ORIGINAL MANUSCRIPT

Minimizing Total Earliness and Tardiness with Periodically Supplied Non-renewable Resource Profiles

Mohammad Namakshenas[a], Aleida Braaksma[b] and Mohammad Mahdavi Mazdeh[a]

[a]School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[b]Center for Healthcare Operations Improvement and Research (CHOIR), University of Twente, Enschede, The Netherlands

ARTICLE HISTORY
Compiled February 21, 2021

ABSTRACT
We consider a special class of resource-constrained single machine scheduling problems. In the classical scheduling context, resource types are classified into renewable and non-renewable; however, a large variety of real-world problems may not fit into one of these classes, e.g., labor regulations in project scheduling, budget allocation to different phases of a construction project, and dose management in a medical imaging center. In this study, we address a class of non-renewable resources supplied, not necessarily immediately, in different periods of the planning horizon. The objective is to assign the jobs to the supply periods and schedule them such that the sum of total tardiness and total earliness is minimized. Several properties and complexity results of the optimal schedules are discussed, then they are used to develop a tractable algorithm. First, we decompose the problem into several single supply problems and then decide the optimal schedule through a polynomial-time optimal algorithm for each single supply problem. The scalability tests indicate the promising performance guarantee of the algorithm compared to provably optimal schedules in the integrated framework.

KEYWORDS
Scheduling; Single machine; Total Earliness and Tardiness; Non-renewable resource

1. Introduction

In most real-life environments, jobs require, besides machines, certain additional resources. Additional resources are classified into renewable (e.g., operators) and non-renewable (e.g., money or energy) resources. In practice, there exist additional resources that do not function with respect to one of these classes, and they can be conceived as a generalization of both renewable and non-renewable resources. We study a type of recourses that behaves like a renewable resource during the planning horizon; however, it acts as a limited non-renewable resource in each period k of the planning horizon. In this study, we map the features of dose management of radiopharmaceuticals through the decay and growth process of an isotope, say Technetium-99m, in a medical imaging center. The process happens inside a generator that produces the isotope daily and becomes inoperable after a week (Andersson and Mattsson 2016). Financial resource-constrained problems are another use-case of this type of resource. Suppose a contractor

CONTACT Mohammad Mahdavi Mazdeh. Email: mazdeh@iust.ac.ir

is constructing a power plant with an agreed budget, which is partially fulfilled by the government at the beginning of each construction phase. Each phase can start if the agreed budget is dedicated in due course (Gafarov et al. 2011).

1.1. *Outline*

In this study, we consider a single machine scheduling problem with a periodically supplied non-renewable resource. The formal description of the problem is as follows. Consider a set $J = \{1, \ldots, n\}$ of independent jobs that must be processed on a single machine. All jobs, $\forall j \in J$, have equal processing time, p. All jobs are ready at time zero and each job has a dedicated due date d_j. A job cannot be preempted once started. Job j requires an amount u_j of a non-renewable resource. There is only one type of resource. The supply periods of the resource, $\{t_k \mid k = 1, \ldots, q\}$, are known in advance; however, the resource is only available for l_k time units after its supply time with the maximum amount v_k such that $\sum_{j \in J} u_j \leq \sum_{k \in K} v_k$. The job j should not start if the remaining available resource in supply period k is less than u_j. Jobs can start at the end of interval $[t_k, t_k + l_k]$. The objective is to minimize the total tardiness and total earliness of jobs. According to the 3-field notation (Pinedo 2016), we identify the problem as $1 \mid p_j = p, RP(NR, t_k, l_k, v_k) \mid \sum_j E_j + \sum_j T_j$. We characterize the *resource profile*, RP, by the resource non-renewability, NR, the known supply starts, t_k, the supply interval at supply period k, l_k, and the maximum supply of the resource at supply period k, v_k. Note that $1 \mid p_j = p \mid \sum_j E_j + \sum_j T_j$ is polynomially solvable (Garey et al. 1988); however, we prove that $1 \mid p_j = p, RP(NR, t_k, l_k, v_k) \mid \sum_j E_j + \sum_j T_j$ is NP-hard.

This problem is motivated by a practical application. The problem is originally adopted from the preparation of patients for scanning at a medical imaging center. There is only one technician who is responsible for injecting the patients with the pharmaceutical. Note that the pharmaceutical is time-sensitive and decays in a short period. The technician can only inject one patient at a time. Then, the patients should wait to get the pharmaceutical to be absorbed in the target organ, and immediately after that, the scan should start. The ideal time to start the scan equals the ideal time to finish the injection (due date) plus the waiting time. Any deviation from this due date is not favorable. If the patient is scanned too early, then the scanned image will be too noisy, and the analysis of the image will be impossible. If the patient is scanned too late, again, it will result in poor image quality. Hence, the objective is to minimize any deviation from the due date (consider a just-in-time system). For more information about the details of this application, see for instance Andersson and Mattsson (2016).

The research area of scheduling problems covering both (periodically supplied) non-renewable resource allocation and the earliness and tardiness objective is rather limited. To the best of the authors' knowledge, no research has addressed both contexts simultaneously. In what follows, we sketch the literature that is closest to our work. Gafarov et al. (2011) developed an interesting classification on some single machine scheduling problems with financial resource constraints. To illustrate the notion of these types of resources as a non-renewable resource, consider a funding problem for a multi-phase project. Each phase cannot begin until it receives the minimum negotiated fund. The jobs should be scheduled such that they do not incur out-of-funding issues. The problem assumes that there is no gap between two resource supplies and supply times are fixed and thus known beforehand. Gafarov et al. (2011) proved some complexity results and dominance properties for several problems with standard

objective functions, e.g., the minimization of makespan, total tardiness, number of tardy jobs, total completion time, and maximum lateness. However, they did not provide information on how one can develop polynomial approximation schemes for NP-hard cases.

Györgyi and Kis (2014) developed different algorithms for minimizing the makespan on a single machine with periodically supplied non-renewable resources. When the resource did not suffice for processing the next job, the resource was replenished immediately. This problem was solved under two distinct assumptions. Györgyi and Kis (2014) approached the problem when the number of replenishments is not part of the input (variable supply events) and when the problem has a constant number of replenishments (fixed supply events). They showed that there exists a pseudo-polynomial time algorithm for the variable supply events case. They also proved that the problem with fixed supply events is harder to solve than the variable counterpart.

Another extension of the problem is resource recovery constraints (Vallikavungal Devassia et al. 2018) where resources are available in batches, and recovery times are required between each batch. These types of resources are available in limited quantities but are recovered to process all jobs in different batches. An example of such a resource is the working time of a worker, which is smoothly recovered every time when he or she finishes a task, but the total working hours are restricted on the planning horizon. For the recoverable applications of non-renewable resources, see for instance Böttcher et al. (1999) and Naber and Kolisch (2014).

The problem excluding the resource constraints, i.e., $1 \mid p_j = p \mid \sum_j E_j + \sum_j T_j$, was intensively investigated by Garey et al. (1988); they developed a time-tabling scheme based on the tree-based heap data structure to transform a given sequence into a minimum cost schedule in $O(n \log(n))$. In general, problem $1 \mid\mid \sum_j E_j + \sum_j T_j$ with each job having a distinct processing time is strongly NP-hard (Wan and Yuan 2013). This problem has an additional level of complexity compared to tardiness minimization problems. One level concerns finding an optimal permutation (sequence) of jobs, and the other concerns computing idle times between any pair of jobs (or the completion times of jobs). This problem has historically gained extensive interest; for more information please see researches of dominance properties (Davis and Kanet 1993; Szwarc and Mukhopadhyay 1995), lower bounds (Kim and Yano 1994), exact and approximation schemes (Kim and Yano 1994; Verma and Dessouky 1998), and heuristic algorithms (Sourd and Kedad-Sidhoum 2003; Sourd 2005; Hendel and Sourd 2007; Wodecki 2009).

Prior to summarizing our results, we first distinguish two insights of any feasible solution. First, the starting time of any job cannot reside outside of its dedicated supply period. Second, the total resource requirements for any assignment of a set of jobs to supply period k cannot be larger than the available resource amount. In this spirit, we construct any feasible and optimal schedule hierarchically, where each step optimizes the schedule input from the previous step. More specifically, the main contributions of this paper are summarized as follows:

(1) We develop a tractable optimal scheme for $1 \mid p_j = p, RP(NR, t_k, l_k, v_k) \mid \sum_j T_j + \sum_j E_j$.
(2) We investigate the structural properties of the problem and prove that the problem is indeed NP-hard by a reduction from the Partition problem.
(3) We provide a decomposition procedure in which jobs are first assigned to different supply periods based on the resource consumption. Then we develop and prove a polynomial-time algorithm to convert an optimal sequence, the permutation of jobs, into an optimal schedule, the timing of jobs, for each separate supply

period.

(4) We provide a fast optimal algorithm for the scheduling of a single machine with the objective of minimizing the total earliness and tardiness considering deadlines on the start times of jobs.
(5) We prove an efficient lower bound for the hard instances in which the last job of supply period k overlaps with the first job of supply period $k+1$.

The remainder of this paper is organized in the following way. The remainder of this section overviews the parameters and decision variables that appeared throughout the study. In Section 2, we first develop an integer programming model and discuss the complexity status and a possible approach to develop a tractable algorithm. Section 3 decomposes the original problem into single supply problems and develops an optimal scheme to schedule jobs in separate supply periods polynomially. Section 4 integrates the proposed schemes into a single algorithm. Section 5 examines the computational competence of the developed scheme and discusses the algorithm's scalability. The last section concludes the study with a summary and suggestions for future research.

1.2. *Notations*

We use the following notational framework throughout the paper. For notational simplicity, objective $\sum_j E_j + \sum_j T_j$ and resource profile $RP(NR, t_k, l_k, v_k)$ are abbreviated as `ET` and `RP`, respectively. Note that the start times of the jobs assigned to supply period k should be within interval $[t_k, t_k + l_k]$. Hence, we denote the *single supply problem* as $1 \mid p_j = p, s_j \in [t_k, t_k + l_k] \mid$ `ET`. We call a sub-sequence of immediately consecutive jobs (i.e., without any idle time) inside each single supply problem a *chain*. We also provide an exhaustive list of the notation as follows.

Constants

n	Number of jobs
b	Number of chains
q	Number of supply periods
p	Processing time of a job
M	Numerically large number

Sets

J	Set of all jobs, $j \in J = \{1, \ldots, n\}$
K	Set of supply periods, $k \in K = \{1, \ldots, q\}$
C	Set of chains, $c \in C = \{1, \ldots, b\}$
π_c	Permutation set of chain c; $\pi_c = \langle 1, \ldots, n_c \rangle$

Parameters

d_j	Due date of job j
l_k	Length of supply interval k
u_j	Resource consumption of job j
t_k	Start time of supply period k
v_k	Amount of resource available at the start of supply period k
n_c	Number of jobs inside chain c

Decision Variables

y_{ij}	1 if job i precedes job j in the schedule (not necessarily immediately); 0 otherwise.
x_{jk}	1 if job j is allocated to the kth supply period; 0 otherwise.
s_j	Start time of job j

c_j	Completion time of job j
E_j	Earliness of job j
T_j	Tardiness of job j

2. Structural Properties

2.1. *Mathematical Model*

In this section, we develop an integer programming model based on the definition of the problem. We also investigate the complexity status of the problem.

Example 2.1. Eight jobs are to be processed on a single machine considering two supply periods. Suppose that $l_1 = 11$, $l_2 = 17$, $v_1 = 28$, $v_2 = 35$, $t_1 = 1$, $t_2 = 15$, and $p = 3$. The due date and resource consumption of each job are given in Table 1. Sequences $\langle 3, 8, 1\rangle$ and $\langle 2, 4, 6, 5, 7\rangle$ minimize the objective, $\texttt{ET} = 8$, according to the

Table 1. Due dates and resource consumptions of jobs in Example 2.1

j	1	2	3	4	5	6	7	8
d_j	10	16	7	20	30	28	32	8
u_j	10	9	7	5	7	6	6	8

first and the second supply periods, respectively. The optimal sequences are shown in Figure 1. In the optimal schedule, jobs 3 and 6 are early, and jobs 1, 2, 4, and 7 are late, and there is an idle period between the completion of job 4 and the start time of job 6.

Objective ET is one of the rare cases in which the optimal schedule favors idle times, as shown in Example 2.1. This complicates the finding of an optimal schedule. Mathematical model P(1) exploits the *disjunctive variables*, y_{ij}, (Baker and Keller 2010) to force precedence relations among jobs and the *assignment variables*, x_{jk}, to allocate each job to a supply period.

$$\text{P(1): minimize} \quad \sum_{j\in J} E_j + \sum_{j\in J} T_j \tag{1}$$

$$\text{subject to} \quad T_j \geq s_j + p - d_j, \quad \forall j \in J, \tag{2}$$

$$E_j \geq d_j - (s_j + p), \quad \forall j \in J, \tag{3}$$

$$s_i + p \leq s_j + M(1 - y_{ij}), \quad \forall i, j \in J \,:\, i \neq j, \tag{4}$$

$$y_{ij} + y_{ji} = 1, \quad \forall i, j \in J \,:\, i \neq j, \tag{5}$$

$$s_j \geq t_k x_{jk}, \quad \forall j \in J, \forall k \in K, \tag{6}$$

$$s_j \leq (t_k + l_k) x_{jk}, \quad \forall j \in J, \forall k \in K, \tag{7}$$

$$\sum_{j\in J} u_j x_{jk} \leq v_k, \quad \forall k \in K, \tag{8}$$

$$\sum_{k\in K} x_{jk} = 1, \quad \forall j \in J, \tag{9}$$

$$x \in \{0,1\}^{n\times q},\ y \in \{0,1\}^{n\times n}, s \in \mathbb{R}^n_+,\ T, E \in \mathbb{R}^n_+. \tag{10}$$

The constraint sets (2) and (3) are the linear definitions of tardiness, $\max(s_j + p - d_j, 0)$, and earliness, $\max(d_j - s_j - p, 0)$. The constraint sets (4) and (5) guarantee

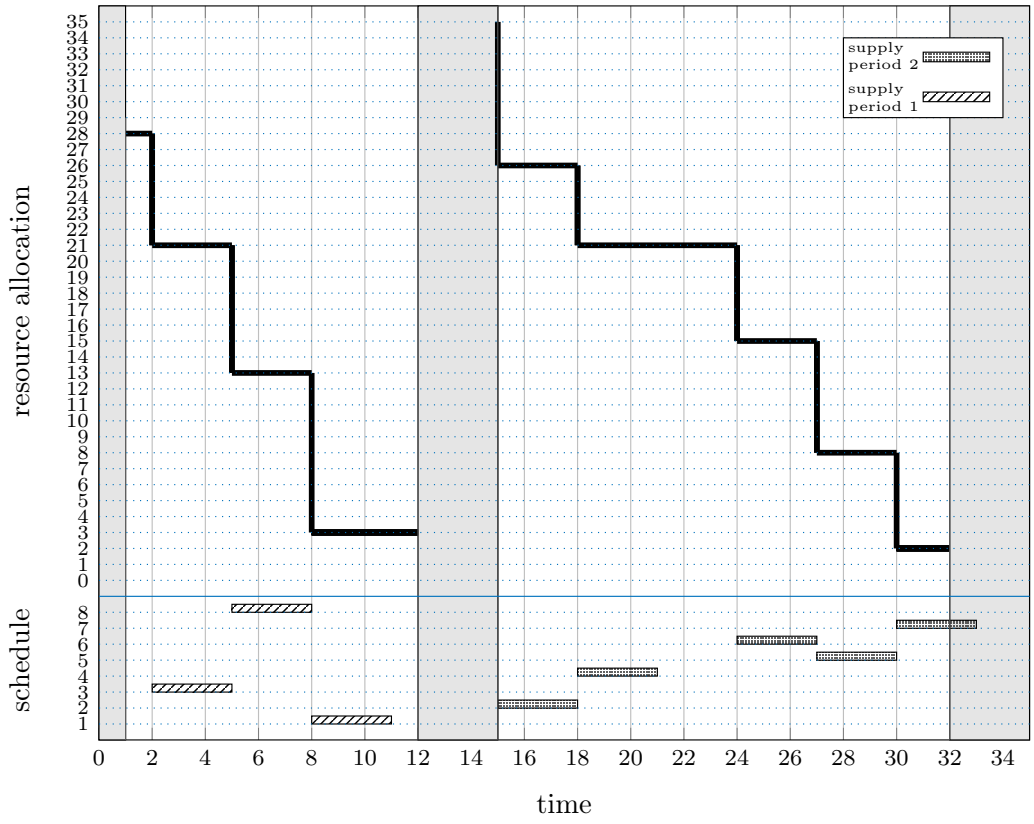

Figure 1. Optimal Gant chart regarding resource profile. (The horizontal boxes indicate the processing of jobs. The white areas are supply intervals, so jobs cannot start in the shaded areas. The vertical lines represent the resource consumption of each job.)

that the start times of any two adjacent jobs i and j are at least p time units apart. The constraint sets (6) and (7) ensure that each job starts within a supply interval. The constraint set (8) ensures that the total allocated resource consumption during a supply period does not exceed the available amount. The constraint set (9) makes sure that each job is allocated to one supply period.

In Section 5, we will examine the scalability issues of integer model P(1). There are $2n^2$ disjunctive constraints (4 and 5) and $q + n$ assignment constraints (8 and 9). Intuitively, the decision space of P(1) does not reflect any computational challenges. However, in Section 5, we will challenge this intuition by solving instances of the problem with a job number larger than a specific threshold and a tight v_k w.r.t u_j and a limited dispersion of due dates. It is possible to restrict the start times of the jobs based on their due dates. In the following proposition, we prove a dominance rule to limit each job's time slots to a tighter interval.

Proposition 2.2. *There exists an optimal schedule for problem* $1 \mid p_j = p, s_j \in [t_k, t_k + l_k] \mid ET$ *in which job* j *is started within time interval* $\left[\max\left\{t_k, d_j - (n-1)(2p-\epsilon) - p\right\}, \min\left\{t_k + l_k, d_j + (n-1)(2p-\epsilon)\right\}\right]$ *where* ϵ *is a sufficiently small positive number.*

Proof. Consider a schedule in which job j is tardy or early. We first prove the case in which job j is completed before d_j (early). Hence, it should start and finish in interval $[0, d_j]$, ignoring the supply period's start and end. Therefore, interval $[0, d_j]$ should at

least be of length $(n-1)(2p-\epsilon)+p$. This length includes $p(n-1)$ jobs processed after job j, at most $(p-\epsilon)(n-1)$ idle times, and p units for processing job j. Hence, the start time of job j without possible improvement in the objective value is no less than $d_j-(n-1)(2p-\epsilon)-p$. The case in which job j is processed after d_j is completely analogous. The result follows. □

2.2. *Complexity Status*

Definition 2.3 (Partition problem)**.** An integer set $N=\{b_1,b_2,\ldots,b_n\}$ of numbers is given such that $b_1\geq b_2\geq\ldots\geq b_n>0$ and $b_i\in\mathbb{Z}_+$, $i=1,2,\ldots,n$. Does there exist a subset $N'\subseteq N$ such that $\sum_{i\in N'}b_i=\frac{1}{2}\sum_{i\in N}b_i$?

Lemma 2.4. *Problem* $1\mid p_j=p,\mathtt{RP}\mid\mathtt{ET}$ *is NP-hard.*

Proof. The proof is done by reduction from the Partition problem. Consider an instance of the problem with the following setting. Let $p=0$, $u_j=b_j$, $t_1=0$, $t_2=l_1=n(\sum_{j\in J}b_j)$, $l_2=n^2(\sum_{j\in J}b_j)$, $d_j=l_1-u_j$, and $v_1=v_2=\frac{1}{2}\sum_{j\in J}b_j$. It is immediate that all jobs are to be scheduled in the supply period $[0,l_1]$ or $[l_1,l_1+l_2]$. Assume the partial schedules π_1 and π_2 correspond to the schedules in these intervals, respectively. Then the optimal schedule should assign the maximum number of jobs to π_1 w.r.t. v_1. It is readily deduced that the optimal schedule has the following property: $s_{j\in\pi_1}=d_j$, $T_{j\in\pi_1}=0$, $E_{j\in\pi_1}=0$, $s_{j\in\pi_2}=l_1$, $T_{j\in\pi_2}=l_1-d_j=u_j$, and $E_{j\in\pi_2}=0$. Then the instance of the partition problem has an answer "YES" if and only if in an optimal schedule $\pi=\langle\pi_1,\pi_2\rangle$ we have $\mathtt{ET}=\frac{1}{2}\sum_{j\in J}b_j$. □

3. Single Supply Problem

We postulate that for problem $1\mid p_j=p,\mathtt{RP}\mid\mathtt{ET}$, choosing d_j-p for s_j and then making schematic corrections on s_j w.r.t. t_k should produce the optimal schedule. We first decompose the original problem into q single supply problems. Next, we decide the optimal sequences and schedules of each decomposed problem. Finally, in the next section, the solutions of the decomposed problems are coordinated to produce the global optimum.

3.1. *Assignment to Supply Periods*

We translate the resource allocation part of the original problem to the generalized assignment problem (Cohen et al. 2006), P(2). In general, problem P(2) assigns jobs to supply intervals based on the deviations of the due dates from the supply starts and

ends.

$$\text{P(2): minimize} \quad \sum_{j\in J} \max\left\{\sum_{k\in K} x_{jk}(t_k-(d_j-p)), \sum_{k\in K} x_{jk}(d_j-p-(t_k+l_k))\right\} \tag{11}$$

$$\text{subject to} \quad \sum_{k\in K} x_{jk}=1, \quad \forall j\in J, \tag{12}$$

$$\sum_{j\in J} u_j x_{jk} \le v_k, \quad \forall k\in K, \tag{13}$$

$$x\in\{0,1\}^{n\times q}. \tag{14}$$

The constraint sets in P(2) are equivalent to constraint sets (8) and (9) introduced in integer model P(1). These constraints guarantee feasible assignments for each supply period. Note that $s_j \leftarrow d_j - p$ is the ideal start time of job j. Hence, job j should ideally be assigned to supply interval k for which $t_k \le d_j - p \le t_k + l_k$. If there is no such interval, the ideal allocation minimizes the distance of $d_j - p$ from interval boundaries $[t_k, t_k + l_k]$. This is what objective (11) does.

3.2. *Schedule-correcting Scheme*

The other concern is to decide the jobs' ordering inside each supply period. In what follows, we show that the Earliest Due Date (EDD) rule, in which the job with the earliest due date is scheduled first, produces the optimal sequence.

Lemma 3.1. *There exists an optimal schedule for problem* $1 \mid p_j = p \mid$ *ET, in which the jobs are ordered according to the EDD rule.*

Proof. Proof by contradiction. Suppose partial sequence $\pi' = \langle j, i\rangle$ such that $d_i \le d_j$, which is not ordered by EDD, is optimal. Let the start times of jobs i and j w.r.t. π' be s'_i and s'_j, respectively. We apply a pairwise exchange of jobs j and i and denote the new schedule $\pi \leftarrow \langle i, j\rangle$. It is verified that $\mathtt{ET}(\pi) = \mathtt{ET}(\pi')$ for cases $s'_j < s'_i \le d_i < d_j$ and $d_i < d_j \le s'_j < s'_i$ after this exchange. On the other hand, $\mathtt{ET}(\pi) < \mathtt{ET}(\pi')$ for cases $d_i \le s'_j < s'_i \le d_j$, $d_i \le s'_j < d_j \le s'_i$, $s'_j < d_i \le s'_i \le d_j$, and $s'_j \le d_i < d_j \le s'_i$. The outcome of these six exchange cases contradicts the optimality of π' and completes the proof. □

Next, we develop a polynomial algorithm to schedule the jobs based on the inputs of the assignment variable, x_{jk}.

Lemma 3.2 (Szwarc and Mukhopadhyay 1995)**.** *Given partial schedule* $\pi = \langle i, j\rangle$*, if* $d_j - d_i \le p$*, then there is no idle time between jobs* i *and* j*.*

Definition 3.3 (Chain)**.** We call sequence π a chain if there is no idle time between any two consecutive jobs in π.

Next, we prove the necessary and sufficient conditions for the objective of total earliness plus total tardiness. The basic idea is to decide the start time of a job and then to compute the start times of other jobs inside a chain, recursively. The following lemma generates a semi-active schedule, a schedule in which no shift that preserves the sequence of the jobs produces a better schedule, for a chain.

Lemma 3.4. *Given a chain π_c ordered by* EDD*, there exists a semi-active schedule for π_c if*

$$\begin{aligned} s_{1\in\pi_c} &\leftarrow \min\left\{\max\left\{t_k, (\gamma_1+\gamma_2)/2\right\}, t_k + l_k - (n_c-1)p\right\}, \\ \gamma_1 &\leftarrow d_{\lfloor (n_c+1)/2 \rfloor} - \lfloor (n_c+1)/2 \rfloor\, p, \\ \gamma_2 &\leftarrow d_{\lceil (n_c+1)/2 \rceil} - \lceil (n_c+1)/2 \rceil\, p. \end{aligned} \tag{15}$$

Proof. We redefine ET based on any job, e.g., job 1, and then we calculate the start times of jobs based on job 1, $s_j \leftarrow s_1 + (j-1)p,\ \forall j \in \pi_c \setminus \{1\}$.

$$\begin{aligned} \texttt{ET} &= \sum_{j\in\pi_c} \left| s_j + p - d_j \right| \\ &= \sum_{j\in\pi_c} \left| s_1 - (d_j - jp) \right|. \end{aligned}$$

Objective ET is a 1-norm function with a single variable, i.e. s_1, and the median of set $\left\{d_j - jp,\ j \in \pi_c\right\}$ minimizes ET (Schwertman et al. 1990). There exists a semi-active schedule for chain π_c if $s_{1\in\pi_c} \leftarrow \text{med}\left\{d_j - jp,\ j \in \pi_c\right\}$ where med$\{A\}$ is the median of set A. It directly follows that set $\left\{d_j - jp,\ j \in \pi_c\right\}$ is non-increasing since π_c preserves both the EDD rule and Lemma 3.2. Hence, whether n_c is an odd or even number, the median is $\left(\left(d_{\lceil n' \rceil} - \lceil n' \rceil p\right) + \left(d_{\lfloor n' \rfloor} - \lfloor n' \rfloor p\right)\right)/2$ where $n' = (n_c+1)/2$. Since the start times of all jobs inside π_c should reside in $[t_k, t_k + l_k]$, the median cannot be earlier than t_k or later than $t_k + l_k$. □

Using the following proposition, we can easily address the case, with fewer iterations, where some jobs of a chain overlap with some jobs of another chain, recursively.

Proposition 3.5. *Consider two chains π_1 and π_2 to which Lemma 3.4 is applied separately and a subset of the jobs of π_1 overlaps with those of π_2. There exists a semi-active schedule if the two chains form a new chain $\pi = \langle \pi_1, \pi_2 \rangle$ and Lemma 3.4 is applied to π.*

Algorithm 1 produces a semi-active schedule for each supply period based on the results of Lemmas 3.1 through 3.4 and Proposition 3.5. Algorithm 1 consists of two primary routines, given the number of jobs dedicated to supply period k, i.e., $\sum_j x^*_{jk}$ by solving P(2). The first routine, steps 1 through 3, groups the jobs into chains and computes the start times of the first and the last jobs in each chain. The second routine, steps 4 and 5, checks whether the current chain is conflicting with the previous or the next chain. If such a case happens, it will join two conflicting chains and compute the start time of the first job in the emerging chain. The routine continues until there are no conflicting chains in the entire schedule. According to Proposition 3.5, the final schedule is semi-active. The detailed version of Algorithm 1 is provided in Appendix A.

Proposition 3.6. *Algorithm 1 generates the optimal schedule for problem $1 \mid p_j = p, s_j \in [t_k, t_k + l_k] \mid$ ET in $O\left(\sum_j x^*_{jk}\right)$ iterations.*

Proof. According to Proposition 3.5, it is immediate that the algorithm generates a

Algorithm 1: Recursive scheme to produce optimal schedule for $1 \mid p_j = p, s_j \in [t_k, t_k + l_k] \mid \texttt{ET}$

Input: number of jobs in supply period k, $\left(\sum_j x^*_{jk}\right)$, processing time ($p$), due dates ($d_j$)
Output: Optimal start times of jobs in supply period k
step 1. Sequence jobs according to EDD rule;
step 2. Create set of chains by grouping jobs j and $j+1$ if $d_{j+1} - d_j \leq p$;
step 3. For set of chains, compute the start times of first jobs in each chain according to Lemma 3.4;
step 4. If there is no overlapping chain, compute the start times of remaining jobs in the chains, i.e., start time of job $j : j \neq 1$ in the chain equals the start time of the first job in that chain plus $(j-1)p$, and then STOP; Otherwise go the next step;
step 5. For the set of chains, join two overlapping chains if the current chain overlaps with the previous or the next chain. Join three overlapping chains if the current chain overlaps with the previous and the next chain. Then update the set of chains and go to step 3.

semi-active schedule. Davis and Kanet (1993) proved that every semi-active schedule is optimal for $1 \mid\mid \texttt{ET}$. The first primary routine, steps 1 to 3, operates in $O\left(\sum_j x^*_{jk}\right)$ iterations. The worst-case running time of the second primary routine, steps 4 and 5, is $b-1$ iterations (consider a case where the leftmost chain recursively overlaps with the next chain). The results follow directly. □

Example 3.7. Given the data in Example 2.1, we first assign the jobs to the supply periods by solving P(2). The optimal assignment is $x_{31} = x_{81} = x_{11} = 1$ and $x_{22} = x_{42} = x_{62} = x_{52} = x_{72} = 1$. Applying the EDD rule and Lemma 3.2, we obtain chain $\pi^{(1)}_1 = \langle 3, 8, 1\rangle$ for the first supply period and also chains $\pi^{(2)}_1 = \langle 2\rangle$, $\pi^{(2)}_2 = \langle 4\rangle$, and $\pi^{(2)}_3 = \langle 6, 5, 7\rangle$ for the second supply period. Next, we apply Lemma 3.4 and the start times are $s_3 = 2$, $s_8 = 5$, $s_1 = 8$, $s_2 = 15$, $s_4 = 17$, $s_6 = 24$, $s_5 = 27$, and $s_7 = 30$. Since jobs 2 and 4 overlap, we join the first two chains in the second supply period and then we apply Lemma 3.4 according to Algorithm 1. Finally, we have the revised start time of job 4 as $s_4 = 18$.

4. Integration of Schemes

At last, we summarize the complete procedure as follows. We abbreviate the complete procedure as ACFS: Assign, Chain, and Fix Scheme. The ACFS first assigns jobs to supply periods by solving problem P(2). Then Algorithm 1 orders jobs based on the EDD rule and iteratively fixes the schedule by shifting overlapping jobs in the correct directions. The computational difficulty of the ACFS increases when the number of jobs increases, and on the other hand, the input size of $(\sum_{k\in K} v_k - \sum_{j\in J} u_j)$ decreases. For computational convenience, we solve the assignment part, P(2), with an external optimization solver. Hence, the complexity of the ACFS is divided into two parts, say the generalized assignment problem as P(2) and Algorithm 1. The only remaining concern is that the last job of supply period k may overlap with the first job of supply

period $k+1$ after applying the ACFS. This exception and the complexity analysis is discussed in the following proposition. Moreover, the complete ACFS procedure is illustrated in Figure 2.

Proposition 4.1. *The ACFS produces an optimal schedule if* $t_{k+1} - t_k - l_k \geq p,\ \forall k \in K \setminus \{q\}$. *If* $t_{k+1} - t_k - l_k < p,\ \exists k \in K \setminus \{q\}$, *then the ACFS produces an lower bound such that*

$$\texttt{ET}\big(P(1)\big) - \texttt{ET}(ACFS) \leq \sum_{k=2}^{q} \delta_k \sum_{k'=1}^{k-1} \max\left\{0, p - t_{k'+1} + t_{k'} + l_{k'}\right\},$$

where $\texttt{ET}(ACFS)$ *and* $\texttt{ET}\big(P(1)\big)$ *evaluate objective* ET *for the ACFS and P(1), respectively, and* $\delta_k = \max\left\{ \left|J'_k\right| \left| \sum_{j' \in J'_k} u_{j'} \leq v_k,\ \left|J'_k\right| \leq \left\lfloor l_k/p \right\rfloor + 1 \right.\right\}$. *If the running time of problem P(2) is* $O\big(f(n)\big)$, *then the ACFS runs in* $O\left(f(n) + \max_{k \in K} \delta_k\right)$.

According to Proposition 4.1, in case $t_{k+1} - t_k - l_k < p,\ \exists k \in K \setminus \{q\}$, the ACFS may confuse to assign jobs to the right supply period based on $u_j,\ j \in J$ due to the small gaps between the end of the current supply period and the start of the next supply period, thus, it produces a lower bound.

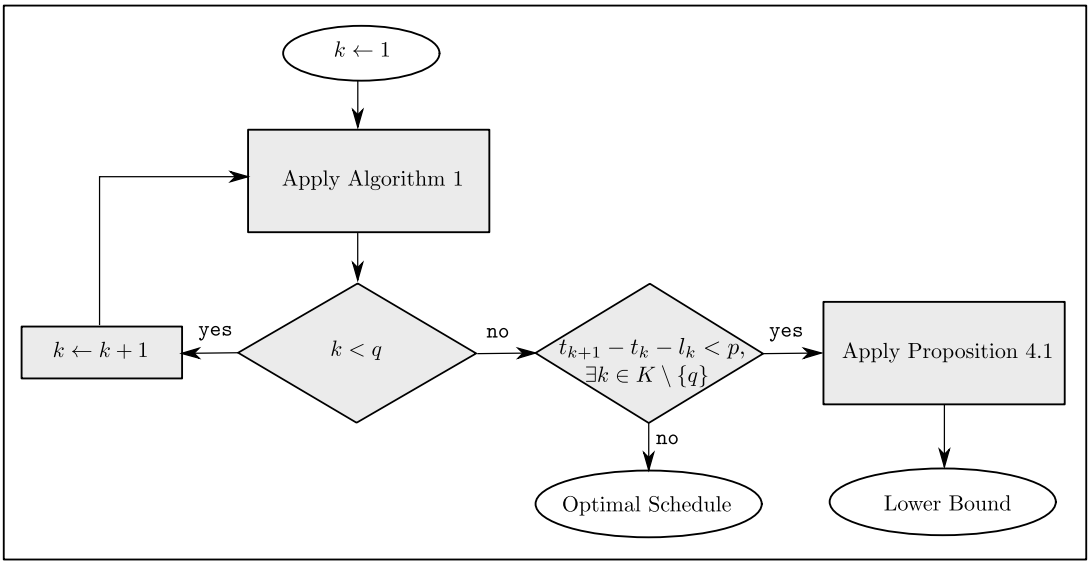

Figure 2. Flowchart for `ACFS` procedure

5. Computational Experiment

In order to evaluate the scalability and overall running time of the ACFS, we ran several computational tests based on pseudo-random instances of the problem. We implemented the mathematical model P(1) and the ACFS in the Python 3.7 platform including Gurobi 8.5 64-bit as the optimization engine. The test problems were run on a Windows 10 64-bit machine with a core i7 2.66 GHz CPU and 12 GB memory. In this spirit, we make use of the discrete uniform distribution $\mathcal{U}(a, b)$ to generate

Table 2. Instance generation regime

$p \sim \mathcal{U}(2,4)$,
$l_k \sim \mathcal{U}(10,20),\ \forall k \in K$,
$v_k \sim \mathcal{U}(10,100),\ \forall k \in K$,
$u_j \sim \mathcal{U}(1, \max_{k\in K} v_k - \xi_1),\ \forall j \in J\ : \sum_{j\in J} u_j + \xi_2 \leq \sum_{k\in K} v_k$,
$t_1 \sim \mathcal{U}(0, l_1)$,
$t'_{k'} \sim \mathcal{U}(l_{k'}, l_{k'} + \xi_3),\ \forall k' \in K \setminus \{1\}$,
$t_k = t_1 + \sum_{k'=2}^{k} t'_{k'},\ \forall k \in K \setminus \{1\}$,
$d_j \sim \mathcal{U}(\xi_4, t_q + l_q - \xi_5),\ \forall j \in J$,
$\xi_s \geq 0,\ \forall s \in \{1,\ldots,5\}$.

samples for parameters p, u_j, t_k, and d_j. The other parameters like n and q are constants and they define the structure of instances during the experiments. It is straightforward that certain numerical features make the problem relatively hard or easy. We generated our test instances according to the probabilistic regime in Table 2 such that $n \leq \left\lfloor \sum_{k\in K} l_k/p \right\rfloor + q$ and $\sum_{k\in K} v_k \geq \sum_{j\in J} u_j$.

Parameter set ξ_s serves to control the computational difficulty of the problem. Empirically, we can deduce that any increase in ξ_1, ξ_2, and ξ_3 and decrease in ξ_4 and ξ_5 tend to alleviate the computational difficulty. It follows that any rise in ξ_1 increases the probability of obtaining homogeneous $u_j,\ j \in J$ and multiple alternatives for a specific resource allocation. Parameter ξ_2 controls the tightness of the difference between the total available resources and the total required resources.

According to Proposition 4.1, if $\xi_3 \geq p$, then the instance is conveniently solved to optimality since finding the optimal solution for the original problem is equivalent to finding the optimal solution for each decomposed sub-problem in terms of the supply periods. The ACFS certifies the globally optimal solution if Proposition 4.1 holds. Recall that the performance of assignment problem P(2) is based on the position of the due dates. Hence, any growth in ξ_4 or ξ_5 reduces the numerical variation in the due dates and thus increases the difficulty of the instance.

We report the overall scalability tests' results in Tables 3 and 4. For each trial, e.g., $n = 10$ and $\sum_{k\in K} l_k = 40$, we generated 100 samples and retrieved the mean and maximum running time and the mean and maximum gap from the best incumbent retrieved by the optimization engine during solving P(1). Table 3 indicates that the ACFS produces the optimal schedule, i.e., zero optimality gap. If the solver or the algorithm does not terminate within 3600 seconds, we call that specific trial an "unsolved problem." For example, the last trial in Table 3 indicates that no problem was solved by P(1) within our maximum time limit; however, the retrieved incumbents were equal to the optimal objective values. It follows that the optimization engine was unable to provide the optimality certificate.

The computational results for hard instances are reported in Table 4. Two main reasons account for the gaps from best incumbent during running the ACFS. The algorithm may confuse to assign jobs to the right supply period based on $u_j,\ j \in J$ due to the small gaps between the end of the current supply period and the start of the next supply period, i.e., $\xi_3 < p$. The computational tests indicate that the ACFS provides tight lower bounds in this case.

In the next performance analysis test, we fix $p = 3$ and design 10 experiments with a moderate setting ($50 \leq \xi_1 \leq 70$, $10 \leq \xi_2 \leq 20$, $\xi_3 \geq p$, and $0 \leq \xi_4, \xi_5 \leq 9$) for $n = 10, \ldots, 40$. We generate the samples based on their interactions via a D-optimal design mechanism. For each experiment, according to Table 2, 20 samples are generated

Table 3. Scalability test analysis and running times for easy and moderate instances ($50 \leq \xi_1 \leq 70$, $10 \leq \xi_2 \leq 20$, $\xi_3 \geq p$, and $\xi_4 = \xi_5 = 0$)

n	$\sum_{k\in K} l_k$	problems solved.	mean time (sec).		max time (sec).		mean gap (%).		max gap (%).	
		P(1)	P(1)	ACFS	P(1)	ACFS	P(1)	ACFS	P(1)	ACFS
10	40	100	0.0331	0.0138	0.3290	0.0390	0	0	0	0
15	50	100	0.7811	0.0195	44.63	0.0431	0	0	0	0
20	60	100	1.03	0.0215	38.84	0.0529	0	0	0	0
25	100	100	5.77	0.223	41.09	0.0478	0	0	0	0
30	120	100	9.27	0.0286	44.86	0.0490	0	0	0	0
40	200	100	138	0.0921	1007	0.5710	0	0	0	0
50	240	65	671	1.28	3600+	2.56	0	0	0	0
60	300	0	3600+	1.51	3600+	2.77	0	0	0	0

Table 4. Scalability test analysis and running times for hard instances ($10 \leq \xi_1 \leq 30$, $5 \leq \xi_2 \leq 10$, $\xi_3 < p$, $\xi_4 = (t_q + l_q)/4$, and $\xi_5 = 3(t_q + l_q)/4$)

n	$\sum_{k\in K} l_k$	problems solved.	mean time (sec).		max time (sec).		mean gap (%).		max gap (%).	
		P(1)	P(1)	ACFS	P(1)	ACFS	P(1)	ACFS	P(1)	ACFS
10	40	100	0.4794	0.0103	3.71	0.0339	0	0.06	0	0.11
15	50	100	3.76	0.0267	67.42	0.0315	0	0.17	0	0.28
20	60	100	4.94	0.0272	109.16	0.0606	0	1.45	0	1.71
25	100	88	14.88	0.0833	678.85	1.76	6.15	1.89	6.15	4.78
30	120	57	29.33	0.1574	456.18	1.43	3.05	3.40	7.09	8.36
40	200	32	138	0.2628	3600+	2.57	6.99	6.03	11.35	10.59
50	240	3	3600+	2.63	3600+	2.83	8.34	7.37	14.72	11.03
60	300	0	3600+	2.97	3600+	3.39	14.61	12.17	15.42	17.67

and Mean Running Times (MNRT), Maximum Running Times (MXRT), Mean number of Tardy Jobs (MNTJ), and Maximum number of Tardy Jobs (MXTJ) are reported in Tables 5 through 8. Note that the number of tardy jobs is the indicator function of the total tardiness objective and computes the number of late jobs. The running time is bound to 3600 seconds.

Table 5. Performance analysis based on interactive design of experiments on ξ_s with moderate setting ($n = 10$)

ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	P(1)				ACFS			
					MNRT	MXRT	MNTJ	MXTJ	MNRT	MXRT	MNTJ	MXTJ
52	12	3	0	9	0.09	1.30	1.1	6	0.0056	0.008	1.1	6
67	19	7	2	5	0.10	4.25	1.5	5	0.0058	0.01	1.5	5
64	12	4	3	4	0.10	1.62	1	3	0.0035	0.005	1	3
54	15	4	3	3	0.09	1.43	1	2	0.0045	0.008	1	2
69	12	5	4	0	0.29	3.56	1	3	0.0051	0.0101	1	3
51	11	9	0	4	0.09	1.24	0.8	3	0.0037	0.005	0.8	3
52	10	4	4	1	0.13	3.02	1.2	5	0.0035	0.0101	1.2	5
51	15	7	2	2	0.09	2.59	1.6	6	0.0033	0.0082	1.6	6
62	19	3	3	8	0.09	1.17	1.3	4	0.004	0.005	1.3	4
54	13	6	0	7	0.09	1.50	2	7	0.0038	0.0102	2	7

According to Tables 5 through 7, both P(1) and the ACFS produce the same MNTJ and MXTJ. MXRT in Table 7 for at least one sample of any experiment is more than 3600 seconds; however in spite of having the same MNTJ and MXTJ, the optimization engine is unable to provide the optimality certificate for the instances of P(1) and it cannot close the gap. On the other hand, Table 8 indicates that the ACFS outperforms P(1) in both the running times and the number of tardy jobs.

In the last experiment, we fix $p = 3$ and choose a slightly difficult setting ($20 \leq \xi_1 \leq 50$, $10 \leq \xi_2 \leq 20$, $\xi_3 < p$, and $\xi_4 = \xi_5 = 0$). We test the performance of the ACFS based on the gap from the best incumbent for different pairs of the number of jobs and the number of supply periods, (n, q). For each pair, we generate 100 samples and the results are reported as the gap percentage for different instances in Figure 3.

Table 6. Performance analysis based on interactive design of experiments on ξ_s with moderate setting ($n = 20$)

ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	P(1)				ACFS			
					MNRT	MXRT	MNTJ	MXTJ	MNRT	MXRT	MNTJ	MXTJ
52	12	3	0	9	91.17	289.29	2.9	7	0.170	0.479	2.9	7
67	19	7	2	5	65.03	286.27	1.6	6	0.110	0.450	1.6	6
64	12	4	3	4	11.78	113.26	1.2	3	0.115	0.448	1.2	3
54	15	4	3	3	35.92	147.21	0.5	1	0.245	0.489	0.5	1
69	12	5	4	0	82.64	743.10	2	4	0.112	0.468	2	4
51	11	9	0	4	69.71	3242.59	0.3	1	0.291	0.460	0.3	1
52	10	4	4	1	48.32	115.71	1	3	0.127	0.450	1	3
51	15	7	2	2	43.25	225.55	1.6	6	0.126	0.420	1.6	6
62	19	3	3	8	125.70	284.02	2	7	0.112	0.420	2	7
54	13	6	0	7	56.34	122.32	1.9	7	0.222	0.489	1.9	7

Table 7. Performance analysis based on interactive design of experiments on ξ_s with moderate setting ($n = 30$)

ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	P(1)				ACFS			
					MNRT	MXRT	MNTJ	MXTJ	MNRT	MXRT	MNTJ	MXTJ
52	12	3	0	9	1324	3600+	5.5	7	0.619	1.804	5.5	7
67	19	7	2	5	712	3600+	8.2	11	0.608	1.638	8.2	11
64	12	4	3	4	1430	3600+	6.8	11	0.538	1.647	6.8	11
54	15	4	3	3	1844	3600+	7.6	10	0.520	1.567	7.6	10
69	12	5	4	0	8382	3600+	8.1	9	0.556	1.688	8.1	9
51	11	9	0	4	1771	3600+	7.2	12	0.553	1.618	7.2	12
52	10	4	4	1	1282	3600+	7.6	13	0.535	1.613	7.6	13
51	15	7	2	2	1830	3600+	7.3	11	0.535	1.578	7.3	11
62	19	3	3	8	836	3600+	6.7	11	0.557	1.618	6.7	11
54	13	6	0	7	1481	3600+	6.8	13	0.589	1.641	6.8	13

Table 8. Performance analysis based on interactive design of experiments on ξ_s with moderate setting ($n = 40$)

ξ_1	ξ_2	ξ_3	ξ_4	ξ_5	P(1)				ACFS			
					MNRT	MXRT	MNTJ	MXTJ	MNRT	MXRT	MNTJ	MXTJ
52	12	3	0	9	2711	3600+	9.8	14	1.798	3.930	7.3	12
67	19	7	2	5	2754	3600+	10.2	16	1.750	2.808	6.8	8
64	12	4	3	4	2187	3600+	11.4	15	1.732	2.947	8.1	12
54	15	4	3	3	2062	3600+	11.4	18	2.693	3.768	7.5	9
69	12	5	4	0	2593	3600+	12.5	18	1.691	2.768	7.3	11
51	11	9	0	4	2375	3600+	10.8	15	1.696	2.808	7.3	12
52	10	4	4	1	2445	3600+	10.1	13	1.682	2.748	7.5	11
51	15	7	2	2	2275	3600+	11.6	19	1.692	2.761	7.8	13
62	19	3	3	8	2635	3600+	11.1	16	2.667	3.733	8.6	13
54	13	6	0	7	2390	3600+	9.7	13	1.656	2.699	7.1	11

As illustrated in Figure 3, the overall gap increases by adding up to n and drop in q. There is at least one sample in instances $(20, 4)$, $(20, 6)$, and $(30, 8)$ for which the ACFS obtains the optimal solution (zero gap). It is natural to observe that in some instances, the lower bound equals the optimal solution. The gap variations for instances $(40, 8)$, $(40, 10)$, and $(40, 12)$ are more significant than for the other instances. A large number of jobs and the variations in the distances of the due dates from the supply starts and ends partially account for the heterogeneous gaps.

6. Conclusion

We provided several optimality results for the problem of minimizing the sum of earliness and tardiness on a single machine with a periodically supplied non-renewable

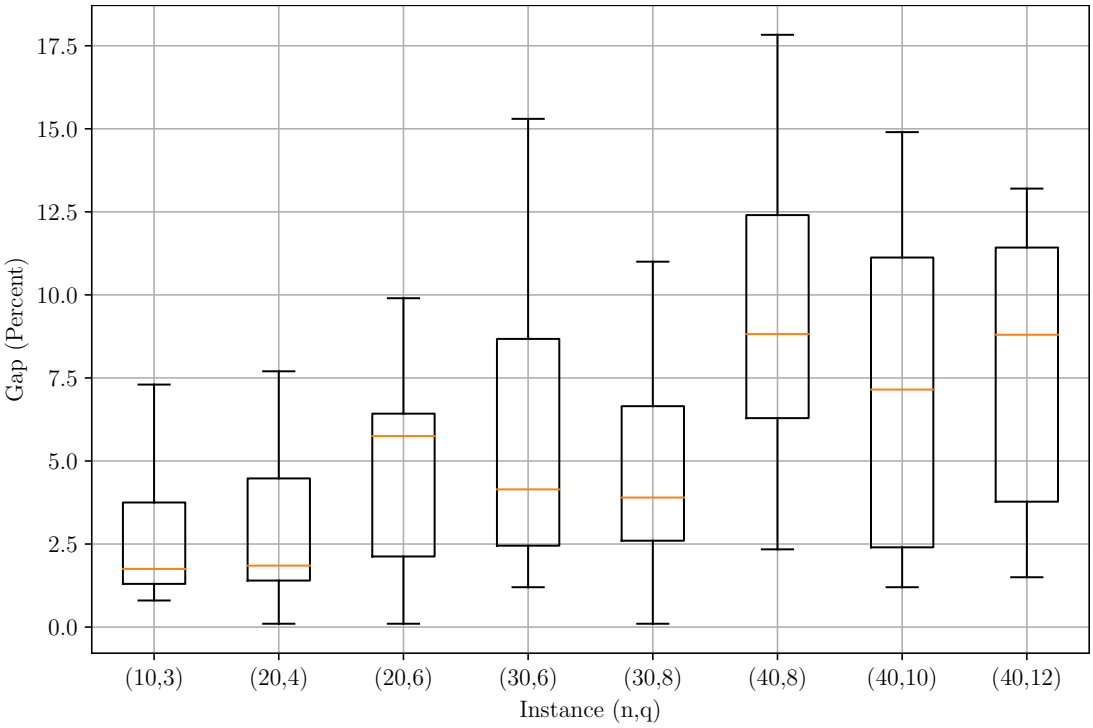

Figure 3. Optimality gap analyses (difficult instances)

resource. We showed that there exists a tractable scheme for this problem. We first formulated a decomposition procedure to assign jobs to appropriate supply periods and then jobs are grouped into chains in each supply period. Next, their respective start times are determined using a polynomial-time optimal algorithm. The scalability tests analyses over easy, moderate, and hard instances indicated that the running time grows sub-linearly by the increase in the number of jobs and the number of supply periods. We also discuss how running times of our procedure could justify the gaps from best incumbents retrieved by the optimization engine during solving P(1). For future research, we propose to investigate the structural and functional properties of the original problem when the supply periods are not given as inputs.

References

M. Andersson, S. Mattsson, Dose management in conventional nuclear medicine imaging and pet, Clinical and Translational Imaging 4 (2016) 21–30.

E. R. Gafarov, A. A. Lazarev, F. Werner, Single machine scheduling problems with financial resource constraints: Some complexity results and properties, Mathematical Social Sciences 62 (2011) 7–13.

M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2016.

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

P. Györgyi, T. Kis, Approximation schemes for single machine scheduling with non-renewable resource constraints, Journal of Scheduling 17 (2014) 135–144.

J. Vallikavungal Devassia, M. A. Salazar-Aguilar, V. Boyer, Flexible job-shop scheduling problem with resource recovery constraints, International Journal of Production Research 56 (2018) 3326–3343.

J. Böttcher, A. Drexl, R. Kolisch, F. Salewski, Project scheduling under partially renewable resource constraints, Management Science 45 (1999) 543–559.
A. Naber, R. Kolisch, MIP models for resource-constrained project scheduling with flexible resource profiles, European Journal of Operational Research 239 (2014) 335–348.
L. Wan, J. Yuan, Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard, Operations Research Letters 41 (2013) 363–365.
J. S. Davis, J. J. Kanet, Single-machine scheduling with early and tardy completion costs, Naval Research Logistics (NRL) 40 (1993) 85–101.
W. Szwarc, S. K. Mukhopadhyay, Optimal timing schedules in earliness-tardiness single machine sequencing, Naval Research Logistics (NRL) 42 (1995) 1109–1114.
Y.-D. Kim, C. A. Yano, Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates, Naval Research Logistics (NRL) 41 (1994) 913–933.
S. Verma, M. Dessouky, Single-machine scheduling of unit-time jobs with earliness and tardiness penalties, Mathematics of Operations Research 23 (1998) 930–943.
F. Sourd, S. Kedad-Sidhoum, The one-machine problem with earliness and tardiness penalties, Journal of scheduling 6 (2003) 533–549.
F. Sourd, Punctuality and idleness in just-in-time scheduling, European Journal of Operational Research 167 (2005) 739–751.
Y. Hendel, F. Sourd, An improved earliness–tardiness timing algorithm, Computers & operations research 34 (2007) 2931–2938.
M. Wodecki, A block approach to earliness-tardiness scheduling problems, The International Journal of Advanced Manufacturing Technology 40 (2009) 797–807.
K. R. Baker, B. Keller, Solving the single-machine sequencing problem using integer programming, Computers & Industrial Engineering 59 (2010) 730–735.
R. Cohen, L. Katzir, D. Raz, An efficient approximation for the generalized assignment problem, Information Processing Letters 100 (2006) 162–166.
N. C. Schwertman, A. Gilks, J. Cameron, A simple noncalculus proof that the median minimizes the sum of the absolute deviations, The American Statistician 44 (1990) 38–39.

Appendix A.

Algorithm 2, which is a detailed version of Algorithm 1, also consists of two primary routines, given the number of jobs dedicated to supply period k, i.e., $\sum_j x^*_{jk}$. The first routine, which starts from line 5 to 20, groups the jobs into chains and computes the start times of the first and the last jobs in each chain. The second routine, from line 21 onward, checks whether the current chain is conflicting with the previous or the next chain. If such a case happens, it will join two conflicting chains and compute the start time of the first job in the emerging chain. The routine continues until there are no conflicting chains in the entire schedule. According to Proposition 3.5, the final schedule is semi-active. Function `update_list`(π, b) refers to performing push operations on the items of a list in $O(1)$. If π_{c+1} is redundant, function `update_list`(π, b) forces $\pi_c \leftarrow \langle \pi_c, \pi_{c+1} \rangle$ and $C \leftarrow \{1, \ldots, b-1\}$ operations and re-indexes the chain list.

Algorithm 2: Recursive scheme to produce optimal schedule for $1 \mid p_j = p, s_j \in [t_k, t_k + l_k] \mid \texttt{ET}$ (detailed version)

Input: Sorted list of jobs based on `EDD`

$J_k \leftarrow \left\{1, \ldots, \sum_j x^*_{jk}\right\}$
if $l_k = \left(|J_k| - 1\right) p$ **then**
  return $\left\{s_j \leftarrow t_k + (j-1)p,\ \forall j \in J_k\right\}$
else
  for $c \in J_k$ **do**
    $\pi_c \leftarrow \langle \emptyset \rangle$
  $c \leftarrow 1$
  for $j \in J_k$ **do**
    if $\pi_c = \langle \emptyset \rangle$ ***and*** $d_{j+1} - d_j \leq p$ **then**
      $\pi_c \leftarrow \langle \pi_c, j, j+1 \rangle$
    else if $d_{j+1} - d_j \leq p$ **then**
      $\pi_c \leftarrow \langle \pi_c, j+1 \rangle$
    else
      $n_c \leftarrow |\pi_c|$
      $s_{1 \in \pi_c} \leftarrow \max\left\{t_k, (\gamma_1 + \gamma_2)/2\right\}$
      $s_{n_c \in \pi_c} \leftarrow \min\left\{t_k + l_k, s_{1 \in \pi_c} + (n_c - 1)p\right\}$
      if $s_{n_c \in \pi_c} := t_k + l_k$ **then**
        $s_{1 \in \pi_c} \leftarrow t_k + l_k - (n_c - 1)p$
      $c \leftarrow c + 1$
  $b \leftarrow c$; $c \leftarrow 1$
  while $c \leq b$ **do**
    if $s_{n_c \in \pi_c} - s_{1 \in \pi_c} < (n_c - 1)p$ **then**
      $s_{1 \in \pi_c} \leftarrow \max\left\{t_k, (\gamma_1 + \gamma_2)/2\right\}$
      $s_{n_c \in \pi_c} \leftarrow \min\left\{t_k + l_k, s_{1 \in \pi_c} + (n_c - 1)p\right\}$
      if $s_{n_c \in \pi_c} := t_k + l_k$ **then**
        $s_{1 \in \pi_c} \leftarrow t_k + l_k - (n_c - 1)p$
      if $c < b$ ***and*** $s_{1 \in \pi_{c+1}} - s_{n_c \in \pi_c} \leq p$ **then**
        $\pi_c \leftarrow \langle \pi_c, \pi_{c+1} \rangle$
        $b \leftarrow b - 1$
        `update_list`$\left(\langle \pi_1, \ldots, \pi_b \rangle, b\right)$
      else if $c > 1$ ***and*** $s_{1 \in \pi_c} - s_{n_{c-1} \in \pi_c} \leq p$ **then**
        $\pi_{c-1} \leftarrow \langle \pi_{c-1}, \pi_c \rangle$ $c \leftarrow c - 1$; $b \leftarrow b - 1$
        `update_list`$\left(\langle \pi_1, \ldots, \pi_b \rangle, b\right)$
      else
        $c \leftarrow c + 1$
  $C \leftarrow \{1, \ldots, b\}$
  return $\{s_{1 \in \pi_c},\ \forall c \in C\}$