processing time, and show that the two problems are solvable in $O(n \log n)$ time and $O(n^3)$ time, respectively.

Finally, in Section 5 we conclude the paper and suggest topics for further research.

2 Problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$

In this section we first show that problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ is binary NP-hard. Then we present an $O(n^2 d)$ time algorithm and an FPTAS that runs in $O(n^5/\epsilon)$ time for this problem. Since there is no existing method for designing approximation algorithms with good performance for NP-hard scheduling problems with deadlines in the literature, our FPTAS may provide a basic foundation for future research in this direction.

2.1 NP-hardness proof

We show the binary NP-hardness of problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ by a reduction from the binary NP-complete Partition problem (Garey and Johnson (1979)).

Partition: Given a set of $t+1$ positive integers $x_1, x_2, \ldots, x_t, X$ such that $\sum_{j=1}^{t} x_j = 2X$, does there exist a partition (I_1, I_2) of the index set $\{1, 2, \ldots, t\}$ such that $\sum_{j \in I_1} x_j = \sum_{j \in I_2} x_j = X$?

Theorem 2.1. *Problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ is binary NP-hard.*

Proof. For a given instance $(x_1, x_2, \ldots, x_t, X)$ of Partition, we define a scheduling instance of problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ in the following way.

The scheduling instance has a total of $n = t+1$ jobs of two types: a *restriction job* J_0 and t *normal jobs* J_j, $j \in \{1, 2, \ldots, t\}$.

The processing times, weights, due dates, and deadlines of the n jobs are displayed in Table 1.

Table 1: The scheduling instance

Job	Processing time	Weight	Due date	Deadline
J_0	$p_0 = X$	$w_0 = 1$	$d_0 = X$	$\bar{d}_0 = 2X$
J_j	$p_j = x_j$	$w_j = X + 1$	$d_j = X$	$\bar{d}_j = 3X$

Let the threshold value for $\sum w_j Y_j$ be $Q = X^2 + 2X$. The decision asks if there is a feasible schedule π for the constructed scheduling instance such that $\sum w_j Y_j(\pi) \leq Q$.

Clearly, the above construction can be performed in polynomial time. We show in the following that the instance $(x_1, x_2, \ldots, x_t, X)$ is a YES instance of Partition if and only if there is a feasible schedule π for the constructed instance of problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ such that $\sum w_j Y_j(\pi) \leq Q$.

Assume that $(x_1, x_2, \ldots, x_t, X)$ is a YES instance of Partition. Then $\{1, 2, \ldots, t\}$ can be partitioned into two subsets I_1 and I_2 such that $\sum_{j\in I_i} x_j = X$ for each $i \in \{1, 2\}$. Set $\mathcal{J}_1 = \{J_j : j \in I_1\}$ and $\mathcal{J}_2 = \{J_j : j \in I_2\}$. Then we construct a schedule π of the $n = t + 1$ jobs such that they are scheduled in the order: $\mathcal{J}_1 \prec J_0 \prec \mathcal{J}_2$.

Note that $\sum_{J_j\in\mathcal{J}_1} p_j = \sum_{j\in I_1} x_j = X$ and $\sum_{J_j\in\mathcal{J}_2} p_j = \sum_{j\in I_2} x_j = X$. From the structure of π, the maximum completion time of the jobs of $\mathcal{J}_1$ is $\sum_{J_j\in\mathcal{J}_1} p_j = X$, the completion time of J_0 is $\sum_{J_j\in\mathcal{J}_1} p_j + p_0 = 2X$, and the maximum completion time of the jobs of $\mathcal{J}_2$ is $\sum_{J_j\in\mathcal{J}_1} p_j + p_0 + \sum_{J_j\in\mathcal{J}_2} p_j = 3X$. From the definitions of deadlines, all the jobs meet their deadlines in π, so π is a feasible schedule. Note that all the jobs have a common due date X. From the definition of late work, $Y_j(\pi) = 0$ for $J_j \in \mathcal{J}_1$, $Y_0(\pi) = p_0 = X$, and $Y_j(\pi) = p_j = x_j$ for $J_j \in \mathcal{J}_2$. Then we have $\sum w_j Y_j(\pi) = w_0 Y_0(\pi) + \sum_{J_j\in\mathcal{J}_2} w_j Y_j(\pi) = X + (X + 1)\sum_{j\in I_2} x_j = X + (X + 1)X = X^2 + 2X = Q$. It follows that π is a feasible schedule for the constructed instance of problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$ such that $\sum w_j Y_j(\pi) \le Q$.

Conversely, assume that π is a feasible schedule for the constructed instance of problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$ such that $\sum w_j Y_j(\pi) \le Q$. Then we have the following statement.

Statement 1. *The completion time of J_0 in π is $C_0(\pi) = 2X$.*

Proof of Statement 1. From the feasibility of π and $\bar{d}_0 = 2X$, we have $C_0(\pi) \le \bar{d}_0 = 2X$.

Suppose to the contrary that $C_0(\pi) \le 2X - 1$. Since $\sum_{j=0}^{t} p_j = p_0 + \sum_{j=1}^{t} x_j = 3X$, the length of the normal jobs scheduled after J_0 is no less than $X + 1$. Note that $p_0 = X$ and all the jobs have a common due date $X = p_0 \le C_0(\pi)$. Then the normal jobs scheduled after J_0 are all late. Since $w_j = X + 1$ for each normal job J_j, we have $\sum w_j Y_j(\pi) \ge \sum_{j=1}^{t} w_j Y_j(\pi) \ge (X + 1)(X + 1) = X^2 + 2X + 1 > Q$, a contradiction. This proves Statement 1.

Since $p_0 = X$ and $\bar{d}_j = 3X$ for each normal job J_j, from Statement 1, the t normal jobs are exactly scheduled in the time intervals $[0, X]$ and $[2X, 3X]$, respectively. Set $I_1 = \{j : 1 \le j \le t, C_j(\pi) < C_0(\pi)\}$ and $I_2 = \{j : 1 \le j \le t, C_j(\pi) > C_0(\pi)\}$. Then we have $\sum_{j\in I_1} x_j = \sum_{j\in I_1} p_j \le X$, $\sum_{j\in I_2} x_j = \sum_{j\in I_2} p_j \le X$, and (I_1, I_2) forms a partition of $\{1, 2, \ldots, t\}$. From the fact that $\sum_{j\in I_1} x_j + \sum_{j\in I_2} x_j = 2X$, we have $\sum_{j\in I_1} x_j = \sum_{j\in I_2} x_j = X$. It follows that (I_1, I_2) is a YES instance of Partition. The result follows. □

2.2 A dynamic programming algorithm

From Theorem 2.1, problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$ is binary NP-hard. We now present a pseudo-polynomial algorithm to solve this problem. Then both results allow us to classify

problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ as NP-hard in the ordinary sense.

Our idea for designing the pseudo-polynomial algorithm is similar to the methods proposed for some scheduling problems in the literature such as problem $1||\sum Y_j$ in Potts and Van Wassenhove (1992a) and problem $F2|d_j = d|\sum w_jY_j$ in Blazewicz et al. (2005). It should be noticed that, in these methods, the critical job or the completion time of the critical job is enumerated, and the best solution is chosen from all the feasible solutions.

Let $TP = \sum_{j=1}^{n} p_j$. If $TP \leq d$, we can schedule all the jobs prior to d such that all the jobs are early. Thus, we assume that $TP > d$.

In a feasible schedule σ, the unique job J_j with $S_j(\sigma) \leq d < S_j(\sigma) + p_j \leq \bar{d}_j$ is called a *critical job*, where $S_j(\sigma)$ is the starting time of J_j in σ. Clearly, there is only one critical job in any feasible schedule and if J_j is a critical job, then we have $d - p_j < S_j(\sigma) \leq \min\{d, \bar{d}_j - p_j\}$.

Next, we define a restricted problem of problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$, where the critical job $J^{(c)}$ is determined.

Problem $\mathcal{P}(J^{(c)})$: For a job $J^{(c)} \in \mathcal{J}$, we define $\mathcal{P}(J^{(c)})$ as a restricted problem of problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ to process the jobs of $\mathcal{J}$, subject to the deadlines on the single machine without idle times, such that $J^{(c)}$ is the critical job.

Note that for each feasible schedule for problem $\mathcal{P}(J^{(c)})$, the jobs scheduled before the critical job $J^{(c)}$ are all early and the jobs scheduled after the critical job $J^{(c)}$ are all late. Obviously, the early jobs scheduled before the critical job $J^{(c)}$ can be scheduled in an arbitrary order, but the late jobs scheduled after the critical job $J^{(c)}$ must satisfy their deadlines. Thus, to solve problem $\mathcal{P}(J^{(c)})$, we re-number the jobs of $\mathcal{J} \setminus \{J^{(c)}\}$ such that $\mathcal{J} \setminus \{J^{(c)}\} = \{J_1, J_2, \ldots, J_{n-1}\}$ and $\bar{d}_1 \leq \bar{d}_2 \leq \cdots \leq \bar{d}_{n-1}$. Moreover, for a job J_j and a time point τ, we use $Y_j(\tau)$ to denote the late work of job J_j when it starts at time τ and $\tau + p_j \leq \bar{d}_j$. Then we have

$$Y_j(\tau) = \begin{cases} 0, & \text{if } \tau + p_j \leq d, \\ \tau + p_j - d, & \text{if } \tau \leq d < \tau + p_j, \\ p_j, & \text{if } \tau > d. \end{cases} \tag{1}$$

By the job-shifting argument, we can obtain the following lemma.

Lemma 2.1. *For problem $\mathcal{P}(J^{(c)})$, there exists an optimal schedule possessing the following properties:*

(i) the jobs scheduled before $J^{(c)}$ are processed consecutively with no idle time from time

0 in an arbitrary order;

(ii) the jobs scheduled after $J^{(c)}$ are scheduled in their index order.

In an optimal schedule π^* for problem $\mathcal{P}(J^{(c)})$ possessing the properties in Lemma 2.1, the starting time $S^{(c)}(\pi^*)$ of job $J^{(c)}$ in π^* must satisfy $S^{(c)}(\pi^*) \leq d < S^{(c)}(\pi^*)+p^{(c)} \leq \bar{d}^{(c)}$. Set $\tau \in \{0,1,\ldots,d\}$, $j \in \{1,2,\ldots,n-1\}$ and $\tau'_j = TP-(\sum_{k=j}^{n-1} p_k - \tau)$. Based on Lemma 2.1, we provide a dynamic programming algorithm as follows.

We define $f_j^{(c)}(\tau)$ to be minimum value of the total weighted processing times of the jobs scheduled in the interval $[\tau'_j, TP]$ in a schedule for jobs $J_j, J_{j+1}, \ldots, J_{n-1}$, which has the following properties:
(i) each job in $\{J_j, J_{j+1}, \ldots, J_{n-1}\}$ is either scheduled in the interval $[0,\tau]$ or scheduled in the interval $[\tau'_j, TP]$;
(ii) the jobs in the interval $[0,\tau]$ are scheduled consecutively in an arbitrary order;
(iii) the jobs in the interval $[\tau'_j, TP]$ are scheduled consecutively in their index order, subject to their deadlines.

Note that job J_j is scheduled in either the interval $[0,\tau]$ or the interval $[\tau'_j, TP]$ as the first job. In the former case, we have $f_j^{(c)}(\tau) = f_{j+1}^{(c)}(\tau - p_j)$; in the latter case, we have $f_j^{(c)}(\tau) = f_{j+1}^{(c)}(\tau) + w_j p_j$, provided that $\tau'_j + p_j \leq \bar{d}_j$. To guarantee that the recursion runs correctly, we introduce a dummy job J_n with $p_n = w_n = 0$, $d_n = d$, and $\bar{d} = +\infty$, and force it to complete at time 0. Then we have $f_n^{(c)}(0) = 0$ and $f_n^{(c)}(\tau) = +\infty$ for $\tau \neq 0$. Consequently, we have the following dynamic programming (DP) for calculating $f_j^{(c)}(\tau)$.

Algorithm DP: For calculating $f_j^{(c)}(\tau)$, $j \in \{1,2,\ldots,n-1\}$ and $\tau \in \{0,1,\ldots,d\}$.

The initial condition: $f_n^{(c)}(0) = 0$ and $f_n^{(c)}(\tau) = +\infty$ for $\tau \neq 0$.

The recursive function: For $j \in \{1,2,\ldots,n-1\}$ and $\tau \in \{0,1,\ldots,d\}$, we have

$$f_j^{(c)}(\tau) = \min\{f_{j+1}^{(c)}(\tau - p_j),\ f_{j+1}^{(c)}(\tau) + w_j p_j + \delta(j,\tau'_j)\},$$

where $\delta(j,\tau'_j) = 0$ if $\tau'_j + p_j \leq \bar{d}_j$, and $\delta(j,\tau'_j) = +\infty$ if $\tau'_j + p_j > \bar{d}_j$.

Algorithm DP runs in $O(nd)$ time and yields all the values $f_1^{(c)}(\tau)$ for $\tau \in \{0,1,\ldots,d\}$. Note that $J^{(c)}$ is the unique critical job in every feasible schedule for problem $\mathcal{P}(J^{(c)})$. If τ is the starting time of $J^{(c)}$, then $\tau \leq d < \tau + p^{(c)} \leq \bar{d}^{(c)}$, implying that the late work of $J^{(c)}$ is $Y^{(c)}(\tau) = \tau + p^{(c)} - d$. By Lemma 2.1, we derive the following result.

Lemma 2.2. *The optimal value of problem $\mathcal{P}(J^{(c)})$ is given by*

$$\min\{w^{(c)}Y^{(c)}(\tau) + f_1^{(c)}(\tau) : \tau \leq d < \tau + p^{(c)} \leq \bar{d}^{(c)}\}.$$

The value τ in Lemma 2.2 has at most $O(d)$ choices, which is dominated by Algorithm DP. Thus, problem $\mathcal{P}(J^{(c)})$ is solvable in $O(nd)$ time. Since problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ can be solved by enumerating $J^{(c)} \in \mathcal{J}$, solving problem $\mathcal{P}(J^{(c)})$ for each $J^{(c)}$ and picking the best solution, we obtain the following result.

Theorem 2.2. *Problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ is solvable in $O(n^2d)$ time.*

To better understand our pseudo-polynomial algorithm, we provide the whole process for solving problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ in the following algorithm.

Algorithm DP′: For problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$.

Step 1: For each job $J^{(c)} \in \mathcal{J}$, do the following:

– Define $J^{(c)}$ to be the critical job and create an instance of problem $\mathcal{P}(J^{(c)})$.

– Invoke Algorithm DP for problem $\mathcal{P}(J^{(c)})$ and obtain the values $\{f_1^{(c)}(\tau) : \tau \in \{0, 1, \ldots, d\}\}$.

– Compute $WY^{(c)} := \min\{w^{(c)}Y^{(c)}(\tau) + f_1^{(c)}(\tau) : \tau \le d < \tau + p^{(c)} \le \bar{d}^{(c)}\}$, which is the optimal value of problem $\mathcal{P}(J^{(c)})$.

Step 2: Enumerate all the jobs $J^{(c)} \in \mathcal{J}$ and compute the value $\min_{J^{(c)}\in\mathcal{J}} WY^{(c)}$, which is the optimal value of problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$.

2.3 An FPTAS for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$

Recall that $TP = \sum_{J_j\in\mathcal{J}} p_j$. If $TP \le d$, then every schedule is optimal. Thus, we assume in the sequel that $TP > d$, which guarantees that $\sum_{j=1}^{n} w_jY_j(\sigma) > 0$ for every feasible schedule σ.

In Section 2.2 we used the auxiliary problem $\mathcal{P}(J^{(c)})$ to obtain a pseudo-polynomial-time algorithm for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$. However, we have no idea on designing an FPTAS for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$ based on Algorithm DP directly. In fact, in problem $\mathcal{P}(J^{(c)})$, the restriction $S^{(c)}(\sigma) \le d < S^{(c)}(\sigma) + p^{(c)} \le \bar{d}^{(c)}$ for a feasible schedule σ hinders the design of an FPTAS for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$. So we adopt a roundabout approach in this section to design an FPTAS for the problem. In the following, we first consider an artificial scheduling problem, denoted as $\mathcal{P}(J^{(c)}, J^{(e)})$, for each pair of jobs $(J^{(c)}, J^{(e)}) \in \mathcal{J}$, in which the above restriction for $J^{(c)}$ in a feasible schedule σ is relaxed to $S^{(c)}(\sigma) \le d$ and another job $J^{(e)}$ is introduced to restrict the jobs scheduled after $J^{(c)}$.

Problem $\mathcal{P}(J^{(c)}, J^{(e)})$: For a pair of jobs $(J^{(c)}, J^{(e)}) \in \mathcal{J}$, we define $\mathcal{P}(J^{(c)}, J^{(e)})$ as the

scheduling problem to process the jobs of $\mathcal{J}$, subject to the deadlines on the single machine without idle times, such that

(i) in a feasible schedule, $J^{(c)}$ must start no later than time d,

(ii) in a feasible schedule, $J^{(e)}$ must be scheduled after $J^{(c)}$ and $w^{(e)}p^{(e)}$ is the maximum value of w_jp_j among all the jobs J_j scheduled after $J^{(c)}$,

(iii) the objective value of a feasible schedule σ is $F^{(c,e)}(\sigma) = \sum_{j=1}^{n} Z_j(\sigma)$, where

– $Z^{(c)}(\sigma) = w^{(c)}Y^{(c)}(\sigma)$,

– $Z_j(\sigma) = 0$ if J_j is scheduled before $J^{(c)}$ in σ, and

– $Z_j(\sigma) = w_jp_j$ if J_j is scheduled after $J^{(c)}$ in σ, and

(iv) the goal of problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is to find a feasible schedule σ such that $F^{(c,e)}(\sigma) = \sum_{j=1}^{n} Z_j(\sigma)$ is as small as possible. We use $F^*(c, e)$ to denote the optimal value of problem $\mathcal{P}(J^{(c)}, J^{(e)})$. In the case of infeasibility, we define $F^*(c, e) = +\infty$.

We next present a useful lemma.

Lemma 2.3. *Let σ be a feasible schedule for problem $\mathcal{P}(J^{(c)}, J^{(e)})$. Then σ is also a feasible schedule for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$. Moreover, we have*

$$w^{(e)}p^{(e)} \leq \sum_{j:J_j\in\mathcal{J}\setminus\{J^{(c)}\}} Z_j(\sigma) \leq (n-1)w^{(e)}p^{(e)} \tag{2}$$

and

$$\sum_{j=1}^{n} w_jY_j(\sigma) \leq \sum_{j=1}^{n} Z_j(\sigma). \tag{3}$$

Proof. Since σ is a feasible schedule for problem $\mathcal{P}(J^{(c)}, J^{(e)})$, each job meets its deadline in σ. Hence, σ is also a feasible schedule for problem $1|d_j = d, \bar{d}_j|\sum w_jY_j$.

The relations (2) follow from the fact that $w^{(e)}p^{(e)} = \max\{Z_j(\sigma) : J_j \in \mathcal{J} \setminus \{J^{(c)}\}\}$ and $J^{(e)} \in \mathcal{J} \setminus \{J^{(c)}\}$.

The relation (3) follows from the observation that $Z_j(\sigma) \geq w_jY_j(\sigma)$ for all $J_j \in \mathcal{J}$. □

To solve problem $\mathcal{P}(J^{(c)}, J^{(e)})$, we perform a preprocessing procedure for instance $\mathcal{J}$.

Preprocessing: Define

$$\mathcal{J}_1^{(c,e)} = \{J_i \in \mathcal{J} \setminus \{J^{(c)}\} : w_ip_i > w^{(e)}p^{(e)}\}$$

and

$$\mathcal{J}_2^{(c,e)} = \{J_i \in \mathcal{J} \setminus \{J^{(c)}\} : w_ip_i \leq w^{(e)}p^{(e)}\}.$$

Let $m = |\mathcal{J}_2^{(c,e)}|$. Re-number the jobs of $\mathcal{J}_2^{(c,e)}$ such that $\mathcal{J}_2^{(c,e)} = \{J_1, J_2, \ldots, J_m\}$ and $\bar{d}_1 \geq \bar{d}_2 \geq \cdots \geq \bar{d}_m$.

Sorting the jobs in nondecreasing order of their weighted processing times, we see that the two sets $\mathcal{J}_1^{(c,e)}$ and $\mathcal{J}_2^{(c,e)}$ can be obtained in $O(n \log n)$ time. Moreover, the sorted sequence $\bar{d}_1 \geq \bar{d}_2 \geq \cdots \geq \bar{d}_m$ can be obtained in $O(n \log n)$ time. Thus, the preprocessing procedure can be performed in $O(n \log n)$ time, which is dominated by the time complexity of the following algorithms.

Note that $J^{(e)} \in \mathcal{J}_2^{(c,e)}$, $\mathcal{J}_1^{(c,e)} \cap \mathcal{J}_2^{(c,e)} = \emptyset$, and $\mathcal{J} = \{J^{(c)}\} \cup \mathcal{J}_1^{(c,e)} \cup \mathcal{J}_2^{(c,e)}$. For any subset $\mathcal{J}' \subseteq \mathcal{J}$, let $p(\mathcal{J}')$ be the total processing time of the jobs of $\mathcal{J}'$. In any feasible schedule, the jobs of $\mathcal{J}_1^{(c,e)}$ must be processed before $J^{(c)}$ and $J^{(e)}$ must be processed after $J^{(c)}$. Thus, if $p(\mathcal{J}_1^{(c,e)}) > d$, then problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is infeasible. Suppose in the following that $p(\mathcal{J}_1^{(c,e)}) \leq d$. Without loss of generality, we only consider the schedules in which the jobs of $\mathcal{J}_1^{(c,e)}$ occupy the interval $[0, p(\mathcal{J}_1^{(c,e)})]$.

Sets of States: For each $j = 1, 2, \ldots, m$, we use $\Omega_j^{(c,e)}$ to denote the set of all the states $(\tau^{(c,e)}, f^{(c,e)})$ associated with the pair of jobs $(J^{(c)}, J^{(e)})$ such that each state $(\tau^{(c,e)}, f^{(c,e)}) \in \Omega_j^{(c,e)}$ corresponds to a schedule for the jobs of $\{J_1, J_2, \ldots, J_j\}$ with the following properties:

(i) the jobs of $\{J_1, J_2, \ldots, J_j\}$ occupy the two intervals $[p(\mathcal{J}_1^{(c,e)}), \tau^{(c,e)}]$ and $[\tau_j'^{(c,e)}, TP]$, where $p(\mathcal{J}_1^{(c,e)}) \leq \tau^{(c,e)} \leq d$ and $\tau_j'^{(c,e)} = TP - (\sum_{k=1}^{j} p_k + p(\mathcal{J}_1^{(c,e)}) - \tau^{(c,e)})$,

(ii) the jobs in the interval $[p(\mathcal{J}_1^{(c,e)}), \tau^{(c,e)}]$ are scheduled consecutively in an arbitrary order and the jobs in the interval $[\tau_j'^{(c,e)}, TP]$ are scheduled consecutively in their reversed index order, subject to their deadlines,

(iii) $J^{(e)}$ must be scheduled in the interval $[\tau_j'^{(c,e)}, TP]$ completely, and

(iv) $f^{(c,e)}$ is the sum of the values $w_i p_i$ of the jobs scheduled in the interval $[\tau_j'^{(c,e)}, TP]$.

Note that in a feasible schedule for scheduling the jobs $J_1, J_2, \ldots, J_j$, job J_j is scheduled either in the interval $[p(\mathcal{J}_1^{(c,e)}), \tau^{(c,e)}]$, subject to the restriction $J_j \neq J^{(e)}$, or in the interval $[\tau_j'^{(c,e)}, p(\mathcal{J})]$ as the first job, subject to the deadline $\bar{d}_j$. For a state $(\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)}) \in \Omega_{j-1}^{(c,e)}$, we have $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)} + p_j, f_{j-1}^{(c,e)})$ in the former case and $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)} + w_j p_j)$ in the latter case. If $J_j = J^{(e)}$, then only the latter happens. Note that if $\tau^{(c,e)}$ is the starting time of $J^{(c)}$, then the late work of $J^{(c)}$ is $Y^{(c)}(\tau^{(c,e)})$, as defined in (1). Next, we provide the following dynamic programming algorithm to solve problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Algorithm $DP(J^{(c)}, J^{(e)})$**:** For problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Step 1: Initially, set $\Omega_0^{(c,e)} = \{(p(\mathcal{J}_1^{(c,e)}), 0)\}$.

Step 2: For $j = 1, 2, \ldots, m$, do the following:

Generating $\Omega_j^{(c,e)}$: For every state $(\tau^{(c,e)}, f^{(c,e)}) \in \Omega_{j-1}^{(c,e)}$, do the following to generate the next state set $\Omega_j^{(c,e)}$:

(2.1) Put $(\tau^{(c,e)} + p_j, f^{(c,e)})$ in $\Omega_j^{(c,e)}$ if $\tau^{(c,e)} + p_j \leq d$ and $J_j \neq J^{(e)}$,

(2.2) Put $(\tau^{(c,e)}, f^{(c,e)} + w_j p_j)$ in $\Omega_j^{(c,e)}$ if $\tau_j'^{(c,e)} + p_j \leq \bar{d}_j$,

(2.3) If there are two states (τ, f') and (τ, f'') in $\Omega_j^{(c,e)}$ such that $f' \leq f''$, then delete the state (τ, f'') from $\Omega_j^{(c,e)}$. This procedure is repeated until no two states in $\Omega_j^{(c,e)}$ have a common τ value.

Note that if $\Omega_j^{(c,e)} = \emptyset$ after the procedure Generating $\Omega_j^{(c,e)}$ for some j, then problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is infeasible.

Step 3: Calculate the optimal value of problem $\mathcal{P}(J^{(c)}, J^{(e)})$ by

$$F^{(c,e)} = \min_{(\tau^{(c,e)}, f^{(c,e)}) \in \Omega_m^{(c,e)}} \{w^{(c)} Y^{(c)}(\tau^{(c,e)}) + f^{(c,e)} : \tau^{(c,e)} + p^{(c)} \leq \bar{d}^{(c)}\}.$$

Then generate the schedule $\sigma^{(c,e)}$ corresponding to $F^{(c,e)}$ by backtracking. In the case where $F^{(c,e)} = +\infty$, problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is infeasible and we just set $\sigma^{(c,e)} = \emptyset$.

Step 4: Output the optimal value $F^{(c,e)}$ and the optimal schedule $\sigma^{(c,e)}$.

Clearly, when $J^{(c)}$ and $J^{(e)}$ are given and problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is feasible, Algorithm $DP(J^{(c)}, J^{(e)})$ solves problem $\mathcal{P}(J^{(c)}, J^{(e)})$. Since each $\Omega_j^{(c,e)}$ has at most $O(d)$ states, Algorithm $DP(J^{(c)}, J^{(e)})$ runs in $O(nd)$ time. We need not consider this time complexity because Algorithm $DP(J^{(c)}, J^{(e)})$ is just an intermediate step for our FPTAS.

In the following we design an FPTAS for problem $\mathcal{P}(J^{(c)}, J^{(e)})$ based on Algorithm $DP(J^{(c)}, J^{(e)})$. We embed the feasibility checking in the algorithm.

Given an arbitrary constant $\epsilon > 0$, define $\nu := \lceil \frac{(n-1)^2}{\epsilon} \rceil$ and $\mu := \frac{(n-1)w^{(e)}p^{(e)}}{\nu}$. We partition the interval $[0, (n-1)w^{(e)}p^{(e)}]$ into ν subintervals

$$I_y = [(y-1)\mu, y\mu], \ y = 1, 2, \ldots, \nu, \tag{4}$$

of equal lengths μ. Then we present the following approximation algorithm for problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Algorithm $A_\epsilon^{(c,e)}$: For problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Step 1: Initially, set $\tilde{\Omega}_0^{(c,e)} = \{(p(\mathcal{J}_1^{(c,e)}), 0)\}$.

Step 2: For $j = 1, 2, \ldots, m$, do the following:

Generating $\tilde{\Omega}_j^{(c,e)}$: For every state $(\tau^{(c,e)}, f^{(c,e)}) \in \tilde{\Omega}_{j-1}^{(c,e)}$, do the following to generate the next state set $\tilde{\Omega}_j^{(c,e)}$:

(2.1) Put $(\tau^{(c,e)} + p_j, f^{(c,e)})$ in $\tilde{\Omega}_j^{(c)}$ if $\tau^{(c,e)} + p_j \leq d$ and $J_j \neq J^{(e)}$;

(2.2) Put $(\tau^{(c,e)}, f^{(c,e)} + w_j p_j)$ in $\tilde{\Omega}_j^{(c,e)}$ if $\tau_j'^{(c,e)} + p_j \leq \bar{d}_j$.

(2.3) If the current $\tilde{\Omega}_j^{(c,e)}$ is not empty, then for every $y \in \{1, 2, \ldots, \nu\}$ with

$$\{(\tau, f) \in \tilde{\Omega}_j^{(c,e)} : f \in I_y\} \neq \emptyset,$$

we define $(\tau_y^{(c,e)}, f_y^{(c,e)})$ as the state in $\{(\tau, f) \in \tilde{\Omega}_j^{(c,e)} : f \in I_y\}$ such that $\tau_y^{(c,e)}$ is as small as possible.

(2.4) Reset $\tilde{\Omega}_j^{(c,e)} = \{(\tau_y^{(c,e)}, f_y^{(c,e)}) : 1 \leq y \leq \nu,\ \{(\tau, f) \in \tilde{\Omega}_j^{(c,e)} : f \in I_y\} \neq \emptyset\}$.

Step 3: Calculate the approximation value of problem $\mathcal{P}(J^{(c)}, J^{(e)})$ by

$$\tilde{F}_\epsilon^{(c,e)} = \min_{(\tau^{(c,e)}, f^{(c,e)}) \in \tilde{\Omega}_m^{(c,e)}} \{w^{(c)}(\tau^{(c,e)} + p^{(c)} - d) + f^{(c,e)} : \tau^{(c,e)} + p^{(c)} \leq \bar{d}^{(c)}\}.$$

Then generate the schedule $\sigma_\epsilon^{(c,e)}$ corresponding to $\tilde{F}_\epsilon^{(c,e)}$ by backtracking. In the case where $\tilde{F}_\epsilon^{(c,e)} = +\infty$, problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is infeasible, so we set $\sigma_\epsilon^{(c,e)} = \emptyset$.

Step 4: Output the approximation value $\tilde{F}_\epsilon^{(c,e)}$ and the approximation schedule $\sigma_\epsilon^{(c,e)}$.

Since each $\tilde{\Omega}_j^{(c,e)}$ in Algorithm $A_\epsilon^{(c,e)}$ has at most $\nu = \lceil \frac{(n-1)^2}{\epsilon} \rceil$ states, we have the following lemma.

Lemma 2.4. *Algorithm $A_\epsilon^{(c,e)}$ runs in $O(n\nu) = O(n^3/\epsilon)$ time.*

Lemma 2.5. *Let $(J^{(c)}, J^{(e)})$ be a pair of jobs of $\mathcal{J}$ such that problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is feasible. Then, for each $j \in \{0, 1, \ldots, m\}$ and for each state $(\tau^{(c,e)}, f^{(c,e)}) \in \Omega_j^{(c,e)}$, Algorithm $A_\epsilon^{(c,e)}$ generates at least one state $(\tilde{\tau}^{(c,e)}, \tilde{f}^{(c,e)}) \in \tilde{\Omega}_j^{(c,e)}$ such that $\tilde{\tau}^{(c,e)} \leq \tau^{(c,e)}$ and $\tilde{f}^{(c,e)} \leq f^{(c,e)} + j\mu$.*

Proof. We prove the lemma by induction on j. From Algorithm $DP(J^{(c)}, J^{(e)})$ and Algorithm $A_\epsilon^{(c,e)}$, we have $\Omega_0^{(c,e)} = \tilde{\Omega}_0^{(c,e)}$. Hence, the lemma holds for $j = 0$.

Suppose in the following that the lemma holds up to $j - 1$. Let $(\tau^{(c,e)}, f^{(c,e)})$ be a state in $\Omega_j^{(c,e)}$. Then Algorithm $DP(J^{(c)}, J^{(e)})$ introduces this state into $\Omega_j^{(c,e)}$ when job J_j is added to some feasible state for jobs $J_1, \ldots, J_{j-1}$. This means that there is a state $(\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)}) \in \Omega_{j-1}^{(c,e)}$ such that either $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)} + p_j, f_{j-1}^{(c,e)})$ or $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)} + w_j p_j)$. Since $(\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)}) \in \Omega_{j-1}^{(c,e)}$, from the induction hypothesis, there exists some state $(\tilde{\tau}, \tilde{f}) \in \tilde{\Omega}_{j-1}^{(c,e)}$ such that

$$\tilde{\tau} \leq \tau_{j-1}^{(c,e)} \text{ and } \tilde{f} \leq f_{j-1}^{(c,e)} + (j-1)\mu. \tag{5}$$

We distinguish the following two cases.

Case 1: $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)} + p_j, f_{j-1}^{(c,e)})$. Since $(\tilde{\tau}, \tilde{f}) \in \tilde{\Omega}_{j-1}^{(c,e)}$ and $\tilde{\tau} \leq \tau_{j-1}^{(c,e)}$ from (5), we have $\tilde{\tau} + p_j \leq \tau_{j-1}^{(c,e)} + p_j = \tau^{(c,e)} \leq d$. Consequently, the state $(\tilde{\tau} + p_j, \tilde{f})$ enters $\tilde{\Omega}_j^{(c,e)}$ in Step (2.1) of Algorithm $A_\epsilon^{(c,e)}$ in iteration j. Assume that $\tilde{f} \in I_y$ for some $y = 1, 2, \ldots, \nu$. Then, at the end of Step (2.4) of Algorithm $A_\epsilon^{(c,e)}$ in iteration j, there is a state $(\tilde{\tau}^{(c,e)}, \tilde{f}^{(c,e)}) \in \tilde{\Omega}_j^{(c,e)}$ such that $\tilde{\tau}^{(c,e)} \leq \tilde{\tau} + p_j$ and $\tilde{f}^{(c,e)}, \tilde{f} \in I_y$. Recall from (4) that $I_y = [(y-1)\mu, y\mu)$.

Now, from (5), we have

$$\tilde{\tau}^{(c,e)} \leq \tilde{\tau} + p_j \leq \tau_{j-1}^{(c,e)} + p_j = \tau^{(c,e)}$$

and

$$\tilde{f}^{(c,e)} \leq \tilde{f} + \mu \leq f_{j-1}^{(c,e)} + (j-1)\mu + \mu = f^{(c,e)} + j\mu.$$

Then $(\tilde{\tau}^{(c,e)}, \tilde{f}^{(c,e)})$ is a required state in $\tilde{\Omega}_j^{(c,e)}$.

Case 2: $(\tau^{(c,e)}, f^{(c,e)}) = (\tau_{j-1}^{(c,e)}, f_{j-1}^{(c,e)} + w_j p_j)$. Since $(\tilde{\tau}, \tilde{f}) \in \tilde{\Omega}_{j-1}^{(c,e)}$ and $\tilde{\tau} \leq \tau_{j-1}^{(c,e)} = \tau^{(c,e)} \leq d$, we have $\tilde{\tau}'_j = TP - (\sum_{k=1}^{j} p_k + p(\mathcal{J}_1^{(c,e)}) - \tilde{\tau}) \leq TP - (\sum_{k=1}^{j} p_k + p(\mathcal{J}_1^{(c,e)}) - \tau^{(c,e)}) = \tau_j'^{(c,e)}$. Then $\tilde{\tau} \leq d$ and $\tilde{\tau}'_j + p_j \leq \tau_j'^{(c,e)} + p_j \leq \bar{d}_j$. Consequently, the state $(\tilde{\tau}, \tilde{f} + w_j p_j)$ enters $\tilde{\Omega}_j^{(c,e)}$ in Step (2.2) of Algorithm $A_\epsilon^{(c,e)}$ in iteration j. Assume that $\tilde{f} + w_j p_j \in I_y$ for some $y = 1, 2, \ldots, \nu$. Then, at the end of Step (2.4) of Algorithm $A_\epsilon^{(c,e)}$ in iteration j, there is a state $(\tilde{\tau}^{(c,e)}, \tilde{f}^{(c,e)}) \in \tilde{\Omega}_j^{(c,e)}$ such that $\tilde{\tau}^{(c,e)} \leq \tilde{\tau}$ and $\tilde{f}^{(c,e)}, \tilde{f} + w_j p_j \in I_x$.

Now, from (5), we have

$$\tilde{\tau}^{(c,e)} \leq \tilde{\tau} \leq \tau_{j-1}^{(c,e)} = \tau^{(c,e)}$$

and

$$\begin{aligned} \tilde{f}^{(c,e)} &\leq \tilde{f} + w_j p_j + \mu \\ &\leq f_{j-1}^{(c,e)} + (j-1)\mu + w_j p_j + \delta(j, \tilde{\tau}') + \mu \\ &= f_{j-1}^{(c,e)} + w_j p_j + j\mu \\ &= f^{(c,e)} + j\mu. \end{aligned}$$

It follows that $(\tilde{\tau}^{(c,e)}, \tilde{f}^{(c,e)})$ is a required state in $\tilde{\Omega}_j^{(c,e)}$.

From the above discussion, the result follows from the principle of induction. □

Recall that $F^*(c,e)$ is the optimal value of problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Lemma 2.6. *Assume that $J^{(c)}$ and $J^{(e)}$ are a pair of jobs of $\mathcal{J}$ such that problem $\mathcal{P}(J^{(c)}, J^{(e)})$ is feasible. Then, for an arbitrary constant $\epsilon > 0$, Algorithm $A_\epsilon^{(c,e)}$ outputs a feasible schedule $\sigma_\epsilon = \sigma_\epsilon^{(c,e)}$ with the function value $\tilde{F}_\epsilon^{(c,e)}$ such that $\tilde{F}_\epsilon^{(c,e)} \leq (1+\epsilon)F^*(c,e)$.*

Proof. Let σ^* be the optimal schedule for problem $\mathcal{P}(J^{(c)}, J^{(e)})$ that is obtained by Algorithm $DP(J^{(c)}, J^{(e)})$. Then σ^* is associated with a state $(\tau^{(c,e)}, f^{(c,e)})$ in $\Omega_m^{(c,e)}$. It follows that

$$\tau^{(c,e)} + p^{(c)} \leq \bar{d}^{(c)} \tag{6}$$

and

$$F^*(c,e) = w^{(c)}Y^{(c)}(\tau^{(c,e)}) + f^{(c,e)}. \tag{7}$$

From Lemma 2.5, there exists a state $(\tilde{\tau}, \tilde{f})$ in $\tilde{\Omega}_m^{(c,e)}$ such that

$$\tilde{\tau} \leq \tau^{(c,e)} \tag{8}$$

and

$$\tilde{f} \leq f^{(c,e)} + m\mu \leq f^{(c,e)} + (n-1)\mu. \tag{9}$$

From (6) and (8), we have $\tilde{\tau} + p^{(c)} \leq \tau^{(c,e)} + p^{(c)} \leq \bar{d}^{(c)}$. This means that there is a feasible schedule, denoted by π, for problem $\mathcal{P}(J^{(c)}, J^{(e)})$ such that $\sum_{j=1}^{n} Z_j(\pi) = w^{(c)}Y^{(c)}(\tilde{\tau}) + \tilde{f}$. Since Algorithm $A_\epsilon^{(c,e)}$ outputs the feasible schedule $\sigma_\epsilon = \sigma_\epsilon^{(c,e)}$ for problem $\mathcal{P}(J^{(c)}, J^{(e)})$ with the objective value $\tilde{F}_\epsilon^{(c,e)}$, we have

$$\tilde{F}_\epsilon^{(c,e)} \leq w^{(c)}Y^{(c)}(\tilde{\tau}) + \tilde{f}. \tag{10}$$

Note that the relation $\tilde{\tau} \leq \tau^{(c,e)}$ in (8) implies that $Y^{(c)}(\tilde{\tau}) \leq Y^{(c)}(\tau^{(c,e)})$. Thus, from (2), (7), (9), and (10), we have

$$\begin{aligned}
\tilde{F}_\epsilon^{(c,e)} &\leq w^{(c)}Y^{(c)}(\tilde{\tau}) + \tilde{f} \\
&\leq w^{(c)}Y^{(c)}(\tau^{(c,e)}) + \tilde{f} \\
&\leq w^{(c)}Y^{(c)}(\tau^{(c,e)}) + f^{(c,e)} + (n-1)\mu \\
&= w^{(c)}Y^{(c)}(\tau^{(c,e)}) + f^{(c,e)} + (n-1)\tfrac{(n-1)w^{(e)}p^{(e)}}{\nu} \\
&= F^*(c,e) + \tfrac{(n-1)^2 w^{(e)}p^{(e)}}{\nu} = F^*(c,e) + \tfrac{(n-1)^2 w^{(e)}p^{(e)}}{\lceil \frac{(n-1)^2}{\epsilon} \rceil} \\
&\leq F^*(c,e) + \epsilon w^{(e)}p^{(e)} \leq F^*(c,e) + \epsilon F^*(c,e) \\
&= (1+\epsilon)F^*(c,e),
\end{aligned}$$

as required. □

Recall that $TP = \sum_{J_j \in \mathcal{J}} p_j$. For each job $J^{(c)} \in \mathcal{J}$, let $\pi^{(c)}$ be the schedule of $\mathcal{J}$ in which $J^{(c)}$ is the last job. If $TP - p^{(c)} \leq d$ and $TP \leq \bar{d}^{(c)}$, then $\pi^{(c)}$ is a feasible schedule for problem $\mathcal{P}(J^{(c)})$. The objective value of $\pi^{(c)}$, denoted $F(\pi^{(c)})$, is

$$F(\pi^{(c)}) = w^{(c)}(TP - d).$$

Alternatively, if either $TP - p^{(c)} > d$ or $TP > \bar{d}^{(c)}$, then $\pi^{(c)}$ is not a feasible schedule for problem $\mathcal{P}(J^{(c)})$. In this case, we define

$$F(\pi^{(c)}) = +\infty.$$

Now we present an FPTAS for problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$ by using Algorithm $A_\epsilon^{(c,e)}$ as a subroutine.

Algorithm A_ϵ**:** For problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$.

Input: A problem instance with a set of jobs $\mathcal{J}$ and an constant $\epsilon > 0$.

Step 1: For each job $J^{(c)} \in \mathcal{J}$, set $\pi^{(c)}$ as an arbitrary schedule of $\mathcal{J}$ in which $J^{(c)}$ is the last job. If $TP - p^{(c)} \leq d$ and $TP \leq \bar{d}^{(c)}$, then set $F(\pi^{(c)}) = w^{(c)}(TP - d)$. Otherwise, set $F(\pi^{(c)}) = +\infty$.

Step 2: For each pair of jobs $(J^{(c)}, J^{(e)}) \in \mathcal{J}$, run Algorithm $A_\epsilon^{(c,e)}$ to obtain the approximation value $\tilde{F}_\epsilon^{(c,e)}$ and the approximation schedule $\sigma_\epsilon^{(c,e)}$ for problem $\mathcal{P}(J^{(c)}, J^{(e)})$.

Step 3: Calculate

$$\tilde{F}_\epsilon = \min \begin{cases} \min\{F(\pi^{(c)}) : J^{(c)} \in \mathcal{J}\}, \\ \min\{\tilde{F}_\epsilon^{(c,e)} : J^{(c)}, J^{(e)} \in \mathcal{J}\} \end{cases}$$

and determine the corresponding schedule σ_ϵ as follows:

– If $\tilde{F}_\epsilon = F(\pi^{(c)})$ for some $J^{(c)} \in \mathcal{J}$, then set $\sigma_\epsilon = \pi^{(c)}$;

– If $\tilde{F}_\epsilon = \tilde{F}_\epsilon^{(c,e)}$ for some $J^{(c)}, J^{(e)} \in \mathcal{J}$, and $\tilde{F}_\epsilon < \min\{F(\pi^{(c)}) : J^{(c)} \in \mathcal{J}\}$, then set $\sigma_\epsilon = \sigma_\epsilon^{(c,e)}$.

Step 4: Calculate $F_\epsilon = \sum_{j=1}^n w_j Y_j(\sigma_\epsilon)$.

Output: The approximation value F_ϵ and the approximation schedule σ_ϵ for problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$.

Lemma 2.7. $F_\epsilon \leq \tilde{F}_\epsilon$.

Proof. If $\tilde{F}_\epsilon = F(\pi^{(c)})$ for some $J^{(c)} \in \mathcal{J}$, then $\sigma_\epsilon = \pi^{(c)}$. In this case, $F_\epsilon = \sum_{j=1}^n w_j Y_j(\sigma_\epsilon) = \sum_{j=1}^n w_j Y_j(\pi^{(c)}) = F(\pi^{(c)}) = \tilde{F}_\epsilon$.

Suppose in the following that $\tilde{F}_\epsilon < \min\{F(\pi^{(c)}) : J^{(c)} \in \mathcal{J}\}$. Then $\tilde{F}_\epsilon = \tilde{F}_\epsilon^{(c,e)}$ and $\sigma_\epsilon = \sigma_\epsilon^{(c,e)}$ for some $J^{(c)}, J^{(e)} \in \mathcal{J}$. Then σ_ϵ is a feasible schedule for problem $\mathcal{P}(J^{(c)}, J^{(e)})$ such that $\sum_{j=1}^n Z_j(\sigma_\epsilon) = \tilde{F}_\epsilon$. From Lemma 2.3, we have $F_\epsilon = \sum_{j=1}^n w_j Y_j(\sigma_\epsilon) \leq \sum_{j=1}^n Z_j(\sigma_\epsilon) = \tilde{F}_\epsilon$. This proves the lemma. □

Theorem 2.3. *For problem* $1|d_j = d, \bar{d}_j| \sum w_j Y_j$*, Algorithm* A_ϵ *is a* $(1+\epsilon)$*-approximation algorithm of running time* $O(n^5/\epsilon)$*.*

Proof. The time complexity of Algorithm A_ϵ is determined by Step 2, in which we invoke Algorithm $A_\epsilon^{(c,e)}$ for each pair of jobs $(J^{(c)}, J^{(e)}) \in \mathcal{J}$. From Lemma 2.4, for each given pair $(J^{(c)}, J^{(e)})$, Algorithm $A_\epsilon^{(c,e)}$ runs in $O(n^3/\epsilon)$ time. Since we have a total of $O(n^2)$ choices for $(J^{(c)}, J^{(e)})$, Step 2 of Algorithm A_ϵ runs in $O(n^5/\epsilon)$ time.

Now let π^* be an optimal schedule for problem $1|d_j = d, \bar{d}_j| \sum w_j Y_j$. Let $J^{(c^*)}$ be the critical job in π^*. To complete the proof, it suffices to show that

$$F_\epsilon \leq (1+\epsilon) \sum_{j=1}^{n} w_j Y_j(\pi^*). \tag{11}$$

If $J^{(c^*)}$ is the last job in π^*, from Lemma 2.7, we have $\sum_{j=1}^n w_j Y_j(\pi^*) = \sum_{j=1}^n w_j Y_j(\pi^{(c^*)}) = F(\pi^{(c^*)}) \geq \tilde{F}_\epsilon \geq F_\epsilon$, as required in (11).

Suppose in the following that $J^{(c^*)}$ is not the last job in π^*. Let $J^{(e^*)}$ be a job with the largest weighted processing time scheduled after $J^{(c^*)}$ in π^*. Then π^* is also an optimal schedule for problem $\mathcal{P}(J^{(c^*)}, J^{(e^*)})$, so $\sum_{j=1}^n w_j Y_j(\pi^*) = \sum_{j=1}^n Z_j(\pi^*) = F^*(c^*, e^*)$. From Lemmas 2.6 and 2.7, we have $F_\epsilon \leq \tilde{F}_\epsilon \leq \tilde{F}_\epsilon^{(c,e)} \leq (1+\epsilon) F^*(c^*, e^*) = (1+\epsilon) \sum_{j=1}^n w_j Y_j(\pi^*)$, as required in (11). □

3 Problem $1|\bar{d}_j| \sum Y_j$

In this section, we show the unary NP-hardness of problem $1|\bar{d}_j| \sum Y_j$ by a reduction from the 3-Partition problem, which is known to be unary NP-complete (see, e.g., Garey and Johnson, 1979). Our proof imitates the unary NP-harness proof for problem $1|\bar{d}_j| \sum U_j$ in Yuan (2017).

3-Partition: Given a set of $3t+1$ positive integers $x_1, x_2, \ldots, x_{3t}, X$ such that $\sum_{j=1}^{3t} x_j = tX$ and $X/4 < x_j < X/2$ for each j with $1 \leq j \leq 3t$, does there exist a partition $(I_1, I_2, \ldots, I_t)$ of the index set $\{1, 2, \ldots, 3t\}$ such that $|I_i| = 3$ and $\sum_{j \in I_i} x_j = X$ for each i with $1 \leq i \leq t$?

Without loss of generality, we assume that $t \geq 2$ and the $3t$ positive integers in the instance of 3-Partition are sorted such that $x_1 \leq x_2 \leq \cdots \leq x_{3t}$ in the sequel. Given an

instance $(x_1, x_2, \ldots, x_{3t}, X)$ of 3-Partition, we define

$$\begin{cases} \Delta_2 &= 3t^3X, \\ \Delta_1 &= 3t^3(\Delta_2 + X), \\ H &= 3t(\Delta_1 + t\Delta_2 + tX) + 1. \end{cases} \tag{12}$$

Then we have $\Delta_1 < 12t^6X$ and $H < (3t+1)\Delta_1$. An instance of the scheduling problem $1|\bar{d}_j|\sum Y_j$, called scheduling instance, can be constructed as follows.

There are totally $n = 3t^2 + 1$ jobs of two types: a *restriction job* J_0 and $3t^2$ *normal jobs* $J_{i,j}$, $i \in \{1, 2, \ldots, t\}$ and $j \in \{1, 2, \ldots, 3t\}$. For each i with $1 \le i \le t$, we set $\mathcal{J}^{(i)} = \{J_{i,j} : 1 \le j \le 3t\}$, and for each j with $1 \le j \le 3t$, we set $\mathcal{J}_j = \{J_{i,j} : 1 \le i \le t\}$. Then $\{J_{i,j}\} = \mathcal{J}^{(i)} \cap \mathcal{J}_j$ for $i \in \{1, 2, \ldots, t\}$ and $j \in \{1, 2, \ldots, 3t\}$. The processing times, due dates, and deadlines of the n jobs are displayed in Table 2.

Table 2: The scheduling instance

Job	Processing time	Due date	Deadline
J_0	$p_0 = H$	$d_0 = H + \frac{1}{t}\sum_{i=1}^{t} p(\mathcal{J}^{(i)})$	$\bar{d}_0 = d_0$
$J_{i,j}$	$p_{i,j} = \Delta_1 + i\Delta_2 + ix_j$	$d_{i,j} = j(\Delta_1 + t\Delta_2 + tX)$	$\bar{d}_{i,j} = H + \frac{1}{t}\sum_{k=1}^{i-1} p(\mathcal{J}^{(k)}) + \sum_{k=i}^{t} p(\mathcal{J}^{(k)})$

Note that

$$\max_{i,j}\{d_{i,j}\} < H \tag{13}$$

and the total processing time of the jobs of $\mathcal{J}^{(i)}$ is given by

$$p(\mathcal{J}^{(i)}) = 3t\Delta_1 + 3ti\Delta_2 + tiX, \text{ for each } i \in \{1, 2, \ldots, t\}. \tag{14}$$

The threshold value for the total late work is given by

$$Q = t(t-1)(3\Delta_1 + \frac{3(t+1)}{2}\Delta_2 + \frac{t+1}{2}X) = \frac{t-1}{t}\sum_{i=1}^{t} p(\mathcal{J}^{(i)}).$$

The decision asks if there is a feasible schedule π for the constructed scheduling instance such that $\sum Y_j(\pi) \le Q$, i.e., $\sum_{i=1}^{t}\sum_{j=1}^{3t} Y_{i,j}(\pi) + Y_0(\pi) \le Q$. For convenience, we call such a schedule σ a *desired schedule* in the sequel.

In the constructed scheduling instance, there are at most $9t^2 + 4$ numbers and the largest one is $\bar{d}_{1,t} = H + \sum_{i=1}^{t} p(\mathcal{J}^{(i)}) < (3t+1)\Delta_1 + t(3t+1)\Delta_1 < 7t^2 \cdot 12t^6X = 84t^8X$, which is a polynomial in t and X. Thus, the instance construction can be performed in polynomial time in unary encoding.

From the definition of Δ_1 and Δ_2, we have

$$\Delta_1 - \frac{3t(t-1)(t+1)}{2}\Delta_2 - \frac{t(t-1)(t+1)}{2}X > 3t^3(\Delta_2+X) - \frac{3t^3}{2}\Delta_2 - \frac{t^3}{2}X > \frac{3t^3}{2}(\Delta_2+X) > 0,$$

and

$$\Delta_2 - \frac{t(t-1)(t+1)}{2}X > 3t^3X - \frac{t^3}{2}X = \frac{5t^3}{2}X > 0.$$

From the above two inequalities, together with the definition of Q, we have

$$Q < (3t^2 - 3t + 1)\Delta_1 \tag{15}$$

and

$$Q < (3t-3)(t\Delta_1 + \frac{t(t+1)}{2}\Delta_2) + \Delta_2. \tag{16}$$

The following two lemma reveals some important properties of a desired schedule.

Lemma 3.1. *Let π be a desired schedule. Then $Y_0(\pi) = 0$ and exactly $3t$ normal jobs are scheduled before J_0 in π.*

Proof. Given the feasibility of π and by the relation $d_0 = \bar{d}_0$, J_0 must be early in π. Thus, $Y_0(\pi) = 0$. Note that each normal job has a processing time greater than Δ_1 and $\frac{1}{t}\sum_{i=1}^{t} p(\mathcal{J}^{(i)}) = 3t\Delta_1 + \frac{3t(t+1)}{2}\Delta_2 + \frac{t(t+1)}{2}X < (3t+1)\Delta_1$.

If more than $3t$ normal jobs are scheduled before J_0 in π, then $C_0(\pi) > H + (3t+1)\Delta_1 > H + \frac{1}{t}\sum_{j=1}^{t} p(\mathcal{J}^{(i)}) = \bar{d}_0$, a contradiction.

If fewer than $3t$ normal jobs are scheduled before J_0 in π, then at least $3t^2 - 3t + 1$ normal jobs are scheduled after J_0. From (13) and the relation $p_0 = H$, all the normal jobs scheduled after J_0 are late in π. Thus, $\sum Y_j(\pi) > (3t^2 - 3t + 1)\Delta_1$. From (15), we conclude that $\sum Y_j(\pi) > Q$, a contradiction again.

Now the lemma follows from the above discussion. □

Lemma 3.2. *Let π be a desired schedule. Then each $\mathcal{J}^{(i)}$ with $1 \le i \le t$ contains exactly three early jobs in π.*

Proof. For each $i \in \{1, 2, \ldots, t\}$, we use $\mathcal{V}^{(i)} = \mathcal{V}^{(i)}(\pi)$ to denote the set of jobs of $\mathcal{J}^{(i)}$ that are early in π. From Lemma 3.1, we have

$$|\mathcal{V}^{(1)}| + |\mathcal{V}^{(2)}| + \cdots + |\mathcal{V}^{(t)}| = 3t. \tag{17}$$

In the following, we will show that $|\mathcal{V}^{(i)}| = 3$ for each $i \in \{1, 2, \ldots, t\}$.

Note that J_0 meets its deadline $\bar{d}_0$ in π and all the jobs of $\mathcal{V}^{(1)} \cup \mathcal{V}^{(2)} \cup \cdots \cup \mathcal{V}^{(t)}$ are scheduled before job J_0 in π. Since $p_0 = H$ and $\bar{d}_0 = H + 3t\Delta_1 + \frac{3t(t+1)}{2}\Delta_2 + \frac{t(t+1)}{2}X$, we have

$$p(\mathcal{V}^{(1)}) + p(\mathcal{V}^{(2)}) + \cdots + p(\mathcal{V}^{(t)}) \leq 3t\Delta_1 + \frac{3t(t+1)}{2}\Delta_2 + \frac{t(t+1)}{2}X. \tag{18}$$

For each $i \in \{1, 2, \ldots, t-1\}$, all the jobs of $\{J_0\} \cup \mathcal{V}^{(1)} \cup \mathcal{V}^{(2)} \cup \cdots \cup \mathcal{V}^{(i)}$ and $\mathcal{J}^{(t)} \cup \mathcal{J}^{(t-1)} \cup \cdots \cup \mathcal{J}^{(i+1)}$ are completed by the common deadline of the t normal jobs of $\mathcal{J}^{(i+1)}$ in π, subject to the deadlines. This means that $H + p(\mathcal{V}^{(1)}) + p(\mathcal{V}^{(2)}) + \cdots + p(\mathcal{V}^{(i)}) + p(\mathcal{J}^{(t)}) + p(\mathcal{J}^{(t-1)}) + \cdots + p(\mathcal{J}^{(i+1)}) \leq H + \frac{1}{t}\sum_{k=1}^{i} p(\mathcal{J}^{(k)}) + \sum_{k=i+1}^{t} p(\mathcal{J}^{(k)})$. Then we have $p(\mathcal{V}^{(1)}) + p(\mathcal{V}^{(2)}) + \cdots + p(\mathcal{V}^{(i)}) \leq \frac{1}{t}\sum_{k=1}^{i} p(\mathcal{J}^{(k)})$. From (14), we have

$$p(\mathcal{V}^{(1)}) + p(\mathcal{V}^{(2)}) + \cdots + p(\mathcal{V}^{(i)}) \leq 3i\Delta_1 + \frac{3i(i+1)}{2}\Delta_2 + \frac{i(i+1)}{2}X,\ i = 1, 2, \ldots, t-1. \tag{19}$$

From (18) and (19) and by using the same discussion in Yuan (2017), we have that $|\mathcal{V}^{(t)}| + |\mathcal{V}^{(t-1)}| + \cdots + |\mathcal{V}^{(i+1)}| = 3(t-i)$ for every index i with $0 \leq i \leq t-1$. This further implies that $|\mathcal{V}^{(i)}| = 3$ for each $i \in \{1, 2, \ldots, t\}$, i.e., each $\mathcal{J}^{(i)}$ has exactly three early jobs in π, $1 \leq i \leq t$. The lemma follows. □

We show in the following that $(x_1, x_2, \ldots, x_{3t}, X)$ is a YES instance of 3-Partition if and only if there is a desired schedule π for the scheduling instance.

The "if" part proof: Assume that $(x_1, x_2, \ldots, x_{3t}, X)$ is a YES instance of 3-Partition. Then $\{1, 2, \ldots, 3t\}$ can be partitioned into t subsets $I_1, I_2, \ldots, I_t$, each of size three, such that $\sum_{j \in I_i} x_j = X$ for each $i \in \{1, 2, \ldots, t\}$. We define a schedule π of the $n = 3t^2 + 1$ jobs in the following way:

– For each $j \in \{1, 2, \ldots, 3t\}$, let $i(j)$ be the unique index in $\{1, 2, \ldots, t\}$ so that $j \in I_{i(j)}$.

– Set $\mathcal{V} = \{J_{i(1),1}, J_{i(2),2}, \ldots, J_{i(3t),3t}\}$.

– The $n = 3t^2 + 1$ jobs are scheduled in the following order in π:

$$J_{i(1),1} \prec J_{i(2),2} \prec \cdots \prec J_{i(3t),3t} \prec J_0 \prec \mathcal{J}^{(t)} \setminus \mathcal{V} \prec \mathcal{J}^{(t-1)} \setminus \mathcal{V} \prec \cdots \prec \mathcal{J}^{(1)} \setminus \mathcal{V}.$$

From the definition of schedule π, we have $|\mathcal{V} \cap \mathcal{J}^{(i)}| = |I_i| = 3$ for each $i \in \{1, 2, \ldots, t\}$. Let $\mathcal{V}^{(i)} = \mathcal{V} \cap \mathcal{J}^{(i)}$, $1 \leq i \leq t$. From the definition of $p_{i,j}$ and the definition of $(I_1, I_2, \ldots, I_t)$, we have

$$p(\mathcal{V}^{(i)}) = 3\Delta_1 + 3i\Delta_2 + iX = \frac{1}{t}p(\mathcal{J}^{(i)}) \text{ for each } i = 1, 2, \ldots, t. \tag{20}$$

The completion time of J_0 in π is $C_0(\pi) = \sum_{i=1}^{t} p(\mathcal{V}^{(i)}) + p_0 = H + \frac{1}{t}\sum_{i=1}^{t} p(\mathcal{J}^{(i)}) = d_0$, so J_0 is early in π.

For each $i \in \{1, 2, \ldots, t\}$, all the normal jobs of $\mathcal{J}^{(i)}$ have a common deadline, denoted by

$$\bar{d}^{(i)} = H + \frac{1}{t}\sum_{k=1}^{i-1} p(\mathcal{J}^{(k)}) + \sum_{k=i}^{t} p(\mathcal{J}^{(k)}). \tag{21}$$

By the structure of π, the maximum completion time of the jobs of $\mathcal{J}^{(i)}$ with $1 \le i \le t$ is $p_0 + p(\mathcal{V}^{(1)}) + p(\mathcal{V}^{(2)}) + \cdots + p(\mathcal{V}^{(i-1)}) + p(\mathcal{J}^{(t)}) + p(\mathcal{J}^{(t-1)}) + \cdots + p(\mathcal{J}^{(i)}) = H + \frac{1}{t}\sum_{k=1}^{i-1} p(\mathcal{J}^{(k)}) + \sum_{k=i}^{t} p(\mathcal{J}^{(k)}) = \bar{d}^{(i)}$. Thus, all the normal jobs meet their deadlines in π. It follows that π is a feasible schedule.

Note that each normal job has a processing time less than $\Delta_1 + t\Delta_2 + tX$. Then the completion time of $J_{i(j),j}$ is less than $j(\Delta_1 + t\Delta_2 + tB) = d_{i(j),j}$. So the first $3t$ normal jobs in π are early, i.e., $Y_{i(j),j}(\pi) = 0$, $1 \le j \le 3t$. From (13) and the fact that $p_0 = H$, the normal jobs scheduled after J_0 are all late, so their late work is given by their processing times. Moreover, since J_0 is early in π, we have $Y_0(\pi) = 0$. Thus, from (14) and (20), we have

$$\begin{aligned} & \textstyle\sum_{i=1}^{t}\sum_{j=1}^{3t} Y_{i,j}(\pi) + Y_0(\pi) = \sum_{i=1}^{t} p(\mathcal{J}^{(i)} \setminus \mathcal{V}) \\ = \; & \textstyle\sum_{i=1}^{t}(p(\mathcal{J}^{(i)}) - p(\mathcal{V}^{(i)})) = \sum_{i=1}^{t} \frac{t-1}{t}(p(\mathcal{J}^{(i)}) \\ = \; & Q. \end{aligned}$$

It follows that π is a desired schedule.

The "only if" part proof: Assume that there exist desired schedules for the scheduling instance. Let π be a desired schedule such that $|\{j \in \{1, 2, \ldots, 3t\} : J_{\pi(j)} \in \mathcal{J}_j\}|$ is as large as possible. Then we have the following statement.

Statement 2. *For each $j \in \{1, 2, \ldots, 3t\}$, we have $J_{\pi(j)} \in \mathcal{J}_j$.*

Proof of Statement 2. Otherwise, there is an index $j \in \{1, 2, \ldots, 3t\}$ such that $J_{\pi(j)} \notin \mathcal{J}_j$ and $J_{\pi(k)} \in \mathcal{J}_k$, $1 \le k \le j-1$.

Suppose first that $J_{\pi(j)} \in \mathcal{J}_1 \cup \mathcal{J}_2 \cup \cdots \cup \mathcal{J}_{j-1}$. Note that each normal job has a processing time larger than Δ_1. Then $C_{\pi(j)} > j\Delta_1$, and so, $Y_{\pi(j)} > j\Delta_1 - (j-1)(\Delta_1 + t\Delta_2 + tX) > \Delta_2$. From (13) and the fact that $p_0 = H$, the $3t^2 - 3t$ normal jobs scheduled after J_0 are late. From Lemma 3.2, there are exactly $3t - 3$ normal jobs of each $\mathcal{J}^{(i)}$ scheduled after J_0. Moreover, each normal job in $\mathcal{J}^{(i)}$ has a processing time larger than $\Delta_1 + i\Delta_2$, $1 \le i \le t$. Thus, the total late work of these $3t^2 - 3t$ normal jobs is larger than $(3t-3)\sum_{i=1}^{t}(\Delta_1 + i\Delta_2) = (3t-3)(t\Delta_1 + \frac{t(t+1)}{2}\Delta_2)$. Then we have $\sum Y_j(\pi) > (3t-3)(t\Delta_1 + \frac{t(t+1)}{2}\Delta_2) + \Delta_2$. From (16), we have $\sum Y_j(\pi) > Q$, a contradiction. Hence, we must have $J_{\pi(j)} \in \mathcal{J}_{j+1} \cup \mathcal{J}_{j+2} \cup \cdots \cup \mathcal{J}_{3t}$. This further implies that $j \le 3t - 1$.

Assume in the following that $J_{\pi(j)} = J_{i,j'}$ for some $j' \in \{j+1, j+2, \cdots, 3t\}$ and $i \in \{1, 2, \cdots, t\}$. Then $J_{\pi(j)} = J_{i,j'} \prec J_{i,j}$ in π. Since $j < j'$, we have $p_{i,j} \le p_{i,j'}$ and

$d_{i,j} < d_{i,j'}$. Let π' be the schedule obtained from π by swapping the two jobs $J_{i,j'}$ and $J_{i,j}$. Since $p_{i,j} \leq p_{i,j'}$, and $J_{i,j'}$ and $J_{i,j}$ have a common deadline, π' is still a feasible schedule in which

$$\begin{cases} C_{i,j}(\pi') \leq C_{i,j'}(\pi), \\ C_{i,j'}(\pi') = C_{i,j}(\pi), \\ C_{\pi(k)}(\pi') \leq C_{\pi(k)}(\pi), \quad \text{for } J_{\pi(k)} \notin \{J_{i,j'}, J_{i,j}\}. \end{cases} \tag{22}$$

Note that each normal job has a processing time less than $\Delta_1 + t\Delta_2 + tX$. Then $C_{i,j}(\pi') \leq C_{i,j'}(\pi) < j(\Delta_1 + t\Delta_2 + tB) = d_{i,j} < d_{i,j'}$. This implies that

$$Y_{i,j'}(\pi) = 0 \text{ and } Y_{i,j}(\pi') = 0. \tag{23}$$

From the fact that $d_{i,j} < d_{i,j'}$ and from the relation $C_{i,j'}(\pi') = C_{i,j}(\pi)$ in (22), we have

$$Y_{i,j'}(\pi') \leq Y_{i,j}(\pi). \tag{24}$$

From the third relation in (22), we further have

$$Y_{\pi(k)}(\pi') \leq Y_{\pi(k)}(\pi) \text{ for } J_{\pi(k)} \notin \{J_{i,j'}, J_{i,j}\}. \tag{25}$$

Combining (23), (24), and (25), we conclude that $\sum Y_j(\pi') \leq \sum Y_j(\pi) \leq Q$. Thus, π' is also a desired schedule. But then $|\{k \in \{1, 2, \ldots, 3t\} : J_{\pi'(k)} \in \mathcal{J}_k\}| < |\{k \in \{1, 2, \ldots, 3t\} : J_{\pi(k)} \in \mathcal{J}_k\}|$. This contradicts the choice of π. Statement 2 follows.

For each $j \in \{1, 2, \ldots, 3t\}$, we use $\mathcal{V}_j(\pi)$ to denote the set of jobs of $\mathcal{J}_j$ that are early in π. From Lemma 3.1, we have

$$|\mathcal{V}_1(\pi)| + |\mathcal{V}_2(\pi)| + \cdots + |\mathcal{V}_{3t}(\pi)| = 3t. \tag{26}$$

From Statement 2, we have

$$\mathcal{V}_j(\pi) = \{J_{\pi(j)}\} \text{ for } j = 1, 2, \ldots, 3t. \tag{27}$$

For each $i \in \{1, 2, \ldots, t\}$, we use $\mathcal{V}^{(i)}(\pi)$ to denote the set of jobs of $\mathcal{J}^{(i)}$ that are early in π. Then from Lemma 3.2, we have

$$|\mathcal{V}^{(i)}(\pi)| = 3. \tag{28}$$

Now define

$$I_i = \{j : J_j^{(i)} \in \mathcal{V}^{(i)}(\pi)\} \text{ and } X_i = \textstyle\sum_{j \in I_i} x_j, \ i = 1, 2, \ldots, t.$$

From (27) and (28), we have $|I_i| = 3$ for each i with $1 \leq i \leq t$, and $(I_1, I_2, \ldots, I_t)$ forms a partition of $\{1, 2, \ldots, 3t\}$. This implies that

$$X_1 + X_2 + \cdots + X_t = tX. \tag{29}$$

Moreover, we have

$$p(\mathcal{V}^{(i)}(\pi)) = 3\Delta_1 + 3i\Delta_2 + iX_i, \ i = 1, 2, \ldots, t. \tag{30}$$

Since π is a desired schedule, from the discussion in Lemma 3.2, the inequality (19) still holds for π. Combining (19) and (30), we have

$$X_1 + 2X_2 + \cdots + iX_i \leq X + 2X + \cdots + iX \text{ for each } i = 1, 2, \ldots, t. \tag{31}$$

From (29) and (31), and by using the same deduction as in Yuan (2017), we obtain

$$X_i = X \text{ for each } i = 1, 2, \ldots, t. \tag{32}$$

It follows that $(I_1, I_2, \ldots, I_t)$ is a partition of $\{1, 2, \ldots, 3t\}$ such that $|I_i| = 3$ and $\sum_{j \in I_i} x_j = X$ for $i = 1, 2, \ldots, t$. Consequently, the instance $(x_1, x_2, \ldots, x_{3t}, X)$ is a YES instance of 3-Partition.

From the above discussion, we conclude the following theorem.

Theorem 3.1. *Problem $1|\bar{d}_j|\sum Y_j$ is unary NP-hard.*

4 Polynomially solvable cases

In this section we consider two special cases of problem $1|\bar{d}_j|\sum w_j Y_j$, namely (i) the jobs have a common due date and a unit weight, and (ii) the jobs have a common processing time. We provide polynomial-time algorithms to solve the two special cases.

4.1 Problem $1|d_j = d, \bar{d}_j|\sum Y_j$

For problem $1|d_j = d, \bar{d}_j|\sum Y_j$, it is observed that every feasible schedule is optimal. As discussed in the Feasibility Checking part in Section 1, problem $1|d_j = d, \bar{d}_j|\sum Y_j$ can be solved by the following procedure, which uses the idea in Jackson (1955) for solving problem $1||L_{\max}$.

Deadline-EDD: Generate a schedule in which the n jobs are sequenced in nondecreasing order of their deadlines.

Note that Deadline-EDD runs in $O(n \log n)$ time. Thus, we have

Theorem 4.1. *Problem* $1|d_j = d, \bar{d}_j|\sum Y_j$ *is solvable in* $O(n \log n)$ *time by the procedure Deadline-EDD.*

4.2 Problem $1|p_j = p, \bar{d}_j|\sum w_j Y_j$

Hariri et al. (1995) showed that problem $1|p_j = p|\sum w_j Y_j$ is solvable in $O(n^3)$ time. For problem $1|p_j = p, \bar{d}_j|\sum w_j Y_j$, due to the existence of deadlines, we make some adjustments of the approach presented by Hariri et al. (1995) as follows.

We define the cost c_{ij} of scheduling job J_j in the i-th position, $i, j \in \{1, 2, \ldots, n\}$, by setting

$$c_{ij} = \begin{cases} 0, & \text{if } ip \le d_j, \\ w_j \min\{p_j, ip - d_j\}, & \text{if } d_j < ip \le \bar{d}_j, \\ +\infty, & \text{otherwise.} \end{cases}$$

Moreover, we introduce indicator variables x_{ij} with $i, j \in \{1, 2, \ldots, n\}$ such that

$$x_{ij} = \begin{cases} 1, & \text{if job } J_j \text{ is scheduled in the } i\text{-th position,} \\ 0, & \text{otherwise.} \end{cases}$$

Then problem $1|p_j = p, \bar{d}_j|\sum w_j Y_j$ is equivalent to the following $n \times n$ linear assignment problem with costs c_{ij}, $i, j \in \{1, 2, ..., n\}$.

$$\begin{array}{lll} \min & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\ & \sum_{i=1}^{n} x_{ij} = 1, & \text{for all } j \in \{1, \ldots, n\}, \\ & \sum_{j=1}^{n} x_{ij} = 1, & \text{for all } i \in \{1, \ldots, n\}, \\ & x_{ij} \ge 0, & \text{for all } i, j \in \{1, \ldots, n\}. \end{array} \tag{33}$$

From Kuhn (2005), the $n \times n$ linear assignment problem in (33) is solvable in $O(n^3)$ time. Thus, we have the following result.

Theorem 4.2. *Problem* $1|p_j = p, \bar{d}_j|\sum w_j Y_j$ *is solvable in* $O(n^3)$ *time.*

5 Conclusions

In this paper we address the scheduling problem with deadlines to minimize the total weighted late work on a single machine. We show the binary NP-hardness of problem $1|d_j = d, \bar{d}_j|\sum w_j Y_j$ and unary NP-hardness of problem $1|\bar{d}_j|\sum Y_j$. For problem

$1|d_j = d, \bar{d}_j| \sum w_j Y_j$, we also develop a pseudo-polynomial dynamic programming solution algorithm that runs in $O(n^2 d)$ time and a fully polynomial-time approximation scheme (FPTAS). We present an $O(n \log n)$ time algorithm to solve problem $1|d_j = d, \bar{d}_j| \sum Y_j$ and an $O(n^3)$-time algorithm to solve problem $1|p_j = p, \bar{d}_j| \sum w_j Y_j$.

It should be noted that when preemption is allowed, the problems are tractable even in the parallel-machine environment with dynamic jobs. Note that in a feasible preemptive schedule σ, the late work $Y_j(\sigma)$ of job J_j is defined to be the total processing time of the parts of job J_j that are scheduled after its due date d_j. By incorporating job deadlines into the study of Leung (2004), without introducing new techniques, one can show that problems $P|r_j, \bar{d}_j, \text{pmtn}| \sum Y_j$, $P|r_j, \bar{d}_j, \text{pmtn}| \sum w_j Y_j$, $Q|r_j, \bar{d}_j, \text{pmtn}| \sum Y_j$, and $Q|r_j, \bar{d}_j, \text{pmtn}| \sum w_j Y_j$ are solvable in $O(n^3 \log n)$, $O(n^4 \log n)$, $O(m^2 n^3 \log mn)$, and $O(m^2 n^4 \log mn)$ times, respectively, using the network flow technique, where m is the number of machines.

For further research, we suggest the following topics:

– Designing effective approximation algorithms for problem $1|\bar{d}_j| \sum w_j Y_j$. Especially, a PTAS is expected for problem $1|\bar{d}_j| \sum Y_j$.

– Designing more efficient polynomial-time algorithms for problems $1|r_j, \bar{d}_j, \text{pmtn}| \sum Y_j$ and $1|r_j, \bar{d}_j, \text{pmtn}| \sum w_j Y_j$.

– Research on scheduling with the late work criterion under the deadline constraint should be extended to other scheduling settings such as shop scheduling, multi-agent scheduling, batch scheduling, and so on.

Acknowledgments

The authors would like to thank the Editor, the Associate Editor and two anonymous referees for their constructive comments and helpful suggestions. This research was supported in part by NSFC under grant numbers 11671368 and 11771406.

References

Abrishami, S., & Naghibzadeh, M. (2012). Deadline-constrained workflow scheduling in software as a service cloud. Scientia Iranica, 19, 680-689.

Abrishami, S., Naghibzadeh, M., & Epema, D.H. (2013). Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Generation

Computer Systems, 29, 158-169.

Alminana, M., Escudero, L.F., Landete, M., Monge, J.F., Rabasa, A., & Sanchez-Soriano, J. (2010). WISCHE: A DSS for water irrigation scheduling. Omega, 38, 492-500.

Blazewicz, J. (1984). Scheduling preemptible tasks on parallel processors with information loss. TSI-Technique et Science Informatiques, 3, 415-420.

Blazewicz, J., & Finke, G. (1987). Minimizing mean weighted execution time loss on identical and uniform processors. Information Processing Letters, 24, 259-263.

Blazewicz, J., Pesch, E., Sterna, M., & Werner, F. (2005). The two-machine flow-shop problem with weighted late work criterion and common due date. European Journal of Operational Research, 165, 408-415.

Chen, X., Sterna, M., Han, X., & Blazewicz, J. (2016). Scheduling on parallel identical machines with late work criterion: offline and online cases. Journal of Scheduling, 19, 729-736.

Chen, R.B., & Yuan, J.J. (2018a). Unary NP-hardness of single-machine scheduling to minimize the total tardiness with deadlines. In submission.

Chen, R.B., & Yuan, J.J. (2018b). Unary NP-hardness of preemptive scheduling to minimize total completion time with release times and deadlines. In submission.

Du, J.Z., & Leung, J.Y.T. (1993). Minimizing mean flow time with release time and deadline constraints. Journal of Algorithms, 14, 45-68.

Federgruen, A., & Groenevelt, H. (1986). Preemptive scheduling of uniform machines by ordinary network flow techniques. Management Science, 32, 341-349.

Garey, M.R., & Johnson, D.S. (1979). Computers and Intractability: A guide to the theory of NP-completeness. San Francisco: Freeman.

Graham, R.L., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, 5, 287-326.

Hariri, A.M.A., Potts, C.N., & Van Wassenhove, L.N. (1995). Single machine scheduling to minimize total weighted late work. ORSA Journal on Computing, 7, 232-242.

Horn, W.A. (1974). Some simple scheduling algorithms. Naval Research Logistics Quarterly, 21, 177-185.

He, C., Lin, H., Lin, Y.X., & Dou, J.M. (2014). Minimizing total completion time for preemptive scheduling with release dates and deadline constraints. Foundations of Computing and Decision Sciences, 39, 17-26.

Jackson, J.R. (1955). Scheduling a production line to minimize maximum tardiness. Technical report, University of California, Los Angeles.

Kethley, R.B., & Alidaee, B. (2002). Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. Computers & Industrial Engineering, 43, 509-528.

Koulamas, C., & Kyparisis, G.J. (2001). Single machine scheduling with release times, deadlines and tardiness objectives. European Journal of Operational Research, 133, 447-453.

Kovalyov, M., Potts, C.N., & Van Wassenhove, L.N. (1994). A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work. Mathematics of Operations Research, 19, 86-93.

Kuhn, H.W. (2005). The Hungarian method for the assignment problem. Naval Research Logistics, 52, 7-21.

Labetoulle, J., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1984). Preemptive scheduling of uniform machines subject to release dates. In Progress in combinatorial optimization, Academic Press, pp. 245-261.

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

Leung, J.Y.T., Vincent, K.M., & Wei, W.D. (1994). Minimizing the weighted number of tardy task units. Discrete Applied Mathematics, 51, 307-316.

Leung, J.Y.T. (2004). Minimizing total weighted error for imprecise computation tasks and related problems. In: Leung, J. Y. T., editor. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Boca Raton: CRC Press; p. 34.1-16.

Lin, B.M., & Hsu, S.W. (2005). Minimizing total late work on a single machine with release and due dates. In SIAM conference on computational science and engineering, Orlando.

Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. Asia-Pacific Journal of Operational Research, 18, 193-242.

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: a review and discussion. Computers & Industrial Engineering, 104, 224-245.

Orlin, J.B. (1993). A faster strongly polynomial minimum cost flow algorithm. Operations Research, 41, 338-350.

Pan, Y.P. (2003). An improved branch and bound algorithm for single machine scheduling with deadlines to minimize total weighted completion time. Operations Research Letters, 31, 492-496

Posner, M.E. (1985). Minimizing weighted completion times with deadlines. Operations Research, 33, 562-574.

Potts, C.N., & Van Wassenhove, L.N. (1983). An algorithm for single machine sequencing with deadlines to minimize total weighted completion time. European Journal of Operational Research, 12, 379-387.

Potts, C.N., & Van Wassenhove, L.N. (1992a). Single machine scheduling to minimize total late work. Operations Research, 40, 586-595.

Potts, C.N., & Van Wassenhove, L.N. (1992b). Approximation algorithms for scheduling a single machine to minimize total late work. Operations Research Letters, 11, 261-266.

Ren, J.F., Zhang, Y.Z., & Sun, G. (2009). The NP-hardness of minimizing the total late work on an unbounded batch machine. Asia-Pacific Journal of Operational Research, 26, 351-363.

Ren, J.F., Du, D.L., & Xu, D.C. (2013). The complexity of two supply chain scheduling problems. Information Processing Letters, 113, 609-612.

Shan, F., Luo, J.Z., & Shen, X.J. (2014). Optimal energy efficient packet scheduling with arbitrary individual deadline guarantee. Computer Networks, 75, 351-366.

Smith, W.E. (1956). Various optimizers for single-stage production. Naval Research Logistics Quarterly, 3, 59-66.

Sterna, M. (2006). Late work minimization in a small manufacturing system. Technical Report, RA-02/06. Poznan: Institute of Computing Science, Poznan University of Technology.

Sterna, M. (2007). Late work minimization in a small flexible manufacturing system. Computers & Industrial Engineering, 52, 210-228.

Sterna, M. (2011). A survey of scheduling problems with late work criteria. Omega, 39, 120-129.

Wan, L., Yuan, J.J., & Geng, Z.C. (2015). A note on the preemptive scheduling to minimize total completion time with release time and deadline constraints. Journal of Scheduling, 18, 315-323.

Wang, D.J., Kang, C.C., Shiau, Y.R., Wu, C.C., & Hsu, P.H. (2017). A two-agent single-machine scheduling problem with late work criteria. Soft Computing, 21, 2015-2033.

Werner, F. (1993). A branch and bound algorithm for minimizing weighted completion times with deadlines. Optimization, 28, 187-199.

Wu, C.C., Yin, Y.Q., Wu, W.H., Chen, H.M., & Cheng, S.R. (2016). Using a branch-and-bound and a genetic algorithm for a single-machine total late work scheduling problem. Soft Computing, 20, 1329-1339.

Xu, Z.Z., Zou, Y.X., & Kong, X.J. (2015). Meta-heuristic algorithms for parallel identical machines scheduling problem with weighted late work criterion and common due date. Springer Plus, 4, 782.

Yuan, J.J. (2017). Unary NP-hardness of minimizing the number of tardy jobs with deadlines. Journal of Scheduling, 20, 211-218.

Zafer, M.A., & Modiano, E. (2009). A calculus approach to energy-efficient data transmission with quality-of-service constraints. IEEE/ACM Transactions on Networking (TON), 17, 898-911.

Zhang, X.G., & Wang, Y. (2017). Two-agent scheduling problems on a single-machine to minimize the total weighted late work. Journal of Combinatorial Optimization, 33, 945-955.

Zhang, Y., & Yuan, J.J. (2019). A note on a two-agent scheduling problem related to the total weighted late work. Journal of Combinatorial Optimization, 37, 989-999.