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

Rubing Chen
(joint work with Prof. Jinjiang Yuan, Prof. C.T. Ng, Prof.T.C.E. Cheng)

Zhengzhou University, China

Global Scheduling Seminar: www.schedulingseminar.com

1. Introduction
2. Revisit problem $1|r_j, \text{pmtn}|\sum C_j$
3. Legal sets
4. $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$
   - $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$
   - $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$
   - $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$
5. Further Research

Single-machine hierarchical scheduling problem $1|\beta|\text{Lex}(f, g)$:
The problem aims to find a feasible schedule that minimizes the secondary criterion g under the condition that the primary criterion f is minimized.

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

- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ $\quad O(n^3)$
- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$ $\quad O(n^2)$
  where each $f_j(\cdot)$ is a regular function, $j = 1, 2, \ldots, n$.

$\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$: the set of jobs to be processed on a single machine.

For each job $J_j \in \mathcal{J}$,

- $p_j > 0$: the processing time,
- $r_j \geq 0$: the release date,
- $d_j \geq 0$: the due date,
- $\bar{d}_j \geq 0$: the deadline,
- $w_j \geq 0$: the weight,
- Preemption is allowed.

$\mathcal{J}$: the job set and the job instance.

The scheduling criterion:

sum-form

- $\sum C_j = \sum_{j=1}^{n} C_j$: the total completion time;
- $\sum T_j = \sum_{j=1}^{n} T_j$: the total tardiness;
- $\sum U_j = \sum_{j=1}^{n} U_j$: the number of tardy jobs;
- $\sum w_j C_j = \sum_{j=1}^{n} w_j C_j$: the total weighted completion time;
- $\sum w_j U_j = \sum_{j=1}^{n} w_j U_j$: the weighted number of tardy jobs;
- $\sum f_j = \sum_{j=1}^{n} f_j(C_j)$: the total scheduling cost.

max-form

- $T_{\max} = \max_{1 \le j \le n} T_j$: the maximum tardiness;
- $WC_{\max} = \max_{1 \le j \le n} w_j C_j$: the maximum weighted completion time;
- $L_{\max} = \max_{1 \le j \le n} L_j$: the maximum lateness;
- $f_{\max} = \max_{1 \le j \le n} f_j(C_j)$: the maximum scheduling cost.

Related NP-hard results:

- $1|r_j, \text{pmtn}, \bar{d}_j|\sum C_j$:
  unary NP-hard (Chen & Yuan, 2021[1])
- $1|r_j, \text{pmtn}|\sum w_j C_j$:
  unary NP-hard (Labetoulle, Lawler, Lenstra, & Rinnooy Kan 1984[2])

[1] Chen, R.B., & Yuan, J.J. (2021). Unary NP-hardness of preemptive scheduling to minimize total completion time with release times and deadlines. *Discrete Applied Mathematics,* 304, 45-54.

[2] 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.

Observation 1

For any criterion $f \in \{\sum w_j C_j, \sum U_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$,
(i) $1|r_j, pmtn|Lex(f, \sum C_j)$*: unary NP-hard,*

(ii) $1|r_j, pmtn|Lex(\sum C_j, f)$*: unknown.*

This stimulates us to study the following problems:

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

- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$
- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f_{\max})$
  where for $j = 1, 2, \ldots, n$, each $f_j(\cdot)$ is a regular function, and $f_j(t)$ is a finite number for each time $t \in [0, +\infty)$, i.e., $-\infty < f_j(t) < +\infty$.

A feasible schedule for $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, f)$
is
an optimal schedule for $1|r_j, \text{pmtn}|\sum C_j$

- $1|r_j, \text{pmtn}|\sum C_j$:
  the SRPT (shortest remaining processing time) rule is optimal. (Schrage 1968[3], Smith 1978[4])

The SRPT Rule:

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

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

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

From Schrage (1968) and Smith (1978), the following lemma holds.

Lemma 1

Every optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$ is generated by the SRPT rule in $O(n \log n)$ time. Moreover, when the instance $\mathcal{J}$ is given, all the optimal schedules have the same sequence of job completion times.

- $\Pi^*(\mathcal{J})$: the set of optimal schedules for $1|r_j, \text{pmtn}|\sum C_j$ on $\mathcal{J}$. $\Pi^*(\mathcal{J}) \leftrightarrow$ the implementations of the SRPT rule on $\mathcal{J}$.
- The n completion times of jobs of $\mathcal{J}$ in all the schedules of $\Pi^*(\mathcal{J})$: $\mathcal{C}(\mathcal{J}) = \{C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})\}$, and $C^{(1)}(\mathcal{J}) < C^{(2)}(\mathcal{J}) < \cdots < C^{(n)}(\mathcal{J})$.
- $\mathcal{T}(\mathcal{J}) = \{r_j : 1 \le j \le n\} \cup \{C^{(i)}(\mathcal{J}) : 1 \le i \le n\}$: the set of decision times in the implementation of the SRPT rule. For convenience, we write $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_m\}$ such that $\tau_1 < \tau_2 < \cdots < \tau_m$. Then $n+1 \le m \le 2n$, $\tau_1 = r_{\min}$, and $\tau_m = C^{(n)}(\mathcal{J})$.

- For each schedule σ of $\Pi^*(\mathcal{J})$ and each decision time τ_i, the remaining processing times (with repetitions being counted) at τ_i are independent of the choice of σ.
- $p_{\min}(\tau_i)$: the smallest remaining processing time at τ_i.
  $p_{\max}(\tau_i)$: the largest remaining processing time at τ_i.

- A job J_j is called **legal** at time $C^{(i)}(\mathcal{J})$ (with respect to instance $\mathcal{J}$), if J_j completes at time $C^{(i)}(\mathcal{J})$ in some schedule in $\Pi^*(\mathcal{J})$.
- $L^{(i)}(\mathcal{J})$: the set of all the jobs of $\mathcal{J}$ that are legal at time $C^{(i)}(\mathcal{J})$.

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

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

- An instance $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ is called *standard*, if $p_j = p_{\min}(r_j)$ for every job $J_j \in \mathcal{J}$.

Procedure RD-Modification:

(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 $j = 1, 2, \ldots, n$, 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**.

Lemma 2

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

Lemma 3

Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be a standard instance and let $J_j \in L^n(\mathcal{J})$, 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.

- Under release dates and preemption
  preemptive-list schedules: a useful tool

$\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$: a permutation of the n jobs of $\mathcal{J}$.

Procedure Pmtn-LS($\mathcal{O}$):

$J_{o(1)}$ is scheduled as early as possible ($[r_{o(1)}, r_{o(1)} + p_{o(1)}]$).

When the first j jobs in $\mathcal{O}$ have been scheduled and $j < n$, the job $J_{o(j+1)}$ is scheduled preemptively in the remaining idle time space (subject to its release date) as early as possible. This procedure is repeated until all the jobs are scheduled.

Pmtn-LS($\mathcal{O}$): the (preemptive) schedule obtained by the above procedure and call it the *Pmtn-LS schedule* (preemptive list schedule) determined by $\mathcal{O}$.

From Yuan, Ng, & Cheng (2015)[5], we can obtain the following result.

Lemma 4

Procedure Pmtn-LS($\mathcal{O}$) takes $O(n \log n)$ time.

- A permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ is called *completion-coinciding* if $C_{o(1)}(\text{Pmtn-LS}(\mathcal{O})) < C_{o(2)}(\text{Pmtn-LS}(\mathcal{O})) < \cdots < C_{o(n)}(\text{Pmtn-LS}(\mathcal{O}))$.

From Yuan, Ng, & Cheng (2020)[6], we can obtain the following result.

Lemma 5

For every permutation $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$, there is a completion-coinciding permutation $\mathcal{O}' = (J_{o'(1)}, J_{o'(2)}, \ldots, J_{o'(n)})$ such that Pmtn-LS($\mathcal{O}$)= Pmtn-LS($\mathcal{O}'$).

[5] 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, 147-153.

[6] 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, 860–875

- For a schedule σ,
  let $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$, where $J_{\sigma(j)}$ is the j-th completed job in σ.

Lemma 6

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 $\mathcal{J}$.

$$\Pi^*(\mathcal{J}) \overset{\text{the completion-coinciding permutation}}{\longleftrightarrow} \text{Pmtn-LS schedules}$$

Each schedule $\sigma \in \Pi^*(\mathcal{J})$ has three identities:

- σ represents an implementation of the SRPT rule on instance $\mathcal{J}$,
- σ is an optimal schedule for problem $1|r_j, \text{pmtn}|\sum C_j$ on instance $\mathcal{J}$,
- $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is a completion-coinciding permutation of $\mathcal{J}$ such that $\sigma = \text{Pmtn-LS}(\mathcal{O}^{(\sigma)})$.

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 $J_{o(j)} \in L^{(j)}(\mathcal{J})$ for $j = 1, 2, \ldots, n$.

Remark:

- The legal permutation provides a clearer description of each job in it.
- For each schedule $\sigma \in \Pi^*(\mathcal{J})$, $\mathcal{O}^{(\sigma)} = (J_{\sigma(1)}, J_{\sigma(2)}, \ldots, J_{\sigma(n)})$ is not only a TC-optimal permutation of $\mathcal{J}$ but also a legal permutation of $\mathcal{J}$.
- Each TC-optimal permutation of $\mathcal{J}$ must be a legal permutation of $\mathcal{J}$.

An important property:

Lemma 7

Let $\mathcal{O} = (J_{o(1)}, J_{o(2)}, \ldots, J_{o(n)})$ be a permutation of the n jobs in $\mathcal{J}$.
$\mathcal{O}$ is a TC-optimal permutation of $\mathcal{J}$

$$\Updownarrow$$

$\mathcal{O}$ is a legal permutation of $\mathcal{J}$.

Each legal set $L^{(i)}(\mathcal{J})$ is the set of available jobs with the smallest remaining processing time at the previous decision time just before $C^{(i)}(\mathcal{J})$.

The available jobs with the smallest remaining processing time $p_{\min}(\tau_i)$ at each decision time τ_i consist of two parts of jobs:

- the jobs newly released at time τ_i
- the jobs inherited from the previous decision time τ_{i-1}

Definition 1

Let i be an index in $\{1, 2, \ldots, m\}$ ($\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_m\}$).
(i) K_i is the set of distinct remaining processing times at time τ_i and set $k_i = |K_i|$.
*λ_i (the **optional index** at time τ_i) is the repetition number of $p_{\min}(\tau_i)$ at time τ_i.*
If $k_i > 0$, $p_{\min}(\tau_i) = \min\{q : q \in K_i\}$. If $k_i = 0$, set $p_{\min}(\tau_i) = +\infty$ and $\lambda_i = 0$.
At time τ_i, the SRPT rule has λ_i choices for processing a job in the interval $[\tau_i, \tau_{i+1}]$.
(ii) For each $q \in K_i$,
*$\mathcal{J}(i, q)$ is the set of available jobs that have the same remaining processing time q at time τ_i in all the schedules of $\Pi^*(\mathcal{J})$ and call it the q-**cluster** at time τ_i,*
(A job J_z is said to be available at time τ_i if $r_z \leq \tau_i < C_z(\sigma)$ in some schedule $\sigma \in \Pi^*(\mathcal{J})$.)
(iii) Define $\tau_0 = -\infty$, $K_0 = \emptyset$, and $k_0 = 0$.

Lemma 8

For $i \in \{1, 2, \ldots, m-1\}$ *with* $K_i \neq \emptyset$ *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*
$\mathcal{J}(i, q) = \mathcal{J}(i-1, p_{\min}(\tau_{i-1})) \cup \{J_z : r_z = \tau_i, p_z = q\}$.

Algorithm 1

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

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 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 $\mathcal{T}(\mathcal{J}) = \{\tau_1, \tau_2, \ldots, \tau_m\}$, where $\tau_1 < \tau_2 < \cdots < \tau_m$.

(2.2) Determine the index sequence $i_1, i_2, \ldots, i_n$ of $\{1, 2, \ldots, m\}$ such that $i_1 < i_2 < \cdots < i_n$ and $\tau_{i_h} = C^{(h)}(\mathcal{J})$ for $h = 1, 2, \ldots, n$.

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

(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 $\mathcal{J}(i, q) :=$

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

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

Output: $C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$ and $L^{(1)}(\mathcal{J}), L^{(2)}(\mathcal{J}), \ldots, L^{(n)}(\mathcal{J})$.

Lemma 9

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

Sometimes, we only need the last legal set $L^{(n)}(\mathcal{J})$.

Lemma 10

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 $j^* = \min\{j : J_j \in L^{(n)}(\mathcal{J})\}$.
Then $L^{(n)}(\mathcal{J}) = \{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\}$.

- Since $\mathcal{J}$ is a standard instance, for each job $J_j \in \mathcal{J}$, $p_j = p_{\min}(r_j)$.
- From Lemma 10, for each job $J_j \in L^n(\mathcal{J})$, $p_j = p_{\max}(r_j) = p_{\min}(r_j)$.

Algorithm 2

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

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 $L^{(n)}(\mathcal{J}) := \{J_j \in \mathcal{J} : j \geq j^*, p_j = p_{\max}(r_j)\}$.

Output: The set $L^{(n)}(\mathcal{J})$.

Lemma 11

For a standard instance $\mathcal{J}$, Algorithm 2 generates $L^{(n)}(\mathcal{J})$ in $O(n)$ time.

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

- $\Pi^*(\mathcal{J})$ is the set of feasible schedules of $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$
- 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 of $\mathcal{J}$ but also a legal permutation of $\mathcal{J}$
- The equivalence of the TC-optimal permutation of $\mathcal{J}$ and the legal permutation of $\mathcal{J}$

Problem $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum f_j)$ on a standard instance $\mathcal{J}$

$\Downarrow$

An $n \times n$ linear assignment problem

- The indicator variables x_{ij} of a legal permutation $\mathcal{O}$ is:

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

- The cost c_{ij} is:

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

The $n \times n$ linear assignment problem:

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

Lemma 12

Let $x^* = (x^*_{ij} : 1 \leq i, j \leq n)$ be an optimal solution for the $n \times n$ linear assignment problem stated in (1).
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 problem $1|r_j, pmtn|Lex(\sum C_j, \sum f_j)$ on instance $\mathcal{J}$.

Algorithm 3

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 1 to obtain the completion times $C^{(1)}(\mathcal{J}), C^{(2)}(\mathcal{J}), \ldots, C^{(n)}(\mathcal{J})$, and the legal sets $L^{(1)}(\mathcal{J}), L^{(2)}(\mathcal{J}), \ldots, L^{(n)}(\mathcal{J})$.

Step 2: Calculate the position cost c_{ij}, $1 \leq i, j \leq n$, and then generate the $n \times n$ linear assignment problem in (1).

Step 3: Solve the $n \times n$ linear assignment problem in (1) 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 5.1

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

★ $1|r_j, \mathrm{pmtn}|\mathrm{Lex}(\sum C_j, f_{\max})$ on a standard instance $\mathcal{J}$

Lemma 13

Consider the problem $1|r_j, pmtn|Lex(\sum C_j, f_{\max})$ on the standard instance $\mathcal{J}$ with $r_1 \leq r_2 \leq \cdots \leq r_n$, and suppose that $J_{j'} \in L^{(n)}(\mathcal{J})$ such that

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

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

Algorithm 4

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 2 on instance $\mathcal{J}_i$ to obtain $L^{(i)}(\mathcal{J}_i)$.

(1.3) Pick a job $J_{j'} \in L^{(i)}(\mathcal{J}_i)$ 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 $f_{\max}(\sigma)$.

Theorem 5.2

Algorithm 4 solves problem $1|r_j, pmtn|Lex(\sum C_j, f_{\max})$ in $O(n^2)$ time.

★ $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \tilde{f})$, $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$

Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, where $r_1 \leq r_2 \leq \ldots \leq r_n$.

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 14

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.

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.

Theorem 5.3

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* $\tilde{f} \in \{\sum w_j C_j, \sum T_j, L_{\max}, T_{\max}, WC_{\max}\}$.

- Y_j (the late work of J_j): the amount of processing of J_j scheduled after its due date d_j.
- When preemption is allowed, Y_j is not uniquely determined by C_j.

Our approach is no longer applicable to the following two problems:

- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum Y_j)$
- $1|r_j, \text{pmtn}|\text{Lex}(\sum C_j, \sum w_j Y_j)$

Thank You