https://doi.org/10.1016/j.ejor.2020.12.006

Single-machine hierarchical scheduling with release dates and preemption to minimize the total completion time and a regular criterion

Rubing Chen[1,2], Jinjiang Yuan[1*], C.T. Ng[3], T.C.E. Cheng[3]

[1]School of Mathematics and Statistics, Zhengzhou University,
Zhengzhou, Henan 450001, People's Republic of China
[2]School of Information Engineering, Zhengzhou University,
Zhengzhou, Henan 450001, People's Republic of China
[3]Logistics Research Centre, Department of Logistics and Maritime Studies,
The Hong Kong Polytechnic University, Hong Kong SAR, People's Republic of China

Abstract: In this paper we consider the single-machine hierarchical scheduling problems with release dates and preemption, where the primary criterion is the total completion time and the secondary criterion is an arbitrarily regular scheduling criterion, which is of either the sum-form or the max-form. We aim to find a feasible preemptive schedule that minimizes the secondary criterion, subject to the condition that the primary criterion is minimized. We show that the variants of the problems under study are polynomially solvable. To address these problems, we develop new solution techniques that establish some hereditary properties for the feasible schedules and instances, and present a complete description of the feasible schedules through some elaborately constructed job-permutations.

Key words: scheduling; hierarchical criteria; release date; preemption; total completion time.

Email address:
zzucrb2009@163.com (Rubing Chen),
yuanjj@zzu.edu.cn (Jinjiang Yuan),
daniel.ng@polyu.edu.hk (C.T. Ng),
edwin.cheng@polyu.edu.hk (T.C.E. Cheng).

*Corresponding author. Email address: yuanjj@zzu.edu.cn (J.J. Yuan)

1 Introduction

Research Motivation: In real-life scheduling, it often occurs that two scheduling criteria are to be optimized. However, in most cases, an optimal schedule for one criterion is not optimal for the other criterion. This has aroused people's interest in bicriterion scheduling. For detailed discussions of research on bicriterion scheduling, the reader may refer to Yen and Wan (2003), Hoogeveen (2005), T'kindt and Billaut (2006), and Agnetis et al. (2014), among others.

Scheduling problems with release dates and preemption have many applications in the real world. For example, printing enterprises are actively engaged in the printing of publicity materials for epidemic prevention and control, contributing to the fight against an epidemic. Suppose that there are n customers placing orders (jobs) to a printer with different release times (release dates in scheduling terms) and expected finish times (due dates). The printing jobs are preemptive, meaning that the printer may print a job up to a certain page, shift to printing some other jobs, and return to printing the rest of the interrupted job later. Suppose that the primary objective of the printer is to find a schedule to complete all the jobs as early as possible, i.e., minimizing the average completion time or, equivalently, minimizing the total completion time of the jobs. On the other hand, tardiness is an important performance indicator of the printer's reputation. Therefore, the secondary objective of the printer is to minimize the number of tardy jobs or total weighted tardiness of the jobs (for jobs with varying importance in tardiness) among all the optimal solutions for the primary objective.

For another example, a computer repair shop requests customers to book for the repair services at least one day beforehand. Through telephone booking, each customer describes the symptoms of the computer and the shop will provide an estimate of the repair duration (processing time). The tasks of repairing computers are naturally preemptive. Suppose that a technician is assigned n computers to repair with different release times (release dates) in a day. The primary objective of the technician is to find a schedule to complete all the jobs as early as possible, i.e., minimizing the total completion time of the jobs. On the other hand, to give good impression to the customers, the technician would also like to make the maximum lateness of the jobs as small as possible, which implies that all the lateness of the jobs is within this limit. Therefore, the secondary objective is to minimize the maximum lateness among all the optimal solutions for the primary objective.

In general, we consider all kinds of sum-forms (e.g., total tardiness and total weighted completion time) and max-forms (e.g., makespan and maximum lateness) for the secondary objective. Therefore, the application range of our problem is very wide. Similar applications can also be found in centralized wireless data networks, bandwidth-sharing networks, traffic systems, and so on (Prakash and Veeravalli, 2007; Aalto and Ayesta,

2009; Ahmad et al., 2013). These practical applications motivate us to study the hierarchical scheduling problem with release dates and preemption.

Problem Formulation: Suppose that $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is a set of n jobs to be preemptively processed on a single machine. Each job J_j has a release date $r_j \geq 0$, a processing time $p_j \geq 0$, a due date $d_j \geq 0$, and a weight $w_j \geq 0$. In a feasible schedule, a job J_j may be decomposed into several parts that have a total processing time p_j. For convenience, we use $\mathcal{J}$ to represent both the job set and the job instance.

Following Pinedo (2002), Leung (2004) and Brucker (2006), we will use the following notation for a feasible schedule π and a job $J_j \in \mathcal{J}$: $C_j(\pi)$ (the completion time of job J_j), $f_j(\pi)$ (the scheduling cost of job J_j), $L_j(\pi)$ (the lateness of job J_j), $T_j(\pi)$ (the tardiness of job J_j), $w_jC_j(\pi)$ (the weighted completion time of job J_j), and $U_j(\pi)$ (the tardy indicator of job J_j).

In this paper, we deal with the following two types of objective functions: $\sum f_j = \sum_{j=1}^{n} f_j(\pi)$ (the total cost) and $f_{\max} = \max\{f_j(\pi) : j = 1, 2, \ldots, n\}$ (the maximum cost).

Remark: We assume that all the cost functions $f_j(\cdot)$ considered in this paper are *regular*, i.e., $f_j(t)$ is a nondecreasing function in $t \in [0, +\infty)$. Moreover, we take the convention in this paper that, for each index $j \in \{1, 2, \ldots, n\}$ and each time $t \in [0, +\infty)$, $f_j(t)$ is a finite number, i.e., $-\infty < f_j(t) < +\infty$.

From Pinedo (2002), Leung (2004) and Brucker (2006), most objective functions in scheduling research are special versions of $\sum f_j$ and $f_{\max}$. For example, $\sum C_j$ (the total completion time), $\sum w_jC_j$ (the total weighted completion time), $\sum T_j$ (the total tardiness), $\sum w_jT_j$ (the total weighted tardiness), $\sum U_j$ (the number of tardy jobs), $\sum w_jU_j$ (the weighted number of tardy jobs), $L_{\max}$ (the maximum lateness), $T_{\max}$ (the maximum tardiness), and $WC_{\max}$ (the maximum weighted completion time), where $WC_{\max}(\pi) = \max\{w_jC_j(\pi) : j = 1, 2, \ldots, n\}$. Following the three-field notation introduced in Graham et al. (1979), we use $1|\beta|f$ to denote a single-machine single-criterion scheduling problem, where β indicates the scheduling requirements and f is the scheduling criterion to be minimized.

According to Yen and Wan (2003), Hoogeveen (2005), T'kindt and Billaut (2006), and Agnetis et al. (2014), the following four types of bicriterion scheduling problems have been widely studied in the literature.

Hierarchical Scheduling Problem: $1|\beta|\text{Lex}(f, g)$. The problem concerns finding a feasible schedule that minimizes the secondary criterion g, subject to the condition that the primary criterion f is minimized.

Constrained Scheduling Problem: $1|\beta|f : g \leq Q$. The problem deals with finding a

feasible schedule that minimizes the criterion f, subject to the condition that $g \leq Q$.

Weighted-Sum Scheduling Problem: $1|\beta|\lambda_1 f + \lambda_2 g$. The problem seeks to find a feasible schedule that minimizes the single criterion $\lambda_1 f + \lambda_2 g$, where λ_1 and λ_2 are two positive constants.

Pareto Scheduling Problem: $1|\beta|^{\#}(f, g)$. The problem is concerned with finding all the Pareto-optimal points and for each Pareto-optimal point, providing the corresponding Pareto-optimal schedule.

For convenience, we use $1|\beta|\{f, g\}$ to denote the family of all the six bicriterion scheduling problems $1|\beta|\text{Lex}(f, g)$, $1|\beta|\text{Lex}(g, f)$, $1|\beta|f : g \leq Q$, $1|\beta|g : f \leq Q$, $1|\beta|\lambda_1 f + \lambda_2 g$, and $1|\beta|^{\#}(f, g)$.

Literature Review and Discussion: Bicriterion scheduling is a classical and popular topic in scheduling research. Representative studies on bicrirerion scheduling can be found in Edmmons (1975), Van Wassenhove and Gelders (1980), Lin (1983), Nelson et al. (1986), Kiran and Unal (1991), Chen and Bulfin (1993), Kyparisis and Douligeris (1993), Hoogeveen and van de Velde (1995), Hoogeveen (1996), Kondakci and Bekiroglu (1997), Sourd (2001), Agnetis et al. (2004), Huo et al. (2007), Huang and Yang (2009), Gao and Yuan (2015), Gao and Yuan (2017), and Akande et al. (2017), among many others.

We now consider the bicriterion scheduling problems in the family $1|r_j, \text{pmtn}|\{\sum C_j, f\}$, where one criterion is the total completion time $\sum C_j$ and the other criterion is an arbitrarily regular objective function f. Labetoulle et al. (1984) showed that problem $1|r_j, \text{pmtn}|\sum w_j C_j$ is unary NP-hard. Du and Leung (1993), and Wan et al. (2015) showed that problem $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$ is binary NP-hard, where $\bar{d}_j$ is the deadline of job J_j. Recently, Chen and Yuan (2018) further showed that problem $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$ is unary NP-hard. Then the unary NP-hardness of problems $1|r_j, \text{pmtn}|\sum w_j C_j$ and $1|r_j, \bar{d}_j, \text{pmtn}|\sum C_j$ leads to the following observation.

Observation 1.1. *For any criterion $f \in \{\sum w_j C_j, \sum U_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$, the hierarchical scheduling problem $1|r_j, pmtn|Lex(f, \sum C_j)$ is unary NP-hard.*

From Observation 1.1, the following observation can be verified directly.

Observation 1.2. *For any criterion $f \in \{\sum w_j C_j, \sum U_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$, apart from the hierarchical scheduling problem $1|r_j, pmtn|Lex(\sum C_j, f)$, all the other problems in the family $1|r_j, pmtn|\{\sum C_j, f\}$ are unary NP-hard.*

Shedding an optimistic light, Schrage (1968) and Baker (1974) showed that the shortest remaining processing time (SRPT) rule solves problem $1|r_j, \text{pmtn}|\sum C_j$ in $O(n \log n)$ time. This stimulates us to study the following two general problems:

$$1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j) \tag{1}$$

and

$$1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max}), \tag{2}$$

where each $f_j(\cdot)$, $j = 1, 2, \ldots, n$, is a regular function. Evidently, a feasible schedule for problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$ must be an optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$. If a job has processing time 0, then it can be completed at its release date without affecting the other jobs, so we assume that $p_j > 0$ for each job $J_j \in \mathcal{J}$.

Note that in the implementation of SRPT rule, there may exist ties for scheduling jobs and resolving the ties leads to different optimal schedules for the $\sum C_j$ criterion. This defines the set of feasible solutions for the problems in (1) and (2).

Contributions: We show in this paper that the problem in (1) is solvable in $O(n^3)$ time and the problem in (2) is solvable in $O(n^2)$ time. We also point out that, for any criterion $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$, problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$ is solvable in $O(n \log n)$ time.

Methodology: To address the problems in (1) and (2), we develop new techniques as follows:

– We show that each feasible schedule can be regarded as a job-permutation that sequences the jobs in an increasing order of their completion times so that the technique "preemptive-list schedules" can be used in our analysis.

– We reduce the general instances to standard instances (defined in Section 2) so that the analysis becomes easier to some extent.

– We establish some hereditary properties for the feasible schedules and for instances, which help reveal the intrinsic nature of the problems.

– We present a complete description of the feasible schedules through some elaborately constructed job-permutations.

– The above techniques enable us to reduce the problem in (1) to an $n \times n$ linear assignment problem, which is solvable in $O(n^3)$ time, and to solve the problem in (2) in $O(n^2)$ time by an adaptation of Lawler's rule, established in Lawler (1973), for solving problem $1||f_{\max}$.

Organization: We organize the rest of the paper as follows: In Section 2, we re-visit the classical problem $1|r_j, \text{pmtn}|\sum C_j$ and present an in-depth study of the structure of its optimal schedules. Specifically, we introduce the concepts "preemptive-list schedules, standard instances, legal sets, and legal permutations", and study their relations. Then we present a complete description of the optimal schedules for problem $1|r_j, \text{pmtn}|\sum C_j$ by using the legal permutations. In Section 3, we show that the legal sets of a standard instance can be generated in $O(n^2)$ time and the last legal set of a standard instance can be generated in $O(n)$ time. In Section 4, we provide polynomial-time algorithms for the

two problems in (1) and (2). In Section 5, we present a numerical example to explain some terms and notation in our research and provide computational experiments for checking the efficiency of our algorithms established in this paper.

2 Re-visit problem $1|r_j, \text{pmtn}|\sum C_j$

We now re-visit the classical scheduling problem $1|r_j, \text{pmtn}|\sum C_j$. Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be the job instance to be scheduled. When there is no risk of confusion, we write schedules for problem $1|r_j, \text{pmtn}|\sum C_j$ on instance $\mathcal{J}$ as "schedules for problem $1|r_j, \text{pmtn}|\sum C_j$" or "schedules for instance $\mathcal{J}$".

2.1 The SRPT rule

In the preprocessing procedure for solving problem $1|r_j, \text{pmtn}|\sum C_j$, we re-number the n jobs of $\mathcal{J}$ by the earliest release date (ERD) rule such that $r_1 \leq r_2 \leq \cdots \leq r_n$. Then we schedule the n jobs of $\mathcal{J}$ preemptively and forwards beginning with time r_1 by the following SRPT rule proposed in Schrage (1968).

The SRPT Rule: *At each decision time τ (when some jobs are released or completed), we schedule an available job (if any) with the smallest remaining processing time. This procedure is repeated until all the jobs are scheduled.* (Here, by "an available job at time τ", we mean that the job is released by time τ and has not been completed.)

Schrage (1968) and Smith (1978) showed that the SRPT rule is the unique strategy for solving problem $1|r_j, \text{pmtn}|\sum C_j$. Their findings can be formally described in the following lemma.

Lemma 2.1. *Every optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$ is generated by the SRPT rule. Moreover, when the instance $\mathcal{J}$ is given, all the optimal schedules have the same sequence of job completion times.*

From Lemma 2.1, the n completion times of the n jobs of $\mathcal{J}$ in all the optimal schedules are uniquely determined by an implementation of the SRPT rule. Denote the n completion times by $C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$ such that $C^{(1)}(\mathcal{J}) < C^{(2)}(\mathcal{J}) < \cdots < C^{(n)}(\mathcal{J})$. Then the optimal objective value of problem $1|r_j, \text{pmtn}|\sum C_j$ is $C^{(1)}(\mathcal{J}) + C^{(2)}(\mathcal{J}) + \cdots + C^{(n)}(\mathcal{J})$. Set $\mathcal{C}(\mathcal{J}) = \{C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})\}$.

Let $\Pi^*(\mathcal{J})$ be the set of schedules generated by the SRPT rule on instance $\mathcal{J}$. From Lemma 2.1, $\Pi^*(\mathcal{J})$ consists of all the optimal schedules for problem $1|r_j, \text{pmtn}|\sum C_j$. Lemma 2.1 also indicates that there is a bijection between all the schedules in $\Pi^*(\mathcal{J})$ and

all the implementations of the SRPT rule on instance $\mathcal{J}$. Thus, in most cases, we discuss the schedules in $\Pi^*(\mathcal{J})$, which is more convenient than discussion of the SRPT rule.

Let σ be a schedule for $\mathcal{J}$. Let $J_j \in \mathcal{J}$ and $\tau \in [r_j, C_j(\sigma))$. It is clear that job J_j is available at time τ in σ. We use $p_j^{(\sigma)}(\tau)$ to denote the remaining processing time of job J_j at time τ in σ and use $J_j^{(\sigma)}(\tau)$ to denote the remaining part of job J_j at time τ in σ. Thus, in schedule σ, $J_j^{(\sigma)}(\tau)$ can be regarded as a job with processing time $p_j^{(\sigma)}(\tau)$.

Let $\mathcal{T}(\mathcal{J})$ be the set of decision times in the implementation of the SRPT rule. Then we have $\mathcal{T}(\mathcal{J}) = \{r_j : 1 \leq j \leq n\} \cup \{C^{(i)}(\mathcal{J}) : 1 \leq i \leq n\}$. For convenience, we write $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_{n'}\}$ such that $\tau_1 < \tau_2 < \cdots < \tau_{n'}$. Then we have $n+1 \leq n' \leq 2n$, $\tau_1 = r_1$, and $\tau_{n'} = C^{(n)}(\mathcal{J})$.

Since ties may arise in the running process of the SRPT rule, different implementations of the SRPT rule generate different optimal schedules. But the following lemma establishes an invariant property of the remaining processing times at every decision time.

Lemma 2.2. *For each schedule $\pi \in \Pi^*(\mathcal{J})$ and each decision time $\tau_i \in \mathcal{T}(\mathcal{J})$, the remaining processing times (with repetitions being counted) of the available jobs at time τ_i are independent of the choice of π.*

Proof. Consider an implementation of the SRPT rule for generating the optimal schedule π. Since $\tau_1 = r_1$, the remaining processing times of the available jobs at time τ_1 are clearly independent of the choice of π. Assume that the result is true for a decision time τ_{i-1} with $2 \leq i \leq n'$. If no jobs are available at time τ_{i-1}, the result is clearly true for the decision time τ_i. If at least one job is available at time τ_{i-1}, by the SRPT rule, each of the available jobs has a processing time of at least $\tau_i - \tau_{i-1}$ and one of such jobs, say, J_j, with the smallest remaining processing time is scheduled in the interval $[\tau_{i-1}, \tau_i]$ in π. It is routine to see that the remaining processing times of the available jobs at time τ_i are independent of the choice of J_j, so independent of π. The lemma follows. □

From Lemma 2.2, for each $\tau_i \in \mathcal{T}(\mathcal{J})$, the remaining processing times (of the available jobs) at time τ_i in every schedule in $\Pi^*(\mathcal{J})$ are invariant. Then the term "the remaining processing times at time τ_i" has its meaning.

2.2 Preemptive-list schedules

The "preemptive-list schedule" is a useful tool for studying scheduling problems with release dates and preemption. Let $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ be a permutation (of the n jobs) of $\mathcal{J}$. As given in Yuan et al. (2015), a feasible schedule related to the permutation $\mathcal{O}$ can be obtained by the following procedure:

Procedure Pmtn-LS($\mathcal{O}$): *We first schedule $J_{o(1)}$ as early as possible, which means that*

$J_{o(1)}$ is scheduled in the interval $[r_{o(1)}, r_{o(1)}+p_{o(1)}]$. When the first j jobs $J_{o(1)}, J_{o(2)}, \ldots, J_{o(j)}$ have been scheduled and $j < n$, we schedule the $(j+1)$-th job $J_{o(j+1)}$ preemptively in the remaining idle time space as early as possible. This procedure is repeated until all the jobs are scheduled.

Procedure Pmtn-LS plays an important role in our research. To have a better understanding for this procedure, we present the implementation details of Procedure Pmtn-LS as follows.

Implementation Details of Procedure Pmtn-LS(O):
Input: A permutation $\mathcal{O} = \{J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)}\}$.
Step 1: Set $i := 1$, $r_{\max} := \max\{r_j : 1 \le j \le n\}$, $P := \sum_{j=1}^{n} p_j$, and $\mathcal{I} := [0, r_{\max} + P]$. (At the end of each iteration i, we obtain a schedule of $\{J_{o(1)}, J_{o(2)}, \ldots, J_{o(i)}\}$, together with the idle-time space, denoted by $\mathcal{I}$, in the interval $[0, r_{\max} + P]$ not occupied by the schedule.)
Step 2: If $i = n + 1$, then all the jobs have been scheduled and stop the algorithm. Otherwise, go to Step 3.
Step 3: Schedule job $J_{o(i)}$ preemptively and fully in the the idle-time space $[r_{o(i)}, r_{\max} + P] \cap \mathcal{I}$ as early as possible. Denote the time space of $[r_{o(i)}, r_{\max} + P] \cap \mathcal{I}$ that is occupied by $J_{o(i)}$ as $\mathcal{I}_i$. Set $\mathcal{I} := \mathcal{I} \setminus \mathcal{I}_i$ and $i := i + 1$. Return to Step 2.

From Yuan et al. (2015), the running time of Procedure Pmtn-LS($\mathcal{O}$) is $O(n \log n)$. We use Pmtn-LS($\mathcal{O}$) to denote the (preemptive) schedule obtained by the above procedure on the permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ and call Pmtn-LS($\mathcal{O}$) the *Pmtn-LS schedule* (preemptive-list schedule) determined by $\mathcal{O}$. Note that Pmtn-LS($\mathcal{O}$) is uniquely determined by $\mathcal{O}$. Thus, when there is no risk of confusion, we use $\mathcal{O}$ to denote the schedule Pmtn-LS($\mathcal{O}$). Given the interfering of the release dates, different permutations may determine a common Pmtn-LS schedule. According to Yuan et al. (2020), a permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ is called *completion-coinciding* if $C_{o(1)}(\mathcal{O}) < C_{o(2)}(\mathcal{O}) < \cdots < C_{o(n)}(\mathcal{O})$. From Yuan et al. (2020), for every permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$, there must be a completion-coinciding permutation $\mathcal{O}' = (J_{o'(1)}, J_{o'(2)}, \ldots, J_{o'(n)})$ such that Pmtn-LS($\mathcal{O}'$) = Pmtn-LS($\mathcal{O}$).

Let σ be a feasible schedule for problem $1|r_j, \text{pmtn}|\sum C_j$. We use $J_{\sigma(i)}$ to denote the i-th completed job in σ, $i = 1, 2, \ldots, n$. Then $(\sigma(1), \sigma(2), \ldots, \sigma(n))$ is a permutation of the job index set $\{1, 2, \ldots, n\}$ such that $C_{\sigma(1)}(\sigma) < C_{\sigma(2)}(\sigma) < \cdots < C_{\sigma(n)}(\sigma)$. Let $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$, and set $\mathcal{O}_i^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(i)})$ and $\mathcal{J}_i^{(\sigma)} = \{J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(i)}\}$ for $i = 1, 2, \ldots, n$. In this paper we define a *time space* as the union of a family of disjoint time intervals. For a subset $\mathcal{J}' \subseteq \mathcal{J}$, the time space occupied by the jobs of $\mathcal{J}'$ in σ is denoted by $\mathcal{SP}^{(\sigma)}(\mathcal{J}')$, which is formally defined as the union

of the time intervals occupied by the processing of the jobs of $\mathcal{J}'$ in σ. The time spaces $\mathcal{SP}^{(\sigma)}(\mathcal{J}_i^{(\sigma)})$, $i = 1, 2, \ldots, n$, are what we are really interested in. For convenience, we define $\mathcal{J}_0^{(\sigma)} = \emptyset$ and $\mathcal{SP}^{(\sigma)}(\mathcal{J}_0^{(\sigma)}) = \emptyset$.

From Procedure Pmtn-LS($\mathcal{O}$), we easily observe the following lemma.

Lemma 2.3. *Let σ be a feasible schedule for $\mathcal{J}$. Then $\sigma =$ Pmtn-LS($\mathcal{O}^{(\sigma)}$) if and only if, for each $i = 1, 2, \ldots, n$, the time space $[r_{\sigma(i)}, C_{\sigma(i)}(\sigma)] \setminus \mathcal{SP}^{(\sigma)}(\mathcal{J}_{i-1}^{(\sigma)})$ (which is the union of the intervals included in $[r_{\sigma(i)}, C_{\sigma(i)}(\sigma)]$ not occupied by the jobs in $\mathcal{J}_{i-1}^{(\sigma)}$ in σ) is fully occupied by $J_{\sigma(i)}$ in σ.*

Since no artificial idle time exists in a Pmtn-LS schedule, we have the following lemma.

Lemma 2.4. *Let σ and σ' be two Pmtn-LS schedules for instance $\mathcal{J}$. Let $i \in \{1, 2, \ldots, n\}$ be an index such that $\mathcal{J}_i^{(\sigma)} = \mathcal{J}_i^{(\sigma')}$. Then $\mathcal{SP}^{(\sigma)}(\mathcal{J}_i^{(\sigma)}) = \mathcal{SP}^{(\sigma')}(\mathcal{J}_i^{(\sigma')})$.*

The following lemma reveals a useful property of the schedules in $\Pi^*(\mathcal{J})$.

Lemma 2.5. *For every schedule $\sigma \in \Pi^*(\mathcal{J})$, we have $\sigma =$ Pmtn-LS($\mathcal{O}^{(\sigma)}$) and $\mathcal{O}^{(\sigma)}$ is a completion-coinciding permutation of $\mathcal{J}$.*

Proof. Since σ is an optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$, there is no artificial idle time in σ. This further implies that, for $i = 1, 2, \ldots, n$, there is no idle time in the interval $[r_{\sigma(i)}, C_{\sigma(i)}(\sigma)]$ in σ.

Suppose to the contrary that $\sigma \neq$ Pmtn-LS($\mathcal{O}^{(\sigma)}$). From Lemma 2.3, there is an index $i \in \{1, 2, \ldots, n\}$ such that the time space $[r_{\sigma(i)}, C_{\sigma(i)}(\sigma)] \setminus \mathcal{SP}^{(\sigma)}(\mathcal{J}_{i-1}^{(\sigma)})$ is not fully occupied by job $J_{\sigma(i)}$ in σ. Then $i \leq n-1$ and there is an index $i' \in \{i+1, i+2, \ldots, n\}$ such that a part of $J_{\sigma(i')}$ is scheduled in the interval $[t, t']$ with $r_{\sigma(i)} \leq t < t' < C_{\sigma(i)}(\sigma)$. From the implementation of the SRPT rule, we have $p_{\sigma(i')}^{(\sigma)}(t) \leq p_{\sigma(i)}^{(\sigma)}(t)$. This further implies that $p_{\sigma(i')}^{(\sigma)}(t') = p_{\sigma(i')}^{(\sigma)}(t) - (t' - t) < p_{\sigma(i)}^{(\sigma)}(t')$. But then, from the implementation of the SRPT rule again, we have $C_{\sigma(i')}(\sigma) < C_{\sigma(i)}(\sigma)$. This contradicts the fact that $i' > i$.

The above discussion shows that $\sigma =$ Pmtn-LS($\mathcal{O}^{(\sigma)}$), so $\mathcal{O}^{(\sigma)}$ is a completion-coinciding permutation of $\mathcal{J}$. The lemma follows. □

From Lemma 2.5, we only consider completion-coinciding permutations in the sequel.

Remark: From the previous discussion, each element $\sigma \in \Pi^*(\mathcal{J})$ has three identities: (i) σ represents an implementation of the SRPT rule on instance $\mathcal{J}$, (ii) σ is an optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$ on instance $\mathcal{J}$, and (iii) $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is a completion-coinciding permutation of $\mathcal{J}$ such that $\sigma =$ Pmtn-LS($\mathcal{O}^{(\sigma)}$). Thus, we use the three expressions σ, $\mathcal{O}^{(\sigma)}$ and $(J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ interchangeably when there is no risk of confusion.

Definition 2.1. *A job J_j is called* ***legal*** *at time $C^{(i)}(\mathcal{J})$ (with respect to instance $\mathcal{J}$), where $1 \le i \le n$, if J_j completes at time $C^{(i)}(\mathcal{J})$ in some schedule in $\Pi^*(\mathcal{J})$. We use $\mathcal{L}^{(i)}(\mathcal{J})$ to denote the set of all the jobs of $\mathcal{J}$ that are legal at time $C^{(i)}(\mathcal{J})$. Then*

$$\mathcal{L}^{(i)}(\mathcal{J}) = \{J_{\sigma(i)} : \sigma \in \Pi^*(\mathcal{J})\}.$$

We call $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$ the ***legal sets*** *of $\mathcal{J}$.*

The next lemma reveals some hereditary properties of schedules, completion times, and legal sets.

Lemma 2.6. *Consider instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, and let $\sigma \in \Pi^*(\mathcal{J})$ and $i \in \{1, 2, \ldots, n-1\}$. Then we have the following two statements.*

(i) $\mathcal{O}_i^{(\sigma)} \in \Pi^*(\mathcal{J}_i^{(\sigma)})$.

(ii) $C^{(h)}(\mathcal{J}_i^{(\sigma)}) = C^{(h)}(\mathcal{J})$ and $\mathcal{L}^{(h)}(\mathcal{J}_i^{(\sigma)}) \subseteq \mathcal{L}^{(h)}(\mathcal{J}) \cap \mathcal{J}_i^{(\sigma)}$ for each $h \in \{1, 2, \ldots, i\}$.

Proof. Let $\mathcal{O}_i' = (J_{o'(1)}, J_{o'(2)}, \ldots, J_{o'(i)})$ be a schedule in $\Pi^*(\mathcal{J}_i^{(\sigma)})$. Let $\mathcal{O}'$ be the permutation of $\mathcal{J}$ that is obtained from $\mathcal{O}^{(\sigma)}$ with $\mathcal{O}_i^{(\sigma)}$ replaced by $\mathcal{O}_i'$. Then

$$\mathcal{O}' = (J_{o'(1)}, \ldots, J_{o'(i)}, J_{\sigma(i+1)}, \ldots, J_{\sigma(n)}).$$

From the choice of $\mathcal{O}_i'$, we have

$$\mathcal{J}_i^{(\mathcal{O}')} = \mathcal{J}_i^{(\sigma)} \tag{3}$$

and

$$\sum_{j=1}^{i} C_{o'(j)}(\mathcal{O}') \le \sum_{j=1}^{i} C_{\sigma(j)}(\sigma). \tag{4}$$

From Lemma 2.4, the relation $\mathcal{J}_i^{(\mathcal{O}')} = \mathcal{J}_i^{(\sigma)}$ in (3) implies that

$$\mathcal{SP}^{(\mathcal{O}')}(\mathcal{J}_i^{(\mathcal{O}')}) = \mathcal{SP}^{(\sigma)}(\mathcal{J}_i^{(\sigma)}).$$

Note that $\mathcal{O}^{(\sigma)} = (\mathcal{O}_i^{(\sigma)}, J_{\sigma(i+1)}, J_{\sigma(i+2)}, \ldots, J_{\sigma(n)})$. From the implementations of Procedure Pmtn-LS on the two permutations $\mathcal{O}'$ and $\mathcal{O}^{(\sigma)}$, we conclude that

$$C_{o'(j)}(\mathcal{O}') = C_{\sigma(j)}(\sigma) \text{ for } j = i+1, i+2, \ldots, n. \tag{5}$$

Since $\sigma \in \Pi^*(\mathcal{J})$, we certainly have $\sum_{j=1}^{n} C_{o'(j)}(\mathcal{O}') \ge \sum_{j=1}^{n} C_{\sigma(j)}(\sigma)$. Combining (4) and (5), we obtain $\sum_{j=1}^{n} C_{o'(j)}(\mathcal{O}') = \sum_{j=1}^{n} C_{\sigma(j)}(\sigma)$ and $\sum_{j=1}^{i} C_{o'(j)}(\mathcal{O}') = \sum_{j=1}^{i} C_{\sigma(j)}(\sigma)$. Consequently, $\mathcal{O}' \in \Pi^*(\mathcal{J})$ and $\mathcal{O}_i^{(\sigma)} \in \Pi^*(\mathcal{J}_i^{(\sigma)})$. Statement (i) follows.

Now, for each $h \in \{1, 2, \ldots, i\}$, from the fact that $\mathcal{O}_i^{(\sigma)} \in \Pi^*(\mathcal{J}_i^{(\sigma)})$, we have $C^{(h)}(\mathcal{J}_i^{(\sigma)}) = C_{\sigma(h)}(\mathcal{O}_i^{(\sigma)}) = C_{\sigma(h)}(\sigma) = C^{(h)}(\mathcal{J})$. From the fact that $\mathcal{O}' \in \Pi^*(\mathcal{J})$ and the arbitrary choice of $\mathcal{O}_i' \in \Pi^*(\mathcal{J}_i^{(\sigma)})$, we further have $\mathcal{L}^{(h)}(\mathcal{J}_i^{(\sigma)}) \subseteq \mathcal{L}^{(h)}(\mathcal{J})$, so $\mathcal{L}^{(h)}(\mathcal{J}_i^{(\sigma)}) \subseteq \mathcal{L}^{(h)}(\mathcal{J}) \cap \mathcal{J}_i^{(\sigma)}$. This proves statement (ii). The lemma follows. □

2.3 A very important property

Definition 2.2. *Let $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ be a permutation of $\mathcal{J}$. $\mathcal{O}$ is called a* ***TC-optimal permutation*** *of $\mathcal{J}$ if $\mathcal{O}$ is a completion-coinciding permutation of $\mathcal{J}$ such that $\mathcal{O} \in \Pi^*(\mathcal{J})$, where "TC" stands for the "total completion time". $\mathcal{O}$ is called a* ***legal permutation*** *of $\mathcal{J}$ if $J_{o(j)} \in \mathcal{L}^{(j)}(\mathcal{J})$ for $j = 1, 2, \ldots, n$.*

Note that for each schedule $\sigma \in \Pi^*(\mathcal{J})$, $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is a legal and TC-optimal permutation of $\mathcal{J}$. We next present a very important property that shows the equivalence of legal permutations and TC-optimal permutations.

Lemma 2.7. *Let $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ be a permutation of $\mathcal{J}$. Then $\mathcal{O}$ is a TC-optimal permutation of $\mathcal{J}$ if and only if $\mathcal{O}$ is a legal permutation of $\mathcal{J}$.*

Proof. Note that each TC-optimal permutation of $\mathcal{J}$ must be a legal permutation of $\mathcal{J}$. Then we only need to show that each legal permutation of $\mathcal{J}$ is also a TC-optimal permutation of $\mathcal{J}$. We prove the result by induction on the number of jobs. When $|\mathcal{J}| = 1$, the result holds trivially.

Inductively, suppose that $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is a job instance with $n \geq 2$ and the result holds for every job instance with at most $n-1$ jobs. Let $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ be a legal permutation of $\mathcal{J}$. Then $J_{o(j)} \in \mathcal{L}^{(j)}(\mathcal{J})$ for $j = 1, 2, \ldots, n$. Let $i^* \in \{1, 2, \ldots, n\}$ such that $r_{o(i^*)} = \max\{r_1, r_2, \ldots, r_n\}$. Let $\mathcal{J}' = \mathcal{J} \setminus \{J_{o(i^*)}\}$. Then $C^{(1)}(\mathcal{J}') < C^{(2)}(\mathcal{J}') < \cdots < C^{(n-1)}(\mathcal{J}')$ are the completion times of the $n-1$ jobs of $\mathcal{J}'$ in every schedule in $\Pi^*(\mathcal{J}')$. By the induction hypothesis, we have the following claim.

Claim 1. Each legal permutation of $\mathcal{J}'$ is also a TC-optimal permutation of $\mathcal{J}'$.

Let $i_0 \in \{1, 2, \ldots, n\}$ be the minimum index such that $r_{o(i^*)} < C^{(i_0)}(\mathcal{J})$. Then $i^* \in \{i_0, i_0 + 1, \ldots, n\}$. At time $r_{o(i^*)}$, all the jobs in $\mathcal{J}$ have been released. From the principle of the SRPT rule, in every schedule in $\Pi^*(\mathcal{J})$, the jobs that are processed in the interval $[r_{o(i^*)}, C^{(n)}(\mathcal{J})]$ must be scheduled in a non-decreasing order of their remaining processing times without preemption. Note that these jobs are completed at times $C^{(i_0)}(\mathcal{J}), C^{(i_0+1)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$, respectively. Then the remaining processing times at time $r_{o(i^*)}$ in every schedule in $\Pi^*(\mathcal{J})$ are given by

$$\tilde{p}_{i_0} \leq \tilde{p}_{i_0+1} \leq \cdots \leq \tilde{p}_n, \tag{6}$$

where $\tilde{p}_{i_0} = C^{(i_0)}(\mathcal{J}) - r_{o(i^*)}$ and $\tilde{p}_i = C^{(i)}(\mathcal{J}) - C^{(i-1)}(\mathcal{J})$ for $i = i_0 + 1, i_0 + 2, \ldots, n$. Since $J_{o(i^*)}$ is processed without preemption in every schedule in $\Pi^*(\mathcal{J})$, from the fact that $J_{o(i^*)} \in \mathcal{L}^{(i^*)}(\mathcal{J})$, we further have

$$\tilde{p}_{i^*} = p_{o(i^*)}. \tag{7}$$

Given a schedule $\sigma \in \Pi^*(\mathcal{J})$, we construct a schedule σ' for $\mathcal{J}'$ as follows: In the interval $[0, r_{o(i^*)}]$, σ' has the same processing pattern as that of σ, and beginning with time $r_{o(i^*)}$, the uncompleted jobs of $\mathcal{J}'$ are scheduled consecutively in the same order as that in σ. Equivalently, σ' is obtained from σ by removing job $J_{o(i^*)}$ and eliminating the idle time interval $[C_{o(i^*)}(\sigma) - p_{o(i^*)}, C_{o(i^*)}(\sigma)]$ (if $o(i^*) \neq \sigma(n)$) by shifting the processing in the interval $[C_{o(i^*)}(\sigma), C^{(n)}(\mathcal{J})]$ left by a length of $p_{o(i^*)}$. From (6) and from the fact that $\sigma \in \Pi^*(\mathcal{J})$, the construction of σ' implies that σ' is obtained by an implementation of the SRPT rule, so $\sigma' \in \Pi^*(\mathcal{J}')$. Thus, without the processing time $\tilde{p}_{i^*} = p_{o(i^*)}$ of job $J_{o(i^*)}$, the remaining processing times at time $r_{o(i^*)}$ in σ', and in fact in every schedule in $\Pi^*(\mathcal{J}')$, are given by

$$\tilde{p}'_{i_0} \leq \tilde{p}'_{i_0+1} \leq \cdots \leq \tilde{p}'_{n-1}, \tag{8}$$

where $\tilde{p}'_i = \tilde{p}_i$ for $i \in \{i_0, i_0+1, \ldots, i^*-1\}$ and $\tilde{p}'_i = \tilde{p}_{i+1}$ for $i \in \{i^*, i^*+1, \ldots, n-1\}$. This further means that

$$C^{(i)}(\mathcal{J}') = \begin{cases} C^{(i)}(\mathcal{J}), & \text{if } i = 1, 2, \ldots, i^*-1, \\ C^{(i+1)}(\mathcal{J}) - p_{o(i^*)}, & \text{if } i = i^*, i^*+1, \ldots, n-1. \end{cases} \tag{9}$$

We have the following useful claim.

Claim 2. Let $\pi = (J_{\pi(1)}, J_{\pi(2)}, \ldots, J_{\pi(n)})$ be a completion-coinciding permutation of $\mathcal{J}$, where $\pi(i^*) = o(i^*)$. Let $\pi' = (J_{\pi(1)}, \ldots, J_{\pi(i^*-1)}, J_{\pi(i^*+1)}, \ldots, J_{\pi(n)})$, which is obtained from π by deleting job $J_{\pi(i^*)}$. Then we have the following two statements.

(i) π' is also a completion-coinciding permutation of $\mathcal{J}'$;

(ii) π is a TC-optimal permutation of $\mathcal{J}$ if and only if π' is a TC-optimal permutation of $\mathcal{J}'$.

In fact, Statement (i) of Claim 2 follows from the principle of Procedure Pmtn-LS on the two permutations π and π', together with the fact that $r_{o(i^*)} = \max\{r_1, r_2, \ldots, r_n\}$, which implies that $J_{o(i^*)} = J_{\pi(i^*)}$ is fully processed in the interval $[C_{o(i^*)}(\pi) - p_{o(i^*)}, C_{o(i^*)}(\pi)]$ in π. To prove Statement (ii), we regard π and π' as Pmtn-LS schedules, i.e., $\pi =$ Pmtn-LS(π) and $\pi' =$ Pmtn-LS(π'). Then schedule π' is obtained from schedule π by removing job $J_{o(i^*)}$ and eliminating the idle time interval $[C_{o(i^*)}(\pi) - p_{o(i^*)}, C_{o(i^*)}(\pi)]$ (if

$i^* < n$). From (6)-(9), we have that

$$\begin{aligned}
& \pi \text{ is a TC-optimal permutation of } \mathcal{J} \\
\Leftrightarrow\ & \pi \text{ is a schedule in } \Pi^*(\mathcal{J}) \\
\Leftrightarrow\ & C_{\pi(i)}(\sigma) = C^{(i)}(\mathcal{J}) \text{ for } i = 1, 2, \ldots, n \\
\Leftrightarrow\ & C_{\pi'(i)}(\sigma') = \begin{cases} C^{(i)}(\mathcal{J}), & \text{if } 1 \le i \le i^* - 1, \\ C^{(i+1)}(\mathcal{J}) - p_{o(i^*)}, & \text{if } i^* \le i \le n-1 \end{cases} \\
\Leftrightarrow\ & C_{\pi'(i)}(\sigma') = C^{(i)}(\mathcal{J}') \text{ for } i = 1, 2, \ldots, n-1 \\
\Leftrightarrow\ & \pi' \text{ is a schedule in } \Pi^*(\mathcal{J}') \\
\Leftrightarrow\ & \pi' \text{ is a TC-optimal permutation of } \mathcal{J}'.
\end{aligned}$$

This proves Statement (ii) and Claim 2 follows.

We next prove another useful claim.

Claim 3. For each index $i \in \{1, 2, \ldots, n\} \setminus \{i^*\}$, there is a schedule $\sigma_i \in \Pi^*(\mathcal{J})$ such that $\sigma_i(i) = o(i)$ and $\sigma_i(i^*) = o(i^*)$.

To prove Claim 3, we fix an index $i \in \{1, 2, \ldots, n\} \setminus \{i^*\}$. Since $J_{o(i)} \in \mathcal{L}^{(i)}(\mathcal{J})$, there is a schedule $\pi_i \in \Pi^*(\mathcal{J})$ such that $\pi_i(i) = o(i)$. If $\pi_i(i^*) = o(i^*)$, by setting $\sigma_i = \pi_i$, we are done. Then we suppose in the following that $\pi_i(i^*) \neq o(i^*)$.

Let $i' \in \{1, 2, \ldots, n\}$ be the index such that $\pi_i(i') = o(i^*)$. Then $i' \notin \{i, i^*\}$ and $i' \ge i_0$. Since $J_{o(i^*)}$ is processed without preemption in every schedule in $\Pi^*(\mathcal{J})$, we have

$$\tilde{p}_{i'} = p_{o(i^*)}. \tag{10}$$

From (7)-(10), we have

$$\tilde{p}_h = p_{o(i^*)} \text{ for each index } h \text{ with } \min\{i', i^*\} \le h \le \max\{i', i^*\}. \tag{11}$$

The relation in (11) implies that $\tilde{p}_{i'} = \tilde{p}_{i^*} = p_{o(i^*)}$, i.e., the two remaining parts (or jobs) completed at times $C^{(i')}(\mathcal{J})$ and $C^{(i^*)}(\mathcal{J})$, respectively, in π_i have the same length at the maximum release date $r_{o(i^*)}$ in π_i. Thus, we define σ_i as the schedule obtained from π_i by exchanging the above two parts (or jobs). The exchange is performed after the maximum release date $r_{o(i^*)}$. Then σ_i is a feasible schedule for $\mathcal{J}$ such that $C_{\sigma_i(h)}(\sigma_i) = C_{\pi_i(h)}(\pi_i) = C^{(h)}(\mathcal{J})$ for $h = 1, 2, \ldots, n$. Consequently, we have $\sigma_i \in \Pi^*(\mathcal{J})$. Now Claim 3 follows by noting that $\sigma_i(i) = o(i)$ and $\sigma_i(i^*) = o(i^*)$.

Consider an arbitrary index $i \in \{1, 2, \ldots, n\} \setminus \{i^*\}$. From Claim 3, there is a TC-optimal permutation $\sigma_i = (J_{\sigma_i(1)}, J_{\sigma_i(2)}, \ldots, J_{\sigma_i(n)})$ of $\mathcal{J}$ such that $\sigma_i(i) = o(i)$ and $\sigma_i(i^*) =$

$o(i^*)$. From Claim 2(ii), $\sigma'_i = (J_{\sigma_i(1)}, \ldots, J_{\sigma_i(i^*-1)}, J_{\sigma_i(i^*+1)}, \ldots, J_{\sigma_i(n)})$ is a TC-optimal permutation of $\mathcal{J}'$. By writing $\sigma'_i = (J_{\sigma'_i(1)}, J_{\sigma'_i(2)}, \ldots, J_{\sigma'_i(n-1)})$, we have $\sigma_i(i) = \sigma'_i(i)$ if $i \in \{1, 2, \ldots, i^* - 1\}$ and $\sigma_i(i) = \sigma'_i(i-1)$ if $i \in \{i^* + 1, i^* + 2, \ldots, n\}$. Since $\sigma'_i \in \Pi^*(\mathcal{J}')$ and $o(i) = \sigma_i(i)$, we further have

$$J_{o(i)} \in \begin{cases} \mathcal{L}^{(i)}(\mathcal{J}'), & \text{if } i \in \{1, 2, \ldots, i^* - 1\}, \\ \mathcal{L}^{(i-1)}(\mathcal{J}'), & \text{if } i \in \{i^* + 1, i^* + 2, \ldots, n\}. \end{cases} \tag{12}$$

Now let $\mathcal{O}' = (J_{o'(1)}, J_{o'(2)}, \ldots, J_{o'(n-1)})$, where

$$o'(h) = \begin{cases} o(h), & \text{if } h \in \{1, 2, \ldots, i^* - 1\}, \\ o(h+1), & \text{if } h \in \{i^*, i^* + 1, \ldots, n-1\}. \end{cases} \tag{13}$$

Then $\mathcal{O}' = (J_{o(1)}, \ldots, J_{o(i^*-1)}, J_{o(i^*+1)}, \ldots, J_{o(n)})$ is a permutation of $\mathcal{J}'$. From (12) and (13), we have $J_{o'(h)} \in \mathcal{L}^{(h)}(\mathcal{J}')$ for $h = 1, 2, \ldots, n-1$. Then $\mathcal{O}'$ is a legal permutation of $\mathcal{J}'$. From Claim 1 (the induction hypothesis), $\mathcal{O}'$ is a TC-optimal permutation of $\mathcal{J}'$. By using Claim 2(ii) again, we conclude that $\mathcal{O}$ is a TC-optimal permutation of $\mathcal{J}$. The lemma follows. □

2.4 Standard instances

For each $\tau_i \in \mathcal{T}(\mathcal{J})$, we define $\mathcal{F}(\tau_i) = \{J_j \in \mathcal{J} : r_j = \tau_i\}$, and define $p_{\min}(\tau_i)$ and $p_{\max}(\tau_i)$ as the smallest and largest remaining processing time at τ_i, respectively. For the case where $\mathcal{F}(\tau_i) \neq \emptyset$, we clearly have $p_{\min}(\tau_i) \leq p_j \leq p_{\max}(\tau_i)$ for all the jobs $J_j \in \mathcal{F}(\tau_i)$.

An instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is called *standard* if, for every job $J_j \in \mathcal{J}$, we have $p_j = p_{\min}(r_j)$. Clearly, in a standard instance, the jobs with a common release date must have the same processing times, i.e., if $\mathcal{F}(\tau_i) \neq \emptyset$, then $p_j = p_{\min}(\tau_i)$ for all $J_j \in \mathcal{F}(\tau_i)$.

Not all the instances are standard. But, given an instance $\mathcal{J}$, we can use the following procedure, denoted as *Procedure RD-Modification*, to obtain a standard instance $\mathcal{J}'$ equivalent to $\mathcal{J}$, in our research context, where "RD" stands for the "release date".

Procedure RD-Modification: *Do the following on instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$.*

(i) Apply the SRPT rule to the jobs in instance $\mathcal{J}$ to obtain a schedule $\sigma \in \Pi^*(\mathcal{J})$ and the set of decision times $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_{n'}\}$ such that $\tau_1 < \tau_2 < \cdots < \tau_{n'}$.

(ii) Set $r'_j := r_j$ for $j = 1, 2, \ldots, n$.

(iii) For $i = 1, 2, \ldots, n'$, do the following: For every job J_j with $r'_j = \tau_i$ and $p_j > p_{\min}(\tau_i)$ (if any), reset $r'_j := \tau_{i+1}$, where $p_{\min}(\tau_i)$ can be obtained from σ directly.

(iv) For each $j = 1, 2, \ldots, n$, define J'_j as a job with release date r'_j and processing time p_j.

(v) Output the instance $\mathcal{J}' = \{J'_1, J'_2, \ldots, J'_n\}$. *We call* $\mathcal{J}'$ *the* $\mathcal{J}$***-standardization***.

Note that the $\mathcal{J}$-standardization is uniquely determined by $\mathcal{J}$. The core part of Procedure RD-Modification is its Step (iii). The principle used in this step is comprehensible. In fact, if $\tau_i \in \mathcal{T}(\mathcal{J})$ and $J_j \in \mathcal{J}$ is a job released at time τ_i such that $p_j > p_{\min}(\tau_i)$, the SRPT rule stipulates that, in every schedule in $\Pi^*(\mathcal{J})$, J_j cannot be processed before time τ_{i+1}. Thus, resetting the release date of J_j as τ_{i+1} will not change the implementations of the SRPT rule and the set $\Pi^*(\mathcal{J})$. That is, by identifying the two jobs J_j and J'_j, we clearly have $\Pi^*(\mathcal{J}') = \Pi^*(\mathcal{J})$ and $\mathcal{C}(\mathcal{J}') = \mathcal{C}(\mathcal{J})$. Thus, Procedure RD-Modification works correctly. Moreover, Step (iii) runs in $O(n^2)$ time since it has $n' \leq 2n$ iterations and, in each iteration, at most n jobs are released. Then we have the following lemma.

Lemma 2.8. *Given an instance* $\mathcal{J}$*, Procedure RD-Modification generates an equivalent and standard instance* $\mathcal{J}'$ *in* $O(n^2)$ *time.*

The next lemma reveals the *hereditary property* of standard instances.

Lemma 2.9. *Let* $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ *be a standard instance and let* $J_{j'} \in \mathcal{L}^{(n)}(\mathcal{J})$*, i.e.,* $J_{j'} = J_{\sigma(n)}$ *for some schedule* $\sigma \in \Pi^*(\mathcal{J})$*. Then* $\mathcal{J} \setminus \{J_{j'}\}$ *is also a standard instance.*

Proof. Let $\sigma' = \mathcal{O}_{n-1}^{(\sigma)}$. From Lemma 2.6, we have $\sigma' \in \Pi^*(\mathcal{J}_{n-1}^{(\sigma)})$. Note that σ' can be obtained from σ by removing the schedule for $J_{\sigma(n)}$. For each $j \in \{\sigma(1), \sigma(2), \ldots, \sigma(n-1)\}$, since $\mathcal{J}$ is a standard instance, p_j is the smallest remaining processing time at time r_j in σ, so p_j is the smallest remaining processing time at time r_j in σ'; thus, p_j is the smallest remaining processing time at time r_j in every schedule in $\Pi^*(\mathcal{J}_{n-1}^{(\sigma)})$. Consequently, $\mathcal{J} \setminus \{J_{j'}\} = \mathcal{J}_{n-1}^{(\sigma)}$ is a standard instance. The lemma follows. □

If $\mathcal{J}'$ is the $\mathcal{J}$-standardization, then we have $\mathcal{T}(\mathcal{J}') \subseteq \mathcal{T}(\mathcal{J})$. It is noticed that the standard instances enable us to simplify the discussions in Sections 3 and 4.

3 Generating the legal sets

Given a standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$, we present in this section an $O(n^2)$-time algorithm to generate all the legal sets of $\mathcal{J}$. Recall that $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_{n'}\}$ is the set of all the decision times, where $\tau_1 < \tau_2 < \cdots < \tau_{n'}$. Since $\mathcal{J}$ is a standard instance, for each decision time τ_i with $\mathcal{F}(\tau_i) \neq \emptyset$, we have

$$p_j = p_{\min}(\tau_i) \text{ for } J_j \in \mathcal{F}(\tau_i). \tag{14}$$

Thus, for every positive number q, we have

$$\{J_j : r_j = \tau_i, p_j = q\} = \begin{cases} \mathcal{F}(\tau_i), & \text{if } q = p_{\min}(\tau_i), \\ \emptyset, & \text{otherwise.} \end{cases} \tag{15}$$

We next introduce some definitions.

Definition 3.1. *Let i be an index in $\{1, 2, \ldots, n'\}$.*

*(i) For a schedule $\sigma \in \Pi^*(\mathcal{J})$ and a job $J_z \in \mathcal{J}$, J_z is said to be (τ_i, σ)-**available** if J_z is available at time τ_i in σ, i.e., $r_z \leq \tau_i < C_z(\sigma)$.*

*(ii) We use K_i to denote the set of distinct remaining processing times at time τ_i and set $k_i = |K_i|$. Then k_i is the number of distinct remaining processing times at time τ_i. When $k_i > 0$, we clearly have $p_{\min}(\tau_i) = \min\{q : q \in K_i\}$. We use λ_i to denote the repetition number of the remaining processing time $p_{\min}(\tau_i)$ at time τ_i and call λ_i the **optional index** at time τ_i. For the case where $k_i = 0$, we define $p_{\min}(\tau_i) = +\infty$ and $\lambda_i = 0$. This means that, at time τ_i, the SRPT rule has λ_i choices for processing a job in the interval $[\tau_i, \tau_{i+1}]$.*

(iii) For a schedule $\sigma \in \Pi^*(\mathcal{J})$ and a positive number q, we use $\mathcal{J}^{(\sigma)}(i, q)$ to denote the set of (τ_i, σ)-available jobs that have the same remaining processing time q at time τ_i in σ. Note that $\mathcal{J}^{(\sigma)}(i, q) \neq \emptyset$ if $q \in K_i$ and $\mathcal{J}^{(\sigma)}(i, q) = \emptyset$ if $q \notin K_i$. Moreover, we have

$$\lambda_i = \begin{cases} |\mathcal{J}^{(\sigma)}(i, p_{\min}(\tau_i))|, & \text{if } k_i > 0, \\ 0, & \text{if } k_i = 0. \end{cases}$$

*(iv) For each $q \in K_i$, we define $\mathcal{J}(i, q) = \bigcup_{\sigma \in \Pi^*(\mathcal{J})} \mathcal{J}^{(\sigma)}(i, q)$ and call it the q-**cluster** at time τ_i.*

(v) We finally define $\tau_0 = -\infty$, $K_0 = \emptyset$, and $k_0 = 0$.

Note that $\tau_1 = r_1$ is the minimum release date of the jobs in $\mathcal{J}$ and $\tau_{n'} = C^{(n)}(\mathcal{J})$. Then we have $k_1 = 1$, $k_{n'-1} = 1$, $k_{n'} = 0$, and $\tau_1 \notin \mathcal{C}(\mathcal{J})$. From Definition 3.1, the following lemma can be observed.

Lemma 3.1. *For each index $i \in \{1, 2, \ldots, n'\}$, we have the following statements.*

(i) $0 \leq |K_i| = k_i \leq n$;

(ii) $k_i = 0$ if and only if $K_i = \emptyset$ and $\lambda_i = 0$.

(iii) If $k_i \geq 1$, then $1 \leq i \leq n' - 1$ and $\mathcal{F}(\tau_i) \subseteq \mathcal{J}^{(\sigma)}(i, p_{\min}(\tau_i)) \subseteq \mathcal{J}(i, p_{\min}(\tau_i))$, where $\sigma \in \Pi^*(\mathcal{J})$.

(iv) If $k_i = 0$ and $i \leq n' - 2$, then $k_{i+1} = 1$ and $\mathcal{F}(\tau_{i+1}) = \mathcal{J}(i + 1, p_{\min}(\tau_{i+1}))$ is the unique cluster at time τ_{i+1}.

The following lemma is a direct consequence of Definition 3.1 and implementations of the SRPT rule.

Lemma 3.2. *Given $i \in \{2, 3, \ldots, n'\}$, the following statements are equivalent.*

(i) $\tau_i \in \mathcal{C}(\mathcal{J})$;

(ii) $\tau_i - \tau_{i-1} = p_{\min}(\tau_{i-1})$;
(iii) $\mathcal{L}^{(h)}(\mathcal{J}) = \mathcal{J}(i-1, p_{\min}(\tau_{i-1}))$, *where* $h \in \{1, 2, \ldots, n\}$ *such that* $\tau_i = \mathcal{C}^{(h)}(\mathcal{J})$.

The next lemma shows how to obtain the clusters at time τ_i from that at time τ_{i-1}.

Lemma 3.3. *For* $i \in \{1, 2, \ldots, n'-1\}$ *with* $k_i > 0$ *and for* $q \in K_i$, *we have the following four statements for the* q*-cluster* $\mathcal{J}(i, q)$ *at time* τ_i.

(i) If $q \neq p_{\min}(\tau_{i-1})$ *and* $q \neq p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})$, *then*

$$\mathcal{J}(i, q) = \mathcal{J}(i-1, q) \cup \{J_z : r_z = \tau_i, p_z = q\}.$$

(ii) If $q = p_{\min}(\tau_{i-1})$ *and* $\lambda_{i-1} > 1$, *then*

$$\mathcal{J}(i, q) = \mathcal{J}(i-1, q) \cup \{J_z : r_z = \tau_i, p_z = q\}.$$

(iii) If $q = p_{\min}(\tau_{i-1})$ *and* $\lambda_{i-1} = 1$, *then*

$$\mathcal{J}(i, q) = \{J_z : r_z = \tau_i, p_z = q\}.$$

(iv) If $q = p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})$, *then*

$$\mathcal{J}(i, q) = \mathcal{J}(i-1, p_{\min}(\tau_{i-1})) \cup \{J_z : r_z = \tau_i, p_z = q\}.$$

Proof. Suppose first that $k_{i-1} = 0$. Then $p_{\min}(\tau_{i-1}) = +\infty$ and we only need to prove Statement (i). From Lemma 3.1, we have $k_i = 1$ and $\mathcal{F}(\tau_i)$ is the unique cluster at time τ_i. This means that $q = p_{\min}(\tau_i)$. From Definition 3.1, the assumption $k_{i-1} = 0$ implies that $\mathcal{J}(i-1, q) = \emptyset$. Consequently, $\mathcal{J}(i, q) = \mathcal{F}(\tau_i) = \{J_z : r_z = \tau_i, p_z = q\} = \mathcal{J}(i-1, q) \cup \{J_z : r_z = \tau_i, p_z = q\}$, as required in Statement (i).

Assume in the following that $k_{i-1} > 0$. We partition the set $\mathcal{J}(i, q)$ into two parts $\mathcal{J}'(i, q)$ and $\mathcal{J}''(i, q)$, where

$$\mathcal{J}'(i, q) = \{J_z \in \mathcal{J}(i, q) : r_z < \tau_i\}$$

and

$$\mathcal{J}''(i, q) = \{J_z \in \mathcal{J}(i, q) : r_z = \tau_i\}.$$

Clearly, we have

$$\mathcal{J}''(i, q) = \{J_z : r_z = \tau_i, p_z = q\}.$$

Then we depict the jobs in $\mathcal{J}'(i,q)$ in the sequel. Since τ_{i-1} and τ_i are two consecutive decision times, from the definition of $\mathcal{J}'(i,q)$, we have the following claim.

Claim 1. $J_z \in \mathcal{J}'(i,q)$ if and only if there is a schedule $\sigma \in \Pi^*(\mathcal{J})$ such that J_z is (τ_{i-1},σ)-available, and $J_z \in \mathcal{J}^{(\sigma)}(i,q)$ if and only if $r_z \leq \tau_{i-1}$ and there is a schedule $\sigma \in \Pi^*(\mathcal{J})$ such that $J_z \in \mathcal{J}^{(\sigma)}(i,q)$.

Corresponding to the four statements of this lemma, we consider the following four cases, respectively. Since $k_{i-1} > 0$, in every schedule $\sigma \in \Pi^*(\mathcal{J})$, the interval $[\tau_{i-1},\tau_i]$ is occupied by some job in $\mathcal{J}^{(\sigma)}(i-1,p_{\min}(\tau_{i-1}))$. This implies that $p_{\min}(\tau_{i-1}) \geq \tau_i - \tau_{i-1}$.

Case 1. $q \neq p_{\min}(\tau_{i-1})$ and $q \neq p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})$. In this case, for each $\sigma \in \Pi^*(\mathcal{J})$ and each $J_z \in \mathcal{J}$ with $r_z < \tau_i$, job J_z has the remaining processing time q at time τ_i in σ, i.e., $J_z \in \mathcal{J}^{(\sigma)}(i,q)$, if and only if J_z has the remaining processing time q at time τ_{i-1} in σ, i.e., $J_z \in \mathcal{J}^{(\sigma)}(i-1,q)$. From Claim 1, $J_z \in \mathcal{J}'(i,q)$ if and only if $J_z \in \mathcal{J}(i-1,q)$, so $\mathcal{J}'(i,q) = \mathcal{J}(i-1,q)$. Consequently, $\mathcal{J}(i,q) = \mathcal{J}(i-1,q) \cup \{J_z : r_z = \tau_i, p_z = q\}$. Statement (i) follows.

Case 2. $q = p_{\min}(\tau_{i-1})$ and $\lambda_{i-1} > 1$. In this case, for each $\sigma \in \Pi^*(\mathcal{J})$ and each $J_z \in \mathcal{J}$ with $r_z < \tau_i$, the relation $J_z \in \mathcal{J}^{(\sigma)}(i,q)$ holds only if J_z has the remaining processing time q at time τ_{i-1} in σ, i.e., $J_z \in \mathcal{J}^{(\sigma)}(i-1,q)$. From Claim 1, $J_z \in \mathcal{J}'(i,q)$ only if $J_z \in \mathcal{J}(i-1,q)$. This implies that $\mathcal{J}'(i,q) \subseteq \mathcal{J}(i-1,q)$.

To prove the opposite inclusion relationship, we pick $J_z \in \mathcal{J}(i-1,q)$ arbitrarily. Then there is a schedule $\sigma \in \Pi^*(\mathcal{J})$ such that $J_z \in \mathcal{J}^{(\sigma)}(i-1,q)$, i.e., J_z has the remaining processing time q at time τ_{i-1} in σ. Since $q = p_{\min}(\tau_{i-1})$ and $|\mathcal{J}^{(\sigma)}(i-1,q)| = \lambda_{i-1} > 1$, the SRPT rule has multiple choices for processing the jobs in $\mathcal{J}^{(\sigma)}(i-1,q)$ at time τ_{i-1}. Therefore, there is a schedule $\sigma' \in \Pi^*(\mathcal{J})$ such that $J_z \in \mathcal{J}^{(\sigma')}(i-1,q)$ and the interval $[\tau_{i-1},\tau_i]$ is not occupied by job J_z in σ'. Then J_z has the remaining processing time q at time τ_i in σ'. As a result, we have $J_z \in \mathcal{J}'(i,q)$. It follows that $\mathcal{J}(i-1,q) \subseteq \mathcal{J}'(i,q)$.

The above discussion shows that $\mathcal{J}'(i,q) = \mathcal{J}(i-1,q)$. Consequently, $\mathcal{J}(i,q) = \mathcal{J}(i-1,q) \cup \{J_z : r_z = \tau_i, p_z = x\}$. Statement (ii) follows.

Case 3. $q = p_{\min}(\tau_{i-1})$ and $\lambda_{i-1} = 1$. In this case, let us consider an arbitrary schedule $\sigma \in \Pi^*(\mathcal{J})$. Since $q = p_{\min}(\tau_{i-1})$ and $|\mathcal{J}^{(\sigma)}(i-1,q)| = \lambda_{i-1} = 1$, the unique job in $\mathcal{J}^{(\sigma)}(i-1,q)$ occupies the interval $[\tau_{i-1},\tau_i]$ in σ. Thus, no job in $\mathcal{J}^{(\sigma)}(i,q)$ is released by time τ_{i-1}. This implies that $\mathcal{J}'(i,q) = \emptyset$. Consequently, $\mathcal{J}(i,q) = \{J_z : r_z = \tau_i, p_z = q\}$. Statement (iii) follows.

Case 4. $q = p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})$. In this case, for each $\sigma \in \Pi^*(\mathcal{J})$ and each $J_z \in \mathcal{J}$ with $r_z < \tau_i$, the relation $J_z \in \mathcal{J}^{(\sigma)}(i,q)$ holds only if J_z has the remaining processing time $p_{\min}(\tau_{i-1})$ at time τ_{i-1} in σ and J_z occupies the interval $[\tau_{i-1},\tau_i]$ in σ, implying that $J_z \in \mathcal{J}^{(\sigma)}(i-1,p_{\min}(\tau_{i-1}))$. From Claim 1, $J_z \in \mathcal{J}'(i,q)$ only if $J_z \in \mathcal{J}(i-1,p_{\min}(\tau_{i-1}))$.

This means that $\mathcal{J}'(i,q) \subseteq \mathcal{J}(i-1, p_{\min}(\tau_{i-1}))$.

To prove the opposite inclusion relationship, we pick $J_z \in \mathcal{J}(i-1, p_{\min}(\tau_{i-1}))$ arbitrarily. Then there is a schedule $\sigma \in \Pi^*(\mathcal{J})$ such that $J_z \in \mathcal{J}^{(\sigma)}(i-1, p_{\min}(\tau_{i-1}))$, i.e., J_z has the minimum remaining processing time $p_{\min}(\tau_{i-1})$ at time τ_{i-1} in σ. From the implementation of the SRPT rule, there is a schedule $\sigma' \in \Pi^*(\mathcal{J})$ such that $J_z \in \mathcal{J}^{(\sigma')}(i-1, p_{\min}(\tau_{i-1}))$ and the interval $[\tau_{i-1}, \tau_i]$ is occupied by job J_z in σ'. Then J_z has the remaining processing time $q = p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})$ at time τ_i in σ'. Therefore, we have $J_z \in \mathcal{J}'(i,q)$. It follows that $\mathcal{J}(i-1, p_{\min}(\tau_{i-1})) \subseteq \mathcal{J}'(i,q)$.

The above discussion shows that $\mathcal{J}'(i,q) = \mathcal{J}(i-1, p_{\min}(\tau_{i-1}))$. Consequently, $\mathcal{J}(i,q) = \mathcal{J}(i-1, p_{\min}(\tau_{i-1})) \cup \{J_z : r_z = \tau_i, p_z = q\}$. Statement (iv) follows and the lemma follows. □

From Lemmas 3.2 and 3.3, we present the following algorithm for determining the legal sets $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$.

Algorithm 3.1. *For generating the legal sets* $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$.

Input: An standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \le r_2 \le \cdots \le r_n$.
Step 1: Apply the SRPT rule to the jobs in instance $\mathcal{J}$ to obtain a schedule $\sigma \in \Pi^*(\mathcal{J})$, together with the corresponding completion times $C_{\sigma(1)}(\sigma), C_{\sigma(2)}(\sigma), \ldots, C_{\sigma(n)}(\sigma)$.
Step 2: From schedule σ, do the following:

(2.1) Set $C^{(h)}(\mathcal{J}) := C_{\sigma(h)}(\sigma)$ for $h = 1, 2, \ldots, n$. Generate the set of decision times $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_{n'}\}$, where $\tau_1 < \tau_2 < \cdots < \tau_{n'}$.

(2.2) Determine the index sequence $i_1, i_2, \ldots, i_n$ of $\{1, 2, \ldots, n'\}$ such that $i_1 < i_2 < \cdots < i_n$ and

$$\tau_{i_h} = C^{(h)}(\mathcal{J}) \text{ for } h = 1, 2, \ldots, n.$$

(2.3) Determine the items $p_{\min}(\tau_i)$, λ_i, k_i, and K_i for $i = 1, 2, \ldots, n'$.

(2.4) For each $i \in \{1, 2, \ldots, n'\}$ and $q \in K_i$, calculate $\mathcal{J}''(i,q) := \{J_z : r_z = \tau_i, p_z = q\}$.
Step 3: Set $k_0 := 0$. For $i = 1, 2, \ldots, n'$ with $k_i > 0$, generate the clusters at time τ_i in the following way:

– For each $q \in K_i$, set $\mathcal{J}(i,q) :=$

$$\begin{cases} \mathcal{J}(i-1,q) \cup \mathcal{J}''(i,q), & \text{if } q \ne p_{\min}(\tau_{i-1}) \text{ and } q \ne p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1}), \\ \mathcal{J}(i-1,q) \cup \mathcal{J}''(i,q), & \text{if } q = p_{\min}(\tau_{i-1}) \text{ and } \lambda_{i-1} > 1, \\ \mathcal{J}''(i,q), & \text{if } q = p_{\min}(\tau_{i-1}) \text{ and } \lambda_{i-1} = 1, \\ \mathcal{J}(i-1, p_{\min}(\tau_{i-1})) \cup \mathcal{J}''(i,q), & \text{if } q = p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1}). \end{cases}$$

Step 4: For $h = 1, 2, \ldots, n$, set $\mathcal{L}^{(h)}(\mathcal{J}) := \mathcal{J}(i_h - 1, p_{\min}(\tau_{i_h - 1}))$.
Output: $C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$ and $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$.

Lemma 3.4. *Algorithm 3.1 generates the n legal sets $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$ correctly in $O(n^2)$ time.*

Proof. The correctness of Algorithm 3.1 follows from Lemmas 3.2 and 3.3. We roughly estimate the time complexity of the algorithm in the following.

Clearly, Steps 1, 2 and 4 run in $O(n^2)$ time in total. Then we only need to estimate the running time of Step 3.

In Step 3, we have a total of $n' \leq 2n$ iterations and, for each $i \in \{1, 2, \ldots, n'\}$ with $k_i > 0$, the $|K_i| = k_i$ clusters at time τ_i, i.e., $\mathcal{J}(i, q)$ for all $q \in K_i$, are generated in the i-th iteration. We distinguish the following two cases.

If $k_{i-1} = 0$, then $|K_i| = k_i = 1$ and the only cluster at time τ_i is $\mathcal{J}(i, p_{\min}(\tau_i)) = \mathcal{F}(\tau_i)$. Thus, the i-th iteration runs in $O(n)$ time.

If $k_{i-1} \geq 1$, then for each $q \in K_{i-1} \setminus \{p_{\min}(\tau_i), p_{\min}(\tau_{i-1})\}$, we have $q \in K_i$ and $\mathcal{J}(i, q) = \mathcal{J}(i-1, q)$. This means that $K_i \setminus K_{i-1} \subseteq \{p_{\min}(\tau_i), p_{\min}(\tau_{i-1}), p_{\min}(\tau_{i-1}) - (\tau_i - \tau_{i-1})\}$, so $|K_i \setminus K_{i-1}| \leq 3$. Then at most three clusters $\mathcal{J}(i, q)$ with $q \in K_i \setminus K_{i-1}$ are newly generated at time τ_i. Note that $|K_i| = k_i \leq n$ and each new cluster, which contains at most n jobs, can be generated in $O(n)$ time. As a result, the i-th iteration runs in $O(n)$ time.

The above discussion shows that Step 3 runs in $O(n^2)$ time. Then the time complexity of Algorithm 3.1 is $O(n^2)$. □

Sometimes, we only need the last legal set $\mathcal{L}^{(n)}(\mathcal{J})$. The following lemma states the feature of $\mathcal{L}^{(n)}(\mathcal{J})$. Recall that $\tau_{n'} = C^{(n)}(\mathcal{J})$, so $k_{n'-1} = 1$.

Lemma 3.5. *Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$ be a standard instance. Let $j^* = \min\{j : J_j \in \mathcal{L}^{(n)}(\mathcal{J})\}$. Then $\mathcal{L}^{(n)}(\mathcal{J}) = \{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\}$.*

Proof. Note that the condition $p_j = p_{\max}(r_j)$ implies $p_{\min}(r_j) = p_j = p_{\max}(r_j)$ since we have $p_j = p_{\min}(r_j)$ for all the jobs $J_j \in \mathcal{J}$ in the standard instance $\mathcal{J}$.

Suppose first that $J_j \in \mathcal{L}^{(n)}(\mathcal{J})$. Then there is a schedule $\pi \in \Pi^*(\mathcal{J})$ such that $J_j = J_{\pi(n)}$. By the SRPT rule, at every decision time $\tau_i \geq r_j$, J_j has the largest remaining processing time in π, i.e., $p_j^{(\pi)}(\tau_i) = p_{\max}(\tau_i)$. Therefore, $p_j = p_j^{(\pi)}(r_j) = p_{\max}(r_j)$. Moreover, the choice of J_{j^*} implies that $j \geq j^*$. Consequently, we have $\mathcal{L}^{(n)}(\mathcal{J}) \subseteq \{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\}$. This also shows that $p_{j^*} = p_{\max}(r_{j^*})$.

In order to prove the opposite inclusion relationship, we pick an arbitrary job $J_j \in \mathcal{J}$ such that $j \geq j^*$ and $p_j = p_{\max}(r_j)$. Then $r_j \geq r_{j^*}$. Since $J_{j^*} \in \mathcal{L}^{(n)}(\mathcal{J})$, there is a schedule $\sigma \in \Pi^*(\mathcal{J})$ such that $J_{j^*} = J_{\sigma(n)}$. Suppose that $J_j = J_{\sigma(h)}$ for some $h \in \{1, 2, \ldots, n\}$. If $h = n$, we have $J_j = J_{\sigma(h)} = J_{j^*} \in \mathcal{L}^{(n)}(\mathcal{J})$. Then we suppose that $h \leq n-1$, so $J_j \neq J_{j^*}$. By the SRPT rule again, at the decision time $r_j \geq r_{j^*}$, J_{j^*} has the largest remaining

processing time in σ. Therefore, $p_{j^*}^{(\sigma)}(r_j) = p_{\max}(r_j) = p_j$; consequently, $J_{j^*}^{(\sigma)}(r_j)$ and J_j can be regarded as two identical jobs at time r_j. Let σ' be the schedule obtained from σ by swapping J_j and $J_{j^*}^{(\sigma)}(r_j)$. Then the two schedules σ and σ' have the same completion times, so $\sigma' \in \Pi^*(\mathcal{J})$ since $\sigma \in \Pi^*(\mathcal{J})$. Noting that $J_j = J_{\sigma'(n)}$, we conclude that $J_j \in \mathcal{L}^{(n)}(\mathcal{J})$. This shows that $\{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\} \subseteq \mathcal{L}^{(n)}(\mathcal{J})$. The lemma follows. □

Based on Lemma 3.5, we present the following algorithm for generating the last legal set of instance $\mathcal{J}$.

Algorithm 3.2. *For generating the last legal set $\mathcal{L}^{(n)}(\mathcal{J})$.*

Input: A standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$.
Step 1: Apply the following modified SRPT rule to the jobs in instance $\mathcal{J}$ to obtain a schedule $\sigma \in \Pi^*(\mathcal{J})$ and the values $p_{\max}(r_j)$ for $j \in \{1, 2, \ldots, n\}$: *At each decision time, we schedule an available job (if any) with the largest job index. This procedure is repeated until all the jobs are scheduled.*
Step 2: Set $j^* := \sigma(n)$ and set $\mathcal{L}^{(n)}(\mathcal{J}) := \{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\}$.
Output: The set $\mathcal{L}^{(n)}(\mathcal{J})$.

Lemma 3.6. *For a standard instance $\mathcal{J}$, Algorithm 3.2 generates $\mathcal{L}^{(n)}(\mathcal{J})$ in $O(n)$ time.*

Proof. Recall that the jobs are initially indexed in the ERD order such that $r_1 \leq r_2 \leq \cdots \leq r_n$. For each decision time τ_i with $\mathcal{F}(\tau_i) \neq \emptyset$, the jobs in $\mathcal{F}(\tau_i)$ have the same processing times $p_{\min}(\tau_i)$ and we have $\mathcal{F}(\tau_i) = \{J_{y_i}, J_{y_i+1}, \ldots, J_{z_i}\}$, where $y_i = \min\{j : J_j \in \mathcal{F}(\tau_i)\}$ and $z_i = \max\{j : J_j \in \mathcal{F}(\tau_i)\}$. Then we set $\overrightarrow{\mathcal{F}}(\tau_i) = (J_{y_i}, J_{y_i+1}, \ldots, J_{z_i})$, which lists the jobs in $\mathcal{F}(\tau_i)$ in an increasing order of their job indices. For the case where $\mathcal{F}(\tau_i) = \emptyset$, we set $\overrightarrow{\mathcal{F}}(\tau_i)$ as empty.

Let us first consider the implementation of Step 1. We use a simple data structure in the following discussion. At each decision time τ_i with $k_i > 0$, we store the available jobs in an increasing order of their job indices in a list, denoted as $\mathbf{List}(\tau_i)$. We also define $\mathbf{List}(\tau_i)$ as empty if $k_i = 0$. We next show that all the lists $\mathbf{List}(\tau_i)$, $i \in \{1, 2, \ldots, n'\}$, have the *desired property*: $\mathbf{List}(\tau_i)$ can be generated from $\mathbf{List}(\tau_{i-1})$ in $O(1 + |\mathcal{F}(\tau_i)|)$ time, and if $k_i > 0$, the available jobs at time τ_i in σ are also listed in a non-increasing order of their remaining processing times in $\mathbf{List}(\tau_i)$.

At time $\tau_1 = r_1$, we have $k_1 = 1$ and the available jobs are given by $\mathcal{F}(\tau_1) = \mathcal{F}(r_1)$. Then $\mathbf{List}(\tau_1) = (J_1, J_2, \ldots, J_{z_1})$ can be obtained in $O(1 + |\mathcal{F}(\tau_1)|)$ time clearly. Since all the jobs in $\mathcal{F}(\tau_1)$ have the same processing time $p_{\min}(\tau_1)$, $\mathbf{List}(\tau_1)$ has the desired property.

Inductively, suppose that $i \in \{2, 3, \ldots, n'\}$ and the desired property holds for each of the lists $\mathbf{List}(\tau_1), \mathbf{List}(\tau_2), \ldots, \mathbf{List}(\tau_{i-1})$. The design of Step 1 means that the SRPT rule is implemented in the interval $[\tau_1, \tau_i]$ in Step 1. Thus, the remaining processing times at time τ_i in σ are exactly those in every schedule in $\Pi^*(\mathcal{J})$. We distinguish the following three possibilities for discussing the list $\mathbf{List}(\tau_i)$.

• $k_{i-1} = 0$. Then $k_i = 1$ and the available jobs at time τ_i are given by $\mathcal{F}(\tau_i)$. Thus, $\mathbf{List}(\tau_i) = (J_{y_i}, J_{y_i+1}, \ldots, J_{z_i})$. Similar to the discussion for $\mathbf{List}(\tau_1)$, the list $\mathbf{List}(\tau_i)$ has the desired property.

• $k_i = 0$. Then $\mathbf{List}(\tau_i)$ is empty, so it has the desired property trivially.

• $k_{i-1} > 0$ and $k_i > 0$. Suppose that $\mathbf{List}(\tau_{i-1}) = (J_{j_1}, J_{j_2}, \ldots, J_{j_a})$. Then the interval $[\tau_{i-1}, \tau_i]$ is occupied by job J_{j_a} in σ. If $p_{\min}(\tau_{i-1}) = \tau_i - \tau_{i-1}$, then $C_{j_a}(\sigma) = \tau_i$ and J_{j_a} is no longer available at time τ_i in σ; and if $p_{\min}(\tau_{i-1}) > \tau_i - \tau_{i-1}$, then $p_{j_a}^{(\sigma)}(\tau_i) > 0$ and J_{j_a} is still available at time τ_i in σ. Thus, we have

$$\mathbf{List}(\tau_i) = \begin{cases} (\mathbf{List}(\tau_{i-1}) \setminus \{J_{j_a}\}, \overrightarrow{\mathcal{F}}(\tau_i)), & \text{if } p_{\min}(\tau_{i-1}) = \tau_i - \tau_{i-1}, \\ (\mathbf{List}(\tau_{i-1}), \overrightarrow{\mathcal{F}}(\tau_i)), & \text{if } p_{\min}(\tau_{i-1}) > \tau_i - \tau_{i-1}. \end{cases}$$

This means that $\mathbf{List}(\tau_i)$ can be formed by shifting the jobs in the list $\overrightarrow{\mathcal{F}}(\tau_i)$ one by one to the list $\mathbf{List}(\tau_{i-1}) \setminus \{J_{j_a}\}$ or $\mathbf{List}(\tau_{i-1})$. Note that deleting the last job J_{j_a} from list $\mathbf{List}(\tau_{i-1})$ or shifting a job of $\overrightarrow{\mathcal{F}}(\tau_i)$ takes a constant time. Consequently, $\mathbf{List}(\tau_i)$ can be generated in $O(1 + |\mathcal{F}(\tau_i)|)$ time. Since $\mathcal{J}$ is a standard instance, when $\mathcal{F}(\tau_i) \neq \emptyset$, all the jobs in $\mathcal{F}(\tau_i)$ have the same processing time $p_{\min}(\tau_i)$. Therefore, the available jobs at time τ_i in σ are also listed in a non-increasing order of their remaining processing times in $\mathbf{List}(\tau_i)$. As a result, $\mathbf{List}(\tau_i)$ has the desired property.

The above discussion shows that all the lists $\mathbf{List}(\tau_i)$, $i \in \{1, 2, \ldots, n'\}$, have the desired property. This statement has two consequences:

(i) Note that $n' \leq 2n$ and $(\mathcal{F}(\tau_i) : 1 \leq i \leq n')$ forms a partition of $\mathcal{J}$. Then the time complexity for generating the lists $(\mathbf{List}(\tau_i) : 1 \leq i \leq n')$ is given by $O(n)$. This also shows that Step 1 runs in $O(n)$ time. Step 2 runs in $O(n)$ time clearly. Then Algorithm 3.2 runs in $O(n)$ time.

(ii) Step 1 is an implementation of the SRPT rule. Thus, $\sigma \in \Pi^*(\mathcal{J})$.

From (ii), we know that $J_{j^*} = J_{\sigma(n)} \in \mathcal{L}^{(n)}(\mathcal{J})$. We next show that j^* is the smallest index of the jobs in $\mathcal{L}^{(n)}(\mathcal{J})$. Then the result follows from Lemma 3.5.

If there exists some job index j' such that $j' < j^*$ and $C_{j'}(\sigma) > r_{j^*}$, then both J_{j^*} and $J_{j'}$ are available at time r_{j^*} in σ. The design of Step 1 implies that, at any decision time $\tau_i \geq r_{j^*}$, an available job with the largest job index is scheduled. Then $J_{j^*} = J_{j^*}^{(\sigma)}(r_{j^*})$ must be scheduled before the part $J_{j'}^{(\sigma)}(r_{j^*})$ of job $J_{j'}$ in σ since $j^* > j'$. This contradicts the fact that $C_{j'}(\sigma) < C_{j^*}(\sigma)$.

The above discussion shows that $C_h(\sigma) \leq r_{j^*}$ for every indices $h = 1, 2, \ldots, j^*-1$. This further means that $C^{(j^*-1)}(\mathcal{J}) \leq r_{j^*}$ and all the jobs $J_1, J_2, \ldots, J_{j^*-1}$ must be completed by time r_{j^*} in every schedule in $\Pi^*(\mathcal{J})$. Consequently, j^* is the smallest index of the jobs in $\mathcal{L}^{(n)}(\mathcal{J})$. The lemma follows. □

4 Problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$

We study problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$, where $f \in \{\sum f_j,\ f_{\max}\}$. In Section 4.1, we show that problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ is solvable in $O(n^3)$ time. In Section 4.2, we show that problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ is solvable in $O(n^2)$ time. In Section 4.3, we show that, for some special choices of f, problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$ is solvable in $O(n \log n)$ time.

From Lemma 2.8, given an instance $\mathcal{J}$, Procedure RD-Modification generates an equivalent and standard instance in $O(n^2)$ time. Then we only consider standard instances in Sections 4.1 and 4.2.

4.1 Problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$

Consider the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ on a standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. Then $\Pi^*(\mathcal{J})$ is the set of feasible schedules of this problem. From Lemmas 2.5 and 2.7, a schedule σ of $\mathcal{J}$ is in $\Pi^*(\mathcal{J})$ if and only if $\mathcal{O}^{(\sigma)}$ is not only a TC-optimal permutation but also a legal permutation of $\mathcal{J}$. This enables us to reduce problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ to an $n \times n$ linear assignment problem. Recall that, for each index $j \in \{1, 2, \ldots, n\}$ and each time $t \in [0, +\infty)$, $f_j(t)$ is a finite number. Then, for each feasible schedule $\sigma \in \Pi^*(\mathcal{J})$, the objective value $\sum_{j=1}^{n} f_j(\sigma)$ is a finite number.

We define indicator variables of a legal permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ of $\mathcal{J}$ as

$$x_{ij} = \begin{cases} 1, & \text{if } J_j \in \mathcal{L}^{(i)}(\mathcal{J}), \\ 0, & \text{otherwise.} \end{cases}$$

Define the cost of assigning job J_j to the i-th position in $\mathcal{O}$, i.e., $J_j = J_{o(i)}$, as

$$c_{ij} = \begin{cases} f_j(C^{(i)}(\mathcal{J})), & \text{if } J_j \in \mathcal{L}^{(i)}(\mathcal{J}), \\ +\infty, & \text{otherwise.} \end{cases} \tag{16}$$

Then we consider the following $n \times n$ linear assignment problem:

$$\begin{array}{lll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \text{s.t.} & \sum_{i=1}^{n} x_{ij} = 1, & \text{for all } j \in \{1, \ldots, n\}, \\ & \sum_{j=1}^{n} x_{ij} = 1, & \text{for all } i \in \{1, \ldots, n\}, \\ & x_{ij} \in \{0, 1\}, & \text{for all } i, j \in \{1, \ldots, n\}. \end{array} \tag{17}$$

For each feasible solution $x = (x_{ij} : 1 \leq i, j \leq n)$ of the problem in (17), we set

$$\text{Value}(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

which is the objective value of the solution x.

Lemma 4.1. *Let $x^* = (x^*_{ij} : 1 \leq i, j \leq n)$ be an optimal solution for the $n \times n$ linear assignment problem stated in (17). Let $\mathcal{O}^* = (J_{o^*(1)}, J_{o^*(2)}, \ldots, J_{o^*(n)})$ be the permutation of $\mathcal{J}$ such that $x^*_{i,o^*(i)} = 1$ for $i = 1, 2, \ldots, n$. Then $\mathcal{O}^*$ is an optimal schedule for the problem $1|r_j, pmtn|Lex(\sum C_j, \sum f_j)$ on instance $\mathcal{J}$.*

Proof. Let $\sigma \in \Pi^*(\mathcal{J})$. Then we have $C_{\sigma(i)}(\sigma) = C^{(i)}(\mathcal{J})$ and $J_{\sigma(i)} \in \mathcal{L}^{(i)}(\mathcal{J})$ for $i = 1, 2, \ldots, n$. Thus, $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is a legal permutation of $\mathcal{J}$. Now we define a solution $x = (x_{ij} : 1 \leq i, j \leq n)$ for (17) as

$$x_{ij} = \begin{cases} 1, & \text{if } j = \sigma(i), \\ 0, & \text{otherwise.} \end{cases}$$

Since $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is a permutation of $\mathcal{J}$, it is observed that $x = (x_{ij} : 1 \leq i, j \leq n)$ is a feasible solution for the problem in (17). From the legality of $\mathcal{O}^{(\sigma)}$, we have

$$\begin{array}{rcl} \text{Value}(x) & = & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ & = & \sum_{i=1}^{n} c_{i,\sigma(i)} \\ & = & \sum_{i=1}^{n} f_{\sigma(i)}(C^{(i)}(\mathcal{J})) \\ & = & \sum_{i=1}^{n} f_{\sigma(i)}(C_{\sigma(i)}(\sigma)) \\ & = & \sum_{j=1}^{n} f_j(\sigma) \\ & < & +\infty. \end{array}$$

From the optimality of x^*, we thus have

$$\text{Value}(x^*) \leq \text{Value}(x) = \sum_{j=1}^{n} f_j(\sigma) < +\infty. \tag{18}$$

If $x^*_{i'j'} = 1$ for some $i' \in \{1, 2, \ldots, n\}$ and $J_{j'} \notin \mathcal{L}^{(i')}(\mathcal{J})$, then $c_{i'j'} = +\infty$. As a result, we have $\text{Value}(x^*) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x^*_{ij} = +\infty$. This contradicts (18). Hence, we have

$$x^*_{ij} = 0 \text{ if } J_j \notin \mathcal{L}^{(i)}(\mathcal{J}). \tag{19}$$

From (19) and from the definition of permutation $\mathcal{O}^* = (J_{o^*(1)}, J_{o^*(2)}, \ldots, J_{o^*(n)})$, which requires that $x^*_{i,o^*(i)} = 1$ for $i = 1, 2, \ldots, n$, we have $J_{o^*(i)} \in \mathcal{L}^{(i)}(\mathcal{J})$ for $i = 1, 2, \ldots, n$. Thus, $\mathcal{O}^*$ is a legal permutation of $\mathcal{J}$. From Lemma 2.7, $\mathcal{O}^*$ is a TC-optimal permutation of $\mathcal{J}$, so $\mathcal{O}^* \in \Pi^*(\mathcal{J})$. This further implies that $C_{o^*(i)}(\mathcal{O}^*) = C^{(i)}(\mathcal{J})$ for $i = 1, 2, \ldots, n$. From (18) and (19), we have

$$\begin{aligned}
\textstyle\sum_{j=1}^{n} f_j(\mathcal{O}^*) &= \textstyle\sum_{i=1}^{n} f_{o^*(i)}(\mathcal{O}^*) \\
&= \textstyle\sum_{i=1}^{n} f_{o^*(i)}(C_{o^*(i)}(\mathcal{O}^*)) \\
&= \textstyle\sum_{i=1}^{n} f_{o^*(i)}(C^{(i)}(\mathcal{J})) \\
&= \textstyle\sum_{i=1}^{n} c_{i,o^*(i)} \\
&= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x^*_{ij} \\
&= \text{Value}(x^*) \\
&\leq \text{Value}(x) \\
&= \textstyle\sum_{j=1}^{n} f_j(\sigma).
\end{aligned}$$

Consequently, $\mathcal{O}^*$ is optimal for the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ on instance $\mathcal{J}$. The lemma follows. □

Based on Lemma 4.1, we present the following algorithm.

Algorithm 4.1. *For solving problem* $1|r_j, pmtn|Lex(\sum C_j, \sum f_j)$.

Input: A standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$.
Step 1: Run Algorithm 3.1 to obtain the completion times $C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$ and the legal sets $\mathcal{L}^{(1)}(\mathcal{J}), \mathcal{L}^{(2)}(\mathcal{J}), \ldots, \mathcal{L}^{(n)}(\mathcal{J})$.
Step 2: Calculate the position cost c_{ij}, $1 \leq i, j \leq n$, by using (16), and then generate the $n \times n$ linear assignment problem in (17).
Step 3: Solve the $n \times n$ linear assignment problem in (17) to obtain its optimal solution $x^* = (x^*_{ij} : 1 \leq i, j \leq n)$ and its optimal value $\text{Value}(x^*) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x^*_{ij}$.
Step 4: Generate the permutation $\mathcal{O}^* = (J_{o^*(1)}, J_{o^*(2)}, \ldots, J_{o^*(n)})$ of $\mathcal{J}$, where, for each $i \in \{1, 2, \ldots, n\}$, $o^*(i)$ is the unique index in $\{1, 2, \ldots, n\}$ such that $x^*_{i,o^*(i)} = 1$.
Step 5: Run Procedure Pmtn-LS($\mathcal{O}^*$) to obtain the schedule $\sigma^* = \text{Pmtn-LS}(\mathcal{O}^*)$ of $\mathcal{J}$.
Output: Schedule $\sigma^* = \text{Pmtn-LS}(\mathcal{O}^*)$ and its objective value $\text{Value}(x^*)$.

Theorem 4.1. *Algorithm 4.1 solves problem* $1|r_j, pmtn|Lex(\sum C_j, \sum f_j)$ *in* $O(n^3)$ *time.*

Proof. The correctness of Algorithm 4.1 follows from Lemmas 3.4 and 4.1. We next estimate the time complexity of the algorithm.

From Lemma 3.4, Step 1 runs in $O(n^2)$ time. Step 2 runs in $O(n^2)$ time clearly. From Lawler (1976), the $n \times n$ linear assignment problem in (17) is solvable in $O(n^3)$ time. Thus, Step 3 can be implemented in $O(n^3)$ time. Step 4 runs in $O(n^2)$ time clearly. From Yuan et al. (2015), Procedure Pmtn-LS($\mathcal{O}^*$) in Step 5 can be implemented in $O(n \log n)$ time. Then the overall time complexity of Algorithm 4.1 is $O(n^3)$. □

4.2 Problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$

Consider the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ on a standard instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$. We use the following lemma to determine the last completed job in an optimal schedule.

Lemma 4.2. *Consider the problem* $1|r_j, pmtn|Lex(\sum C_j, f_{\max})$ *on instance* $\mathcal{J}$ *and suppose that* $J_{j'} \in \mathcal{L}^{(n)}(\mathcal{J})$ *such that*

$$f_{j'}(C^{(n)}(\mathcal{J})) = \min\{f_{j''}(C^{(n)}(\mathcal{J})) : J_{j''} \in \mathcal{L}^{(n)}(\mathcal{J})\}.$$

Then there is an optimal schedule for the problem in which $J_{j'}$ *is the last completed job.*

Proof. Let σ be an optimal schedule such that $J_{j'}$ is completed as late as possible. Since $\sigma \in \Pi^*(\mathcal{J})$, we have $C_{\sigma(i)}(\sigma) = C^{(i)}(\mathcal{J})$ for $i = 1, 2, \ldots, n$. Then we only need to show that $j' = \sigma(n)$.

Suppose to the contrary that $j' = \sigma(i')$ for some index $i' \in \{1, 2, \ldots, n-1\}$. Then

$$r_{j'} < C_{j'}(\sigma) = C_{\sigma(i')}(\sigma) < C^{(n)}(\mathcal{J}).$$

Suppose that the intervals occupied by job $J_{j'}$ in σ are given by

$$[\tau_{i_1}, \tau_{i_1+1}] \leq [\tau_{i_2}, \tau_{i_2+1}] \leq \cdots \leq [\tau_{i_a}, \tau_{i_a+1}],$$

where, for two intervals $[\tau, \tau']$ and $[t, t']$, the expression $[\tau, \tau'] \leq [t, t']$ indicates that $\tau < \tau' \leq t < t'$. Then $r_{j'} \leq \tau_{i_1}$ and $\tau_{i_a+1} = C_{j'}(\sigma) = C_{\sigma(i')}(\sigma)$. We have the following claim.

Claim 1. For each $\tau \in \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_a}\}$, we have

(i) $p_{j'}^{(\sigma)}(\tau)$ is the smallest remaining processing time at time τ in σ,

(ii) $J_{j'}$ is the unique job with the smallest remaining processing time at time τ in σ, and

(iii) in every schedule in $\Pi^*(\mathcal{J})$, $p_{j'}^{(\sigma)}(\tau)$ is the smallest remaining processing time at time τ and exactly one available job at time τ has the remaining processing time $p_{j'}^{(\sigma)}(\tau)$.

In fact, Statement (i) follows from the principle of the SRPT rule, since $\sigma \in \Pi^*(\mathcal{J})$.

To prove Statement (ii), we suppose to the contrary that there is a decision time $\tau \in \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_a}\}$ such that a (τ, σ)-available job J_j other than $J_{j'}$ also has the smallest remaining processing time at time τ in σ. Then $p_j^{(\sigma)}(\tau) = p_{j'}^{(\sigma)}(\tau)$. By the SRPT rule, the part $J_j^{(\sigma)}(\tau)$ is completely processed after the part $J_{j'}^{(\sigma)}(\tau)$ in σ. Let π be the schedule of $\mathcal{J}$ that is obtained from σ by swapping the two parts $J_j^{(\sigma)}(\tau)$ and $J_{j'}^{(\sigma)}(\tau)$. Since $\sigma \in \Pi^*(\mathcal{J})$ and $p_j^{(\sigma)}(\tau) = p_{j'}^{(\sigma)}(\tau)$, we have $\pi \in \Pi^*(\mathcal{J})$. Note that $C_{j'}(\pi) > C_{j'}(\sigma)$ and $J_{j'}$ is the only job whose completion time in π is greater than that in σ. Since $C_{j'}(\pi) \le C^{(n)}(\mathcal{J})$, we have $f_{j'}(\pi) \le f_{j'}(C^{(n)}(\mathcal{J})) = \min\{f_{j''}(C^{(n)}(\mathcal{J})) : J_{j''} \in \mathcal{L}^{(n)}(\mathcal{J})\} \le f_{\sigma(n)}(\sigma) \le f_{\max}(\sigma)$. This means that $f_{\max}(\pi) \le \max\{f_{\max}(\sigma), f_{j'}(\pi)\} = f_{\max}(\sigma)$. Consequently, π is also an optimal schedule for problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$. But then, the relation $C_{j'}(\pi) > C_{j'}(\sigma)$ contradicts the choice of σ. This proves Statement (ii).

Statement (iii) is just a consequence of Lemma 2.2 and the above two statements. This completes the proof of Claim 1.

Since $J_{j'} \in \mathcal{L}^{(n)}(\mathcal{J})$, from the principle of the SRPT rule, there is a schedule $\sigma' \in \Pi^*(\mathcal{J})$ such that $C_{j'}(\sigma') = C^{(n)}(\mathcal{J})$, so, at every decision time in $\mathcal{T}(\mathcal{J}) \cap [r_{j'}, C^{(n)}(\mathcal{J}))$, job $J_{j'}$ is available and has the largest remaining processing time in σ'. From Lemma 2.2, the remaining processing times at each decision time are independent of the choices of the schedules in $\Pi^*(\mathcal{J})$. Then we have the following claim.

Claim 2. At each time $\tau \in \{\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_a}\}$, $p_{j'}^{(\sigma')}(\tau)$ is the largest remaining processing time in σ'.

The following claim is critical for our discussion.

Claim 3. For each $a' \in \{1, 2, \ldots, a\}$, we have $p_{j'}^{(\sigma)}(\tau_{i_{a'}}) = p_{j'}^{(\sigma')}(\tau_{i_{a'}})$ and the interval $[\tau_{i_{a'}}, \tau_{i_{a'}+1}]$ is fully occupied by job $J_{j'}$ in σ'.

To prove Claim 3, we first consider the states of σ and σ' at the decision time τ_{i_1} and the interval $[\tau_{i_1}, \tau_{i_1+1}]$. Since τ_{i_1} is the starting time of job $J_{j'}$ in σ, we have $p_{j'}^{(\sigma)}(\tau_{i_1}) = p_{j'}$. From Claims 1 and 2, we have $p_{j'} = p_{j'}^{(\sigma)}(\tau_{i_1}) \le p_{j'}^{(\sigma')}(\tau_{i_1}) \le p_{j'}$. Consequently, we have $p_{j'}^{(\sigma)}(\tau_{i_1}) = p_{j'}^{(\sigma')}(\tau_{i_1})$. From Claim 1(iii) and Claim 2, $J_{j'}$ is the unique available job at time τ_{i_1} in σ'. Since τ_{i_1} and τ_{i_1+1} are two consecutive decision points, no jobs in $\mathcal{J}$ are released in the interval $(\tau_{i_1}, \tau_{i_1+1})$. Consequently, the time interval $[\tau_{i_1}, \tau_{i_1+1}]$ is fully occupied by job $J_{j'}$ in σ'. As a result, Claim 3 holds for $a' = 1$.

Inductively, suppose that $2 \le a' \le a$ and the intervals

$$[\tau_{i_1}, \tau_{i_1+1}], [\tau_{i_2}, \tau_{i_2+1}], \ldots, [\tau_{i_{a'-1}}, \tau_{i_{a'-1}+1}]$$

are fully occupied by job $J_{j'}$ in σ'. From Claims 1 and 2, and by using the induction hypothesis, we have

$$\begin{aligned} p_{j'}^{(\sigma)}(\tau_{i_{a'}}) &= p_{j'} - (|[\tau_{i_1}, \tau_{i_1+1}]| + |[\tau_{i_2}, \tau_{i_2+1}]| + \cdots + |[\tau_{i_{a'-1}}, \tau_{i_{a'-1}+1}]|) \\ &\geq p_{j'}^{(\sigma')}(\tau_{i_{a'}}) \geq p_{j'}^{(\sigma)}(\tau_{i_{a'}}). \end{aligned}$$

It follows that $p_{j'}^{(\sigma)}(\tau_{i_{a'}}) = p_{j'}^{(\sigma')}(\tau_{i_{a'}})$. From Claim 1(iii) and Claim 2 again, $J_{j'}$ is the unique available job at time $\tau_{i_{a'}}$ in σ', so the interval $[\tau_{i_{a'}}, \tau_{i_{a'}+1}]$ is fully occupied by job $J_{j'}$ in σ'. This proves Claim 3.

From Claim 3, we have $C_{j'}(\sigma') = C_{j'}(\sigma) < C^{(n)}(\mathcal{J})$. This contradicts the fact that $C_{j'}(\sigma') = C^{(n)}(\mathcal{J})$. The lemma follows. □

We provide the following algorithm to solve problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$.

Algorithm 4.2. *For solving problem* $1|r_j, pmtn|Lex(\sum C_j, f_{\max})$.

Input: A standard job instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$.
Step 1: Generate a permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ of $\mathcal{J}$ in the following way.
(1.1) Set $\mathcal{J}_n = \mathcal{J}$. Set $i := n$.
(1.2) Run Algorithm 3.2 on instance $\mathcal{J}_i$ to obtain $\mathcal{L}^{(i)}(\mathcal{J}_i)$.
(1.3) Pick a job $J_{j'} \in \mathcal{L}^{(i)}(\mathcal{J}_i)$ such that $f_{j'}(C^{(i)}(\mathcal{J}_i))$ is as small as possible. Set $o(i) := j'$.
(1.4) If $i = 1$, then go to Step 2. If $i > 1$, then set $\mathcal{J}_{i-1} := \mathcal{J}_i \setminus \{J_{j'}\}$ and go to Step (1.5).
(1.5) Set $i := i - 1$ and go to Step (1.2).
Step 2: Generate the schedule $\sigma = \text{Pmtn-LS}(\mathcal{O})$ and calculate the value $f_{\max}(\sigma)$.
Output: Schedule σ and its objective value $f_{\max}(\sigma)$.

We have the following theorem.

Theorem 4.2. *Problem* $1|r_j, pmtn|Lex(\sum C_j, f_{\max})$ *is solvable by Algorithm 4.2 in* $O(n^2)$ *time.*

Proof. From Lemmas 2.9 and 3.6, and from the design of the algorithm, all the instances $\mathcal{J}_n, \mathcal{J}_{n-1}, \ldots, \mathcal{J}_1$ generated in the algorithm are standard, and their last legal sets $\mathcal{L}^{(i)}(\mathcal{J}_i)$, $i = n, n-1, \ldots, 1$, are generated correctly.

From Step (1.3), for $i \in \{1, 2, \ldots, n\}$, we have

$$J_{o(i)} \in \mathcal{L}^{(i)}(\mathcal{J}_i) \tag{20}$$

and

$$f_{o(i)}(C^{(i)}(\mathcal{J}_i)) = \min\{f_j(C^{(i)}(\mathcal{J}_i)) : J_j \in \mathcal{L}^{(i)}(\mathcal{J}_i)\}. \tag{21}$$

From (20), for each $i \in \{1, 2, \ldots, n\}$, there is a schedule $\sigma_i \in \Pi^*(\mathcal{J}_i)$ such that $\sigma_i(i) = o(i)$, so by repeatedly using Lemma 2.6, we have $C^{(i)}(\mathcal{J}_i) = C^{(i)}(\mathcal{J}_{i+1}) = \cdots = C^{(i)}(\mathcal{J}_n) = C^{(i)}(\mathcal{J})$ and $\mathcal{L}^{(i)}(\mathcal{J}_i) \subseteq \mathcal{L}^{(i)}(\mathcal{J}_{i+1}) \subseteq \cdots \subseteq \mathcal{L}^{(i)}(\mathcal{J}_n) = \mathcal{L}^{(i)}(\mathcal{J})$. Thus, from (20) and (21), we have

$$C^{(i)}(\mathcal{J}_i) = C^{(i)}(\mathcal{J}) \text{ and } J_{o(i)} \in \mathcal{L}^{(i)}(\mathcal{J}), \ \forall i. \tag{22}$$

The relation in (22) just implies that $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ is a legal permutation of $\mathcal{J}$. From Lemma 3.4, $\mathcal{O}$ is a TC-optimal permutation of $\mathcal{J}$. Consequently, we have $\sigma \in \Pi^*(\mathcal{J})$, i.e., σ is a feasible schedule for the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ on instance $\mathcal{J}$.

From Lemma 4.2, the relation in (21) implies that, for each $i \in \{1, 2, \ldots, n\}$, there is an optimal schedule σ_i^* for the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ on instance $\mathcal{J}_i$ such that $\sigma_i^*(i) = o(i)$. Since $\mathcal{J}_n = \mathcal{J}$, σ_n^* is optimal for the problem on instance $\mathcal{J}$. From (21) and (22), we have

$$f_{o(i)}(\sigma) = f_{o(i)}(C^{(i)}(\mathcal{J}_i)) = f_{o(i)}(\sigma_i^*) \leq f_{\max}(\sigma_i^*). \tag{23}$$

Since $\mathcal{J}_1 \subset \mathcal{J}_2 \subset \cdots \subset \mathcal{J}_n = \mathcal{J}$, we further have

$$f_{\max}(\sigma_1^*) \leq f_{\max}(\sigma_2^*) \leq \cdots \leq f_{\max}(\sigma_n^*). \tag{24}$$

Combining (23) and (24), we have $f_{\max}(\sigma) = \max\{f_{o(i)}(\sigma) : i = 1, 2, \ldots, n\} \leq f_{\max}(\sigma_n^*)$. This means that σ is an optimal schedule for the problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ on instance $\mathcal{J}$. Then the correctness of Algorithm 4.2 follows.

The running time of the algorithm is dominated by that of Step 1, which has n iterations. In each iteration, Step (1.2) runs in $O(n)$ time (which follows from Lemma 3.6) and Step (1.3) takes $O(n)$ time clearly, so $O(n)$ time is needed. Hence, the overall time complexity of Algorithm 4.2 is $O(n^2)$. The theorem follows. □

4.3 Problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$

Consider problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$ with $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$. The problem can be solved in polynomial time by the algorithms in Section 4.1 or Section 4.2. However, we show that it can be more efficiently solved in $O(n \log n)$ time. Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, where $r_1 \leq r_2 \leq \ldots \leq r_n$.

First, we define a permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ of $\mathcal{J}$ associated with each problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$ in Table 1, where $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$.

Note that each permutation $\mathcal{O}$ in Table 1 can be obtained in $O(n \log n)$ time. By the pairwise job exchange argument, we can easily show the following lemma.

Table 1: The $\mathcal{O}$-permutation for problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$

Scheduling problem	$\mathcal{O}$-permutation
$1\|r_j, \text{pmtn}\|\text{Lex}(\sum C_j, \sum w_j C_j)$	$w_{o(1)} \geq w_{o(2)} \geq \cdots \geq w_{o(n)}$
$1\|r_j, \text{pmtn}\|\text{Lex}(\sum C_j, \sum T_j)$	$d_{o(1)} \leq d_{o(2)} \leq \cdots \leq d_{o(n)}$
$1\|r_j, \text{pmtn}\|\text{Lex}(\sum C_j, L_{\max})$	$d_{o(1)} \leq d_{o(2)} \leq \cdots \leq d_{o(n)}$
$1\|r_j, \text{pmtn}\|\text{Lex}(\sum C_j, T_{\max})$	$d_{o(1)} \leq d_{o(2)} \leq \cdots \leq d_{o(n)}$
$1\|r_j, \text{pmtn}\|\text{Lex}(\sum C_j, WC_{\max})$	$w_{o(1)} \geq w_{o(2)} \geq \cdots \geq w_{o(n)}$

Lemma 4.3. *For problem $1|r_j, pmtn|Lex(\sum C_j, \tilde{f})$, there is an optimal schedule $\sigma \in \Pi^*(\mathcal{J})$ such that, at each decision time $\tau_i \in \mathcal{T}(\mathcal{J})$ with $k_i \geq 1$, if multiple available jobs have the smallest remaining processing time, the one with the smallest index under permutation $\mathcal{O}$ is processed in the interval $[\tau_i, \tau_{i+1}]$ in σ.*

Based on Lemma 4.3, problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$ can be solved by the following procedure, called the SRPT($\tilde{f}$) Rule.

SRPT($\tilde{f}$) Rule: *At each decision time, we schedule an available job with the smallest remaining processing time, with ties being broken by choosing the candidate job with the smallest index under permutation $\mathcal{O}$. This procedure is repeated until all the jobs are scheduled.*

It is noted that the SRPT($\tilde{f}$) Rule runs in $O(n \log n)$ time. Thus, we have the following theorem.

Theorem 4.3. *Problem $1|r_j, pmtn|Lex(\sum C_j, \tilde{f})$ is solvable by the SRPT($\tilde{f}$) Rule in $O(n \log n)$ time, where $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$.*

Remark: For problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum U_j)$, we fail to find a scheduling strategy similar to SRPT($\tilde{f}$) because breaking the ties by the EDD rule does not guarantee resulting in an optimal schedule. But we do feel that the problem can also be solved in $O(n \log n)$ time.

5 A numerical example and computational experiments

A Numerical Example: We provide a numerical example to illustrate the concepts of completion-coinciding permutation, standard instance, and legal set. Consider the job instance given in Table 2.

Table 2: The job instance $\mathcal{J}$

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
release dates	1	2	2	5	5	10	10
Processing times	3	1	1	2	1	2	1

Figure 1 illustrates the implementation of the SRPT rule on instance $\mathcal{J}$, where an available job with the smallest remaining processing time is scheduled at each decision time. Note that there may be more than one available job with the smallest remaining processing time at some decision times.

From Figure 1, we find a schedule σ such that

$$C_2(\sigma) < C_3(\sigma) < C_5(\sigma) < C_1(\sigma) < C_4(\sigma) < C_7(\sigma) < C_6(\sigma).$$

Then $(J_2, J_3, J_5, J_1, J_4, J_7, J_6)$ is a completion-coinciding permutation.

From Figure 1, the seven completion times are $C^{(1)}(\mathcal{J}) = 3$, $C^{(2)}(\mathcal{J}) = 4$, $C^{(3)}(\mathcal{J}) = 6$, $C^{(4)}(\mathcal{J}) = 7$, $C^{(5)}(\mathcal{J}) = 9$, $C^{(6)}(\mathcal{J}) = 11$, and $C^{(7)}(\mathcal{J}) = 13$. Moreover, we note that both J_2 and J_3 can each be completed at $C^{(1)}(\mathcal{J})$ and $C^{(2)}(\mathcal{J})$, while no other jobs can be completed at $C^{(1)}(\mathcal{J})$ or $C^{(2)}(\mathcal{J})$. Thus, the legal sets $\mathcal{L}^{(1)}(\mathcal{J}) = \mathcal{L}^{(2)}(\mathcal{J}) = \{J_2, J_3\}$. In fact, using Algorithm 3.1, we can further obtain that $\mathcal{L}^{(3)}(\mathcal{J}) = \mathcal{L}^{(4)}(\mathcal{J}) = \{J_1, J_5\}$, $\mathcal{L}^{(5)}(\mathcal{J}) = \{J_4\}$, $\mathcal{L}^{(6)}(\mathcal{J}) = \{J_7\}$, and $\mathcal{L}^{(7)}(\mathcal{J}) = \{J_6\}$.

Now we apply Procedure RD-Modification to obtain an equivalent and standard instance $\mathcal{J}'$.

From Figure 1, the set of decision times is $\mathcal{T}(\mathcal{J}) = \{1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13\}$, i.e., $(\tau_1, \tau_2, \ldots, \tau_{11}) = (1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13\}$.

- First, we set $r'_j := r_j$, $j = 1, 2, \ldots, 7$.
- At time τ_1, we have $r'_1 = \tau_1$. Since $p_1 = p_{\min}(\tau_1) = 3$, J'_1 is a new job with release date τ_1 and processing time p_1.
- At time τ_2, we have $r'_2 = r'_3 = \tau_2$. Since $p_2 = p_3 = p_{\min}(\tau_2) = 1$, J'_2 is a new job with release date τ_2 and processing time p_2, and J'_3 is a new job with release date τ_2 and processing time p_3.

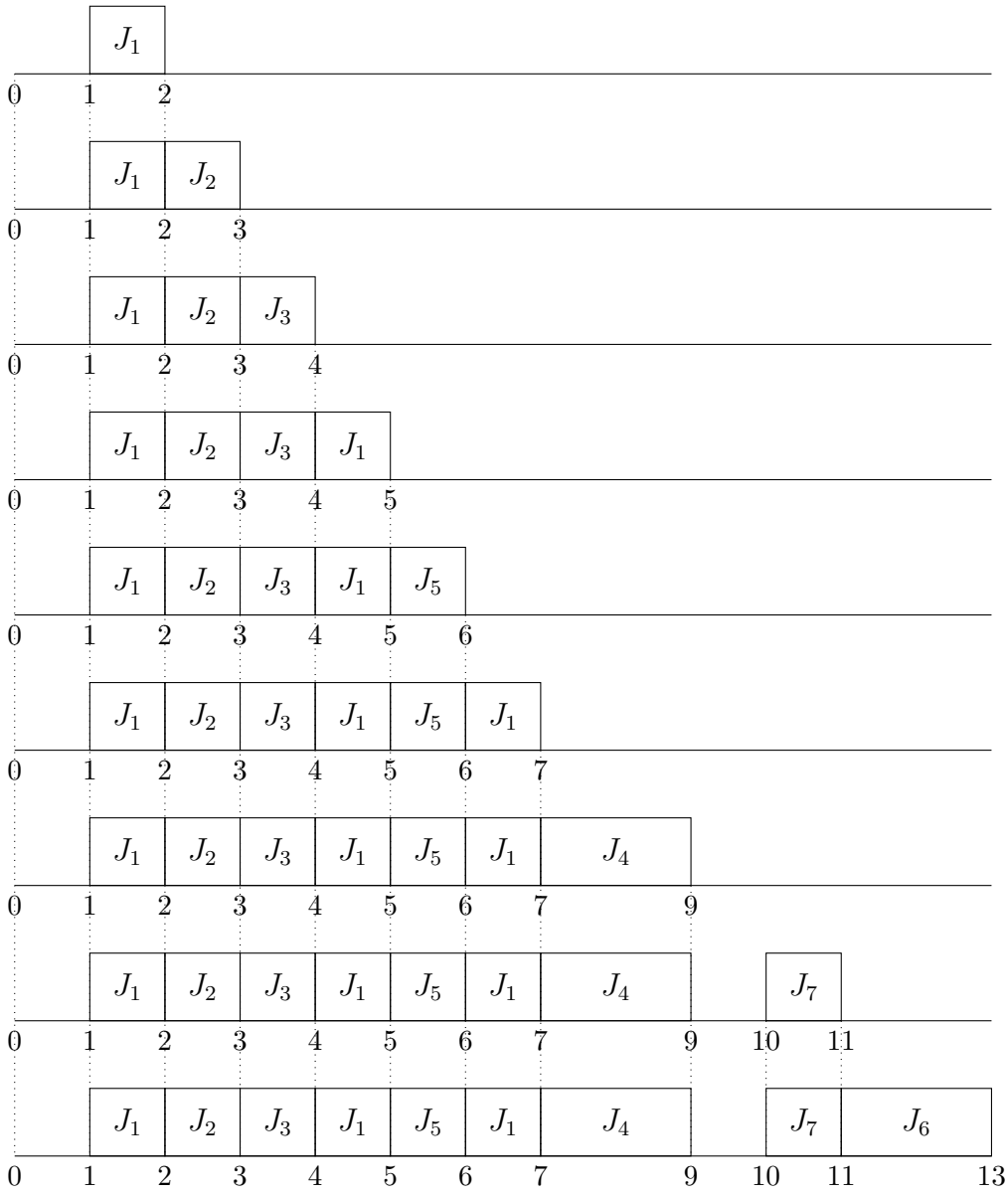

Figure 1: Implementation of the SRPT rule

• At times τ_3 and τ_4, no job is released.

• At time τ_5, we have $r'_4 = r'_5 = \tau_5$. Since $p_4 > p_{\min}(\tau_5) = 1$, we reset $r'_4 := \tau_6$. Since $p_4 > p_{\min}(\tau_6) = 1$, we reset $r'_4 := \tau_7$. Since $p_4 = p_{\min}(\tau_7) = 2$, J'_4 is a new job with release date τ_7 and processing time p_4. Furthermore, since $p_5 = p_{\min}(\tau_5) = 1$, J'_5 is a new job with release date τ_5 and processing time p_5.

• At times τ_6, τ_7 and τ_8, no job is released.

• At time τ_9, we have $r'_6 = r'_7 = \tau_9$. Since $p_6 > p_{\min}(\tau_9) = 1$, we reset $r'_6 := \tau_{10}$. Since $p_6 = p_{\min}(\tau_{10}) = 2$, J'_6 is a new job with release date τ_{10} and processing time p_6. Furthermore, since $p_7 = p_{\min}(\tau_9) = 1$, J'_7 is a new job with release date τ_9 and processing time p_7.

Thus, an equivalent and standard instance $\mathcal{J}' = \{J'_1, J'_2, \ldots, J'_7\}$ is obtained, whose details are illustrated in Table 3.

Table 3: The standard instance $\mathcal{J}'$

Jobs	J'_1	J'_2	J'_3	J'_4	J'_5	J'_6	J'_7
release dates	1	2	2	7	5	11	10
Processing times	3	1	1	2	1	2	1

Computational Experiments: We next present computational experiments of applying our algorithms to solve different instances of the two problems in (1) and (2). To evaluate the performance of the proposed algorithms, we coded them in MATLAB and performed the experiments on a personal computer powered by an Intel(R) Core(TM)i7-7500U CPU at 2.70GHz with 8GB RAM operating under Windows 10. We generated the instances with $n = 10, 50, 100, 300, 500$ jobs. For each job J_j $(j = 1, 2, \ldots, n)$, we sampled its release date r_j and processing time p_j from the uniform integer distributions $U[1, 50]$ and $U[1, 30]$, respectively. Moreover, we set the cost function of job J_j as a linear function $f_j(t) = a_j t + b_j$ with the values of the parameters a_j and b_j being integers randomly selected from $U[1, \alpha]$, where $\alpha \in \{10, 100\}$. We present the results of the experiments in Table 4, which show that our Algorithms 4.1 and 4.2 are very fast in solving the problems in (1) and (2). Given that all our algorithms run in polynomial time, i.e., they are efficient in theory, the computational results are expected.

It should be noticed that although the values of the release dates of the jobs in the scheduling instances are selected randomly from $U[1, 50]$, the release dates in the induced standard instances may be much greater than 50 because our algorithms are based on the standard instances.

Table 4: Average running time (in seconds)

Algorithm	α	n				
		10	50	100	300	500
Algorithm 4.1	10	0.017	0.0410	0.113	1.309	5.843
	100	0.021	0.0423	0.125	1.317	5.863
Algorithm 4.2	10	0.003	0.018	0.051	0.360	1.024
	100	0.005	0.019	0.049	0.362	1.036

Acknowledgments

We sincerely thank an Associate Editor and three anonymous referees for their helpful comments and suggestions on an early version of our paper. This research was supported in part by the NSFC under grant numbers 11671368, 11771406, and 11971443.

References

Aalto S., & Ayesta U. (2009). SRPT applied to bandwidth-sharing networks. *Annals of Operations Research,* 170(1), 3-19.

Agnetis A., Billaut J. C., Gawiejnowicz S., Pacciarelli D., & Soukhal A. (2014). *Multi-agent Scheduling: Models and Algorithms.* Springer, Berlin.

Agnetis A., Mirchandani P. B., Pacciarelli D., & Pacifici A. (2004). Scheduling problems with two competing agents. *Operations Research,* 52(2), 229-242.

Ahmad F., Mahmud S. A., Khan G. M., & Yousaf F. Z. (2013). Shortest remaining processing time based schedulers for reduction of traffic congestion. *International Conference on Connected Vehicles and Expo.*

Akande S., Oluleye A. E., & Oyetunji E. (2017). Minimization of total tardiness and total flowtime on single machine with non-zero release dates. *International Journal of Engineering Research in Africa,* 29, 154-164.

Baker K. R. (1974). *Introduction to sequencing and scheduling.* John Wiley & Sons, New York.

Brucker P. (2006). *Scheduling Algorithms.* Fifth Edition, Springer, Berlin.

Chen C. L., & Bulfin R. L. (1993). Complexity of single machine, multi-criteria scheduling problems. *European Journal of Operational Research,* 70(1), 115-125.

Chen R. B., & Yuan J. J. (2018). 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(1), 45-68.

Emmons H. (1975). One machine sequencing to minimize mean flow time with minimum number tardy. *Naval Research Logistics Quarterly,* 22(3), 585-592.

Gao Y., & Yuan J. J. (2015). A note on Pareto minimizing total completion time and maximum cost. *Operations Research Letters,* 43(1), 80-82.

Gao Y., & Yuan J. J. (2017). Bi-criteria Pareto-scheduling on a single machine with due indices and precedence constraints. *Discrete Optimization,* 25, 105-119.

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.

Hoogeveen H. (2005). Multicriteria scheduling. *European Journal of Operational Research,* 167(3), 592-623.

Hoogeveen J. A. (1996). Single machine scheduling to minimize a function of two or three maximum cost criteria. *Journal of Algorithms,* 21(2), 415-433.

Hoogeveen J. A., & van de Velde S. L. (1995). Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time. *Operations Research Letters,* 17(5), 205-208.

Huang R. H., & Yang C. L. (2009). An algorithm for minimizing flow time and maximum earliness on a single machine. *Journal of the Operational Research Society,* 60(6), 873-877.

Huo Y. M., Leung J. Y. T., & Zhao H. R. (2007). Complexity of two dual criteria scheduling problems. *Operations Research Letters,* 35(2), 211-220.

Kiran A. S., & Unal A. T. (1991). A single-machine problem with multiple criteria. *Naval Research Logistics,* 38(5), 721-727.

Kondakci S. K., & Bekiroglu T. (1997). Scheduling with bicriteria: total flowtime and number of tardy jobs. *International Journal of Production Economics,* 53(1), 91-99.

Kyparisis J., & Douligeris C. (1993). Single machine scheduling and selection to minimize total flow time with minimum number tardy. *Journal of the Operational Research Society,* 44(8), 835-838.

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* (pp. 245-261). Academic Press.

Lawler E. L. (1973). Optimal sequencing of a single machine subject to precedence constraints. *Management Science,* 19(5), 544-546.

Lawler E. L. (1976). *Combinatorial Optimization: networks and matroids.* Courier Corporation.

Leung J. Y. T. (2004). *Handbook of Scheduling: algorithms, models, and performance analysis.* Chapman and Hall/CRC.

Lin K. S. (1983). Hybrid algorithm for sequencing with bicriteria. *Journal of Optimization Theory and Applications,* 39(1), 105-124.

Nelson R. T., Sarin R. K., & Daniels R. L. (1986). Scheduling with multiple performance measures: the one-machine case. *Management Science,* 32(4), 464-479.

Prakash R., & Veeravalli V. V. (2007). Centralized wireless data networks with user arrivals and departures. *IEEE Transactions on Information Theory,* 53(2), 695-713.

Pinedo M. (2002). *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, New Jersey.

Schrage L .(1968). A proof of the optimality of the shortest remaining processing time discipline. *Operations Research,* 16(3), 687-690.

Smith D. R. (1978). A new proof of the optimality of the shortest remaining processing time discipline. *Operations Research,* 26(1), 197-199.

Sourd F. (2001). Preemptive scheduling with two minimax criteria. *Annals of Operations Research,* 107(1-4), 303-319.

T'kindt V., & Billaut J. C. (2006). *Multicriteria Scheduling: Theory, Models and Algorithms.* Springer Science Business Media.

Van Wassenhove L. N., & Gelders L. F. (1980). Solving a bicriterion scheduling problem. *European Journal of Operational Research,* 4(1), 42-48.

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(3), 315-323.

Yen B. P. C., & Wan G. H. (2003). Single machine bicriteria scheduling: A survey. *International Journal of Industrial Engineering,* 10(3), 222-231.

Yuan J. J., Ng C. T., & Cheng T. C. E. (2015). Two-agent single-machine scheduling with release dates and preemption to minimize the maximum lateness. *Journal of Scheduling,* 18(2), 147-153.

Yuan J. J., Ng C. T., & Cheng T. C. E. (2020). Scheduling with release dates and preemption to minimize multiple max-form objective functions. *European Journal of Operational Research,* 280(3), 860-875.