https://doi.org/10.1016/j.ejor.2019.07.072

Scheduling with release dates and preemption to minimize multiple max-form objective functions

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

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

Abstract: In this paper we study multi-agent scheduling with release dates and preemption on a single machine, where the scheduling objective function of each agent to be minimized is regular and of the maximum form (max-form). The multi-agent aspect has three versions, namely ND-agent (multiple agents with non-disjoint job sets), ID-agent (multiple agents with an identical job set), and CO-agent (multiple competing agents with mutually disjoint job sets). We consider three types of problems: The first type (type-1) is the constrained scheduling problem, in which one objective function is to be minimized, subject to the restriction that the values of the other objective functions are upper bounded. The second type (type-2) is the weighted-sum scheduling problem, in which a positive combination of the objective functions is to be minimized. The third type (type-3) is the Pareto scheduling problem, for which we aim to find all the Pareto-optimal points and their corresponding Pareto-optimal schedules. We show that the type-1 problems are polynomially solvable, and the type-2 and type-3 problems are strongly NP-hard even when all jobs' release dates are zero and processing times are one. When the number of the scheduling criteria is fixed and they are all lateness-like, such as minimizing $C_{\max}, F_{\max}, L_{\max}, T_{\max}$, and $WC_{\max}$, where $WC_{\max}$ is the maximum weighted completion time of the jobs, the type-2 and type-3 problems are polynomially solvable. To address the type-3 problems, we develop a new solution technique that guesses the Pareto-optimal points through some elaborately constructed schedule-configurations.

Key words: machine scheduling; multi-criteria; multi-agent; Pareto scheduling

*Corresponding author. Email address: daniel.ng@polyu.edu.hk

1 Introduction

In multi-criteria scheduling, we have m scheduling objective functions $f^{(1)}, f^{(2)}, \cdots, f^{(m)}$ to be minimized. In this paper we only consider the regular scheduling objective functions, whereby a scheduling objective function f is *regular* if it is a function nondecreasing in the completion times of the jobs. Let α be the machine environment and β the processing requirements of the jobs. The following three types of problems are often studied in multi-objective scheduling research.

Constrained Scheduling Problem (CSP): $\alpha|\beta|f^{(m)} : f^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1$, where $X_1, X_2, \cdots, X_{m-1}$ are $m-1$ given threshold values. The goal of the problem is to find a feasible schedule π that minimizes $f^{(m)}(\pi)$, subject to the constraint that $f^{(i)}(\pi) \leq X_i$ for $i = 1, 2, \cdots, m-1$.

Weighted-Sum Scheduling Problem (WSP): $\alpha|\beta|\sum_{i=1}^{m} \lambda_i f^{(i)}$, where $\lambda_1, \lambda_2, \cdots, \lambda_m$ are positive weights. The goal of the problem is to find a feasible schedule π that minimizes $\sum_{i=1}^{m} \lambda_i f^{(i)}(\pi)$.

Pareto Scheduling Problem (PSP): $\alpha|\beta|(f^{(1)}, f^{(2)}, \cdots, f^{(m)})$, where the third field indicates that there are m independent minimization criteria. The goal of the problem is to find all the Pareto-optimal points and their corresponding Pareto-optimal schedules. We give the formal definitions of Pareto-optimal points and Pareto-optimal schedules in Section 2.1.

As observed by Hoogeveen [15], and T'Kindt and Billaut [27], there are some basic relations between the above three types of problems. Results on the CSP can be taken as preprocessing for research on the corresponding WSP and PSP. The NP-hardness of the CSP or WSP implies the NP-hardness of the corresponding PSP. The polynomial solvability of the PSP implies the polynomial solvability of the corresponding CSP and WSP (although the time complexity for solving such problems may sometimes be very different). Moreover, each optimal schedule for the WSP is also a Pareto-optimal schedule for the corresponding PSP. For convenience, we use

$$\alpha|\beta|\{f^{(1)}, f^{(2)}, \cdots, f^{(m)}\}$$

to denote all the above three multi-criteria scheduling problems.

In this paper we consider multi-criteria scheduling on a single machine with release dates and preemption. So α is 1 and β is "r_j, pmtn". Moreover, the m scheduling criteria are of the max-form. Consequently, we express $f^{(1)}, f^{(2)}, \cdots, f^{(m)}$ as $f^{(1)}_{\max}, f^{(2)}_{\max}, \cdots, f^{(m)}_{\max}$, respectively, in the sequel.

Problem Formulation: We present the problems studied in this paper in the multi-agent scheduling framework introduced by Agnetis et al. [1] as follows: We are given n jobs $\mathcal{J} = \{1, 2, \cdots, n\}$, where each job j has a release date $r_j \geq 0$ and a processing time $p_j > 0$, that are to be scheduled preemptively on a single machine. There are m agents $\{1, 2, \cdots, m\}$ and m subsets $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \cdots, \mathcal{J}^{(m)}$ of $\mathcal{J}$, where $\mathcal{J}^{(i)}$ is owned by agent i and the jobs in $\mathcal{J}^{(i)}$ are called the jobs of agent i for $i = 1, 2, \cdots, m$. We use $n_i = |\mathcal{J}^{(i)}|$

to denote the number of jobs of agent i. Then a natural assumption is that

$$\mathcal{J}^{(1)} \cup \mathcal{J}^{(2)} \cup \cdots \cup \mathcal{J}^{(m)} = \mathcal{J} = \{1, 2, \cdots, n\}. \tag{1}$$

Moreover, each job $j \in \mathcal{J}^{(i)}$ has a due date $d_j^{(i)}$ and a weight $w_j^{(i)} \geq 0$ with respect to agent i, $1 \leq i \leq m$. We assume in this paper that all the data $r_j, p_j, d_j^{(i)}$, and $w_j^{(i)}$ are integers.

Given a schedule π of the n jobs, we use the following notation throughout the paper.

- $C_j(\pi)$ is the *completion time* of job j under schedule π.
- $F_j(\pi) = C_j(\pi) - r_j$ is the *flow time* of job j under schedule π.
- $f_j^{(i)}(C_j(\pi))$ is the *cost* of job $j \in \mathcal{J}^{(i)}$ with respect to agent i in schedule π, where $f_j^{(i)}(\cdot)$ is a nondecreasing function over $[0, +\infty)$.
- $L_j^{(i)}(\pi) = C_j(\pi) - d_j^{(i)}$ is the *lateness* of job $j \in \mathcal{J}^{(i)}$ with respect to agent i under schedule π.
- $T_j^{(i)}(\pi) = \max\{0, C_j(\pi) - d_j^{(i)}\} = \max\{0, L_j^{(i)}(\pi)\}$ is the *tardiness* of job $j \in \mathcal{J}^{(i)}$ with respect to agent i under schedule π.
- $w_j^{(i)} C_j(\pi)$ is the *weighted completion time* of job $j \in \mathcal{J}^{(i)}$ with respect to agent i under schedule π.
- $C_{\max}^{(i)}(\pi) = \max\{C_j(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *makespan* of agent i under schedule π.
- $F_{\max}^{(i)}(\pi) = \max\{F_j(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *maximum flow time* of agent i under schedule π.
- $L_{\max}^{(i)}(\pi) = \max\{L_j^{(i)}(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *maximum lateness* of agent i under schedule π.
- $T_{\max}^{(i)}(\pi) = \max\{T_j^{(i)}(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *maximum tardiness* of agent i under schedule π.
- $WC_{\max}^{(i)}(\pi) = \max\{w_j^{(i)} C_j(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *maximum weighted completion time* of agent i under schedule π.
- $f_{\max}^{(i)}(\pi) = \max\{f_j^{(i)}(\pi) : j \in \mathcal{J}^{(i)}\}$ is the *maximum cost* of agent i under schedule π. We call $f_{\max}^{(i)}$ the scheduling objective function of agent i. Then each of the objective functions $C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}$, and $WC_{\max}^{(i)}$ is a specific choice for $f_{\max}^{(i)}$.

According to Agnetis et al. [1], there are two typical versions of multi-agent scheduling models related to our research as follows:

– If $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \cdots, \mathcal{J}^{(m)}$ are mutually disjoint, then the m agents are called *competing agents*. In this case, we add CO in the β field to describe the scheduling problem and call it *CO-agent scheduling*. Then we denote our CO-agent scheduling problems by

$$1|r_j, \text{pmtn}, \text{CO}|\{f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}\}.$$

– If $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \cdots, \mathcal{J}^{(m)}$ are not restricted to be mutually disjoint, then the m agents are called *non-disjoint agents*. In this case, we add ND to the β field to describe the

scheduling problem and call it *ND-agent scheduling*. Then we denote our ND-agent scheduling problems by

$$1|r_j, \text{pmtn, ND}|\{f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}\}.$$

It is evident that CO-agent scheduling is a special version of ND-agent scheduling.

The classical multi-criteria scheduling problem to minimize m max-form scheduling criteria is in fact another special version of ND-agent scheduling. In this case, we have $\mathcal{J}^{(1)} = \mathcal{J}^{(2)} = \cdots = \mathcal{J}^{(m)} = \mathcal{J}$. Since the m agents have identical sets of jobs, we add ID to the β field to describe the scheduling problem and call it *ID-agent scheduling*. Then we denote our ID-agent scheduling problems by

$$1|r_j, \text{pmtn, ID}|\{f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}\}.$$

Research Motivation: The scheduling problems considered in this paper are motivated by cooperation and competition of multiple agents, where each agent can be regarded as a firm, a factory, or an investor. When several agents cooperate to work on a joint project, each agent seeks to minimize its own loss or maximize its own profit, yet they need to work cooperatively to complete the project. So the scheduling of the joint activities of the agents in carrying out the project needs to strike a proper balance between the benefit and cost to each participating agent. As stated in Agnetis et al. [2], management problems in which multiple agents compete on the usage of a common processing resource are receiving increasing attention in different application environments and different methodological fields. Agnetis et al. [1] presented applications of multi-agent scheduling in various contexts, including job re-scheduling, railway scheduling, aircraft landing, multi-project scheduling, cross-docking distribution, and communication network. T'Kindt and Billaut [27] discussed applications of multi-criteria scheduling in the contexts of bottle manufacturing, electroplating and chemical processing, steel manufacturing, car assembling, cheque processing, transport scheduling, timetabling, sports scheduling, and satellite scheduling. The common feature in these examples is that m agents with different objectives work together to perform n jobs.

Literature Review: Multi-criteria scheduling research has been rapidly developing in the past two decades. Some classical results play important roles in the research on this topic. For example, Hoogeveen [14] addressed the two-criteria scheduling problems $1||\{f_{\max}^{(1)}, f_{\max}^{(2)}\}$ and the three-criteria scheduling problems $1||\{f_{\max}^{(1)}, f_{\max}^{(2)}, f_{\max}^{(3)}\}$, and Hoogeveen and van de Velde [16] addressed the two-criteria scheduling problem $1||\{\sum C_j, f_{\max}\}$. The methodologies applied in these studies, which are commonly adopted in subsequent related research, have helped the development of this research stream. Multi-criteria scheduling has become a very popular topic in scheduling research. With over a thousand papers published on this topic, Hoogeveen [15], Nelson et al. [21], T'kindt and Billaut [27], and Agnetis et al. [1] provided comprehensive reviews of the related research results.

Multi-agent scheduling models were first introduced by Agnetis et al. [2], and Baker and Smith [5]. Some early research on multi-agent scheduling can be found in Agnetis

et al. [2, 3], Baker and Smith [5], Cheng et al. [8, 9], Ng et al. [22], and Yuan et al. [32]. The phenomenal growth of multi-agent scheduling has confirmed its importance in scheduling research. Some recent research on multi-agent scheduling includes Agnetis et al. [1], Gao and Yuan [11], Gao et al. [12], He and Leung [13], Li et al. [20], Oron et al. [23], Sadi and Soukhal [24], Sadi et al. [25], Yuan et al. [31], and Yuan [28, 29, 30].

In this paper we only consider the max-form scheduling criteria. We summarize the most related known results in Table 1.

Table 1: The complexity results of related research in the literature.

Problem	Complexity	Reference
$1\|\text{ID}\|f_{\max}^{(m)} : f_{\max}^{(i)}\ \forall i \leq m-1$	$O(mn^2)$	Hoogeveen [14]
$1\|\text{ID}\|\lambda_1 f_{\max}^{(1)} + \lambda_2 f_{\max}^{(2)}$	$O(n^4)$	Hoogeveen [14]
$1\|\text{ID}\|(f_{\max}^{(1)}, f_{\max}^{(2)})$	$O(n^4)$	Hoogeveen [14]
$1\|\text{ID}\|L_{\max}^{(2)} : L_{\max}^{(1)}$	$O(n \log n)$	Hoogeveen [14]
$1\|\text{ID}\|\lambda_1 L_{\max}^{(1)} + \lambda_2 L_{\max}^{(2)}$	$O(n^3 \log n)$	Hoogeveen [14]
$1\|\text{ID}\|(L_{\max}^{(1)}, L_{\max}^{(2)})$	$O(n^3 \log n)$	Hoogeveen [14]
$1\|\text{ID}\|\lambda_1 f_{\max}^{(1)} + \lambda_2 f_{\max}^{(2)} + \lambda_3 f_{\max}^{(3)}$	$O(n^8)$	Hoogeveen [14]
$1\|\text{ID}\|(f_{\max}^{(1)}, f_{\max}^{(2)}, f_{\max}^{(3)})$	$O(n^8)$	Hoogeveen [14]
$1\|\text{ID}\|\sum_{i=1}^{m} \lambda_i f_{\max}^{(i)}$	Strongly NP-hard	Hoogeveen [14]
$1\|r_j, \text{pmtn, ID}\|f_{\max}^{(2)} : f_{\max}^{(1)}$	$O(n^2)$	Sourd [26]
$1\|r_j, \text{pmtn, ID}\|\lambda_1 f_{\max}^{(1)} + \lambda_2 f_{\max}^{(2)}$	$O(n^4)$	Sourd [26]
$1\|r_j, \text{pmtn, ID}\|(f_{\max}^{(1)}, f_{\max}^{(2)})$	$O(n^4)$	Sourd [26]
$1\|\text{CO}\|f_{\max}^{(2)} : f_{\max}^{(1)}$	$O(n^2)$	Agnetis et al. [2]
$1\|\text{CO}\|\lambda_1 f_{\max}^{(1)} + \lambda_2 f_{\max}^{(2)}$	$O(n_1 n_2 n^2)$	Agnetis et al. [2]
$1\|\text{CO}\|(f_{\max}^{(1)}, f_{\max}^{(2)})$	$O(n_1 n_2 n^2)$	Agnetis et al. [1]
$1\|\text{CO}\|L_{\max}^{(2)} : L_{\max}^{(1)}$	$O(n \log n)$	Yuan et al. [32]
$1\|\text{CO}\|\lambda_1 L_{\max}^{(1)} + \lambda_2 L_{\max}^{(2)}$	$O(n_1 n_2 n)$	Yuan et al. [32]
$1\|\text{CO}\|(L_{\max}^{(1)}, L_{\max}^{(2)})$	$O(n_1 n_2 n)$	Agnetis et al. [1]
$1\|\text{CO}\|\{C_{\max}^{(1)}, C_{\max}^{(2)}\}$	$O(n)$	Agnetis et al. [1]
$1\|\text{CO}\|f_{\max}^{(m)} : f_{\max}^{(i)}\ \forall i \leq m-1$	$O(n^2)$	Agnetis et al. [3]
$1\|\text{CO}\|\sum_{i=1}^{m} \lambda_i C_{\max}^{(i)}$ with fixed m	$O(n)$	Agnetis et al. [1]
$1\|\text{CO}\|\sum_{i=1}^{m} \lambda_i L_{\max}^{(i)}$ with fixed m	$O((n_1 \cdots n_m)^m n \log n)$	Agnetis et al. [1]
$1\|\text{CO}\|\sum_{i=1}^{m} \lambda_i f_{\max}^{(i)}$ with fixed m	Pseudo-polynomial time	Cheng et al. [9]
$1\|\text{CO}, w_j^{(i)} = w^{(i)}\|\sum_{i=1}^{m} \lambda_i W C_{\max}^{(i)}$	$O(n + m \log m)$	Cheng et al. [9]
$1\|\text{CO}\|\sum_{i=1}^{m} W C_{\max}^{(i)}$	Strongly NP-hard	Cheng et al. [9]
$1\|\text{CO}\|\sum_{i=1}^{m} L_{\max}^{(i)}$ or $\sum_{i=1}^{m} T_{\max}^{(i)}$	Binary NP-hard	Cheng et al. [9]
$1\|\text{CO}\|\sum_{i=1}^{m} \lambda_i L_{\max}^{(i)}$ or $\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$	Strongly NP-hard	Yuan [28]
$1\|r_j, \text{pmtn, CO}\|f_{\max}^{(2)} : f_{\max}^{(1)}$	$O(n^2)$	Leung et al. [19]
$1\|\text{ND}\|\{C_{\max}^{(1)}, C_{\max}^{(2)}\}$	$O(n)$	Agnetis et al. [1]
$1\|\text{ND}\|L_{\max}^{(2)} : L_{\max}^{(1)}$	$O(n \log n)$	Agnetis et al. [1]
$1\|\text{ND}\|f_{\max}^{(2)} : f_{\max}^{(1)}$	$O(n^2)$	Agnetis et al. [1]
$1\|\text{ND}\|\sum_{i=1}^{m} \lambda_i L_{\max}^{(i)}$ with fixed m	$O(n)$	Agnetis et al. [1]
$1\|\text{ND}\|f_{\max}^{(m)} : f_{\max}^{(i)}\ \forall i \leq m-1$	$O(mn^2)$	Agnetis et al. [1]

Methodology Discussion: For problem $1|\text{prec}|f_{\max}$, Lawler [17] presented an $O(n^2)$ algorithm, called Lawler's rule, for solving the problem. Lawler's rule has a huge impact on scheduling research. Almost all the studies concerning the max-form scheduling criterion may use Lawler's rule or its derivatives. For example, Baker et al. [4] presented an $O(n^2)$

algorithm to solve problem $1|r_j, \text{pmtn}|f_{\max}$ and Hoogeveen [14] presented an $O(mn^2)$ algorithm to solve problem $1||f_{\max}^{(m)} : f_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1$. In Section 3 we extend the results in Baker et al. [4] and Hoogeveen [14] to devise a method to solve the constrained ND-agent scheduling problem $1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(m)} : f_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1$.

The "ε-constraint method" (see Hoogeveen [15] and T'kindt and Billaut [27]) is an efficient way to solve the Pareto scheduling problem $\alpha|\beta|(f, g)$, where f and g are two regular scheduling objective functions to be minimized. The idea underpinning the ε-constraint method is as follows: *Suppose that $\tilde{x}$ is a number such that the constrained problem $\alpha|\beta|g : f \leq \tilde{x}$ is feasible. Let y be the optimal value of problem $\alpha|\beta|g : f \leq \tilde{x}$ and let x be the optimal value of problem $\alpha|\beta|f : g \leq y$. Then (x, y) is a Pareto-optimal point of problem $\alpha|\beta|(f, g)$.*

As described in Hoogeveen [15] and T'kindt and Billaut [27], the ε-constraint method can be used to generate all the Pareto-optimal points of problem $\alpha|\beta|(f, g)$ by iteratively solving problems $\alpha|\beta|g : f \leq x$ and $\alpha|\beta|f : g \leq y$.

Agnetis et al. [2] studied the Pareto CO-agent scheduling problem $1|\text{CO}|(f^{(A)}, f^{(B)})$ using the "schedule-structure analysis" approach, where there are two competing agents A and B with scheduling objective functions $f^{(A)}$ and $f^{(B)}$, respectively. They showed that, for $X \in \{A, B\}$, if $f^{(X)} = \sum C_j^{(X)}$, then the X-jobs are scheduled in the shortest processing time (SPT) order in a Pareto-optimal schedule, and if $f^{(X)} = L_{\max}^{(X)}$, then the X-jobs are scheduled in the earliest due date (EDD) order in a Pareto-optimal schedule. Applying the schedule-structure analysis approach, Agnetis et al. [2] showed that the Pareto CO-agent scheduling problems $1|\text{CO}|(\sum C_j^{(A)}, L_{\max}^{(B)})$ and $1|\text{CO}|(L_{\max}^{(A)}, L_{\max}^{(B)})$ are polynomially solvable. Following Agnetis et al. [2], Agnetis et al. [3], Cheng et al. [8, 9], Oron et al. [23], and Yuan et al. [32] also adopted the schedule-structure analysis approach to address related problems in their studies.

For the Pareto scheduling problem $\alpha|\beta|(f^{(1)}, f^{(2)}, \cdots, f^{(m)})$, we present in Section 2.1 two lemmas (Lemmas 2.1 and 2.2) to replace the role of the ε-constraint method. For the case where each of the objective functions $f_{\max}^{(i)}$, $i = 1, 2, \cdots, m$, is *lateness-like* (defined in Section 2.3), our problem $1|r_j, \text{pmtn}, \text{ND}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ can be solved by a new technique by guessing the Pareto-optimal points through some elaborately constructed schedule-configurations.

It is noted that we use in this paper the Pmtn-LS schedule (defined in Section 2.2) for scheduling jobs with release dates and preemption. Based on some basic properties of the Pmtn-LS schedule, we can only consider permutations of the jobs for scheduling. We believe that this technique can be effectively used to address other scheduling problems with release dates and preemption.

Our Contributions: In this paper we study the single-machine multi-agent scheduling problem $1|r_j, \text{pmtn}, \text{A}|\{f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}\}$, where $\text{A} \in \{\text{ND}, \text{ID}, \text{CO}\}$. For most cases, we assume that each $f_{\max}^{(i)}$ is a lateness-like objective function, $i = 1, 2, \ldots, m$. We summarize our main findings in Table 2.

Table 2: Complexity results of polynomially solvable problems in this paper.

m	r_j, pmtn	Agent	Problem	$f_{\max}^{(i)}$	Time complexity	References
Any	r_j, pmtn	ND	CSP	Regular	$O(mn^2)$	Theorem 3.1
Any	r_j, pmtn	ID	CSP	Regular	$O(mn^2)$	Theorem 3.1
Any	r_j, pmtn	CO	CSP	Regular	$O(n^2)$	Theorem 3.2
Any	$r_j = 0$, no pmtn	ND	CSP	Lateness-like	$O(mn + n\log n)$	Theorem 3.4
Any	$r_j = 0$, no pmtn	ID	CSP	Lateness-like	$O(mn + n\log n)$	Theorem 3.4
Any	$r_j = 0$, no pmtn	CO	CSP	Lateness-like	$O(n\log n)$	Theorem 3.4
Fixed	r_j, pmtn	ND	PSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n^2)$	Theorem 4.2
Fixed	r_j, pmtn	ND	WSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n^2)$	Theorem 4.2
Fixed	r_j, pmtn	ID	PSP	Lateness-like	$O(n^{m^2-m+2})$	Theorem 4.2
Fixed	r_j, pmtn	ID	WSP	Lateness-like	$O(n^{m^2-m+2})$	Theorem 4.2
Fixed	r_j, pmtn	CO	PSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n)$	Theorem 4.5
Fixed	r_j, pmtn	CO	WSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n)$	Theorem 4.5
Fixed	$r_j = 0$, no pmtn	ND	PSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n\log n)$	Theorem 4.3
Fixed	$r_j = 0$, no pmtn	ND	WSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n\log n)$	Theorem 4.3
Fixed	$r_j = 0$, no pmtn	ID	PSP	Lateness-like	$O(n^{m^2-m+1}\log n)$	Theorem 4.3
Fixed	$r_j = 0$, no pmtn	ID	WSP	Lateness-like	$O(n^{m^2-m+1}\log n)$	Theorem 4.3
Fixed	$r_j = 0$, no pmtn	CO	PSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n)$	Theorem 4.5
Fixed	$r_j = 0$, no pmtn	CO	WSP	Lateness-like	$O((n_1\cdots n_m)^{m-1}n)$	Theorem 4.5

Moreover, we show that the weighted-sum multi-agent scheduling problems $1|\mathrm{ID}, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$, $1|\mathrm{ID}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$, $1|\mathrm{ID}, p_j = 1|\sum_{i=1}^{m} T_{\max}^{(i)}$, and $1|\mathrm{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$, and $1|\mathrm{CO}, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$ are strongly NP-hard.

Organization of the Paper: We organize the rest of the paper as follows: In Section 2 we present some preliminaries. In Section 3 we show that the constrained multi-agent scheduling problems are polynomially solvable. In Section 4 we show that, when m is fixed and the scheduling criteria are lateness-like, the multi-agent Pareto scheduling problems are polynomially solvable. In Section 5 we provide strong NP-hardness results for some restricted weighted-sum multi-agent scheduling problems. We conclude the paper and suggest topics for future research in Section 6.

2 Some preliminaries

In Section 2.1 we formally define the Pareto scheduling problem and provide some basic properties. In Section 2.2 we introduce the Pmtn-LS schedule for scheduling jobs with release dates and preemption, followed by some discussions. In Section 2.3 we introduce the lateness-like criteria and the EDD-like permutations.

2.1 Pareto scheduling problems

Consider the Pareto scheduling problem $\alpha|\beta|(f^{(1)}, f^{(2)}, \cdots, f^{(m)})$. The *objective vector* of a feasible schedule π is given by $(f^{(1)}(\pi), f^{(2)}(\pi), \cdots, f^{(m)}(\pi))$. An m-vector $(X_1, X_2, \cdots, X_m)$ is called a *Pareto-optimal point* if there is a feasible schedule π^* such

that $(f^{(1)}(\pi^*), f^{(2)}(\pi^*), \cdots, f^{(m)}(\pi^*)) = (X_1, X_2, \cdots, X_m)$ and there is no feasible schedule π with

$$(f^{(1)}(\pi), f^{(2)}(\pi), \cdots, f^{(m)}(\pi)) < (X_1, X_2, \cdots, X_m),$$

i.e., $f^{(i)}(\pi) \leq X_i$ for all i and at least one of the inequalities is strict. In this case, π^* is called a *Pareto-optimal schedule* corresponding to $(X_1, X_2, \cdots, X_m)$.

The following lemma is directly implied from the above definition.

Lemma 2.1. *Let $(X_1, X_2, \cdots, X_m)$ be a Pareto-optimal point. Then each optimal schedule of the constrained scheduling problem*

$$\alpha|\beta|f^{(m)} : f^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1$$

is a Pareto-optimal schedule corresponding to $(X_1, X_2, \cdots, X_m)$.

Suppose that we are given a set of m-vectors $\mathcal{X}$ and we are informed that $\mathcal{X}$ includes all the Pareto-optimal points of problem $\alpha|\beta|(f^{(1)}, f^{(2)}, \cdots, f^{(m)})$. Usually, some vectors in $\mathcal{X}$ may not be Pareto-optimal. Then a question arises: How do we determine whether or not a vector $(X_1, X_2, \cdots, X_m) \in \mathcal{X}$ is Pareto-optimal? We provide the following lemma to address the question.

Lemma 2.2. *Let $(X_1, X_2, \cdots, X_m)$ be an m-vector. Then $(X_1, X_2, \cdots, X_m)$ is a Pareto-optimal point of problem $\alpha|\beta|(f^{(1)}, f^{(2)}, \cdots, f^{(m)})$ if and only if, for each $k \in \{1, 2, \cdots, m\}$, the constrained scheduling problem*

$$\alpha|\beta|f^{(k)} : f^{(i)} \leq X_i \ \forall i \neq k$$

is feasible and has the optimal value X_k.

Proof. The necessity ($\Rightarrow$) is implied in Lemma 2.1 directly.

To prove the sufficiency ($\Leftarrow$), we suppose to the contrary that $(X_1, X_2, \cdots, X_m)$ is not a Pareto-optimal point. Since, for each $k \in \{1, 2, \cdots, m\}$, the constrained scheduling problem $\alpha|\beta|f^{(k)} : f^{(i)} \leq X_i, \forall i \neq k$, is feasible and has the optimal value X_k, we conclude that there must be a Pareto-optimal point $(X'_1, X'_2, \cdots, X'_m)$ such that $(X'_1, X'_2, \cdots, X'_m) < (X_1, X_2, \cdots, X_m)$. Let π' be a Pareto-optimal schedule corresponding to $(X'_1, X'_2, \cdots, X'_m)$ and let $k \in \{1, 2, \cdots, m\}$ be such that $X'_k < X_k$. Then π' is a feasible schedule for problem $\alpha|\beta|f^{(k)} : f^{(i)} \leq X_i \ \forall i \neq k$. The fact that $f^{(k)}(\pi') = X'_k < X_k$ implies that the optimal value of problem $\alpha|\beta|f^{(k)} : f^{(i)} \leq X_i \ \forall i \neq k$ is less than X_k. This contradicts the hypothesis. The lemma follows. □

2.2 Pmtn-LS schedule

A nonpreemtive schedule $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ of the n jobs $\mathcal{J} = \{1, 2, \cdots, n\}$ is called a *permutation schedule* if the jobs are scheduled consecutively without preemption from time 0 to $\sum_{j=1}^{n} p_j$ in the order $\pi(1), \pi(2), \cdots, \pi(n)$. In this case, we have

$C_{\pi(j)}(\pi) = p_{\pi(1)} + p_{\pi(2)} + \cdots + p_{\pi(j)}$ for $j = 1, 2, \cdots, n$. Given a regular scheduling objective function $f(\cdot)$, it is well known that the optimal value of problem $1||f$ can be achieved by a permutation schedule. We will see that the feasible schedules for problem $1|r_j, \text{pmtn}|f$ do not differ much from the permutation schedules.

Given a permutation (list) $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ of the n jobs $\mathcal{J} = \{1, 2, \cdots, n\}$, the following algorithm, called Pmtn-LS(π), generates a feasible preemptive list schedule for problem $1|r_j, \text{pmtn}|f$.

Pmtn-LS(π): Schedule the jobs preemptively using the strategy that, at any decision point τ (when some jobs are completed or some jobs are released), schedule the remaining part of the first available job in the list π. By "a job is available at time τ", we mean that the job is released by time τ and has not been completed. The schedule obtained by Pmtn-LS(π) is called the *Pmtn-LS schedule* determined by π.

As given in Yuan et al. [31], Pmtn-LS(π) can be implemented in the following way: *We first schedule job $\pi(1)$ as early as possible. When the first j jobs in π have been scheduled and $j < n$, we schedule the $(j+1)$-st job $\pi(j+1)$ preemptively in the remaining idle time space as early as possible. We repeat this procedure until all the jobs are scheduled.*

In the above discussion, the term "time space" is understood as "a set of time intervals" or "a set of time periods". We use such a term repeatedly in the following discussion. For example, the time space occupied by a schedule π refers to the set of time intervals in which the machine is busy in π.

Yuan et al. [31] showed that each of the above two implementations of Pmtn-LS(π) runs in $O(n \log n)$ time. For convenience, for a permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$, we also use π to denote the Pmtn-LS schedule determined by π. Then $S_j(\pi)$ (starting time), $C_j(\pi)$ (completion time), $f_j(C_j(\pi))$, and $f_{\max}(\pi)$ have their meanings.

It is noted that, for distinct permutations π and π' of the n jobs, the two Pmtn-LS schedules determined by π and π', respectively, occupy the same time space, so we have $C_{\max}(\pi) = C_{\max}(\pi')$. Based on this fact, for each $\mathcal{J}' \subseteq \mathcal{J}$, we use the following two notation in our discussion:

- $\mathcal{T}(\mathcal{J}')$ denotes the time space occupied by a Pmtn-LS schedule of $\mathcal{J}'$.
- $C(\mathcal{J}')$ denotes the makespan of a Pmtn-LS schedule of $\mathcal{J}'$.

Recall that the earliest release date (ERD) order of the n jobs $\mathcal{J} = \{1, 2, \cdots, n\}$ is a permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ of the n jobs such that $r_{\pi(1)} \leq r_{\pi(2)} \leq \cdots \leq r_{\pi(n)}$.

Lemma 2.3. *Suppose that the ERD order of the n jobs in $\mathcal{J}$ is given in advance. Then $\mathcal{T}(\mathcal{J})$ and $C(\mathcal{J})$ can be calculated in $O(n)$ time.*

Proof. Let $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ be the ERD order of the n jobs in $\mathcal{J}$ given in advance. Then $r_{\pi(1)} \leq r_{\pi(2)} \leq \cdots \leq r_{\pi(n)}$. Given the ERD order, the schedule obtained by algorithm Pmtn-LS(π) has no preemption. Then $C_j(\pi) = S_j(\pi) + p_j$ for every job j. Set $C_{\pi(0)} = 0$. From the implementation of Pmtn-LS(π), for $i = 1, 2, \cdots, n$, the values

$S_{\pi(i)}(\pi)$ and $C_{\pi(i)}(\pi)$ can be calculated iteratively in the following way:

$$\begin{cases} S_{\pi(i)}(\pi) &= \max\{r_{\pi(i)}, C_{\pi(i-1)}(\pi)\}, \\ C_{\pi(i)}(\pi) &= S_{\pi(i)}(\pi) + p_{\pi(i)}. \end{cases}$$

Clearly, the time complexity for calculating these values is given by $O(n)$. Especially, $C(\mathcal{J}) = C_{\pi(n)}(\pi)$ can be calculated in $O(n)$ time. Since $\mathcal{T}(\mathcal{J}) = \bigcup_{i=1}^{n}[S_{\pi(i)}(\pi), C_{\pi(i)}(\pi)]$, $\mathcal{T}(\mathcal{J})$ can also be calculated in $O(n)$ time. The lemma follows. □

Given a permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ of the n jobs in $\mathcal{J}$, for each $k \in \{1, 2, \cdots, n\}$, we introduce the following three notation:

- $\pi|_k = (\pi(1), \pi(2), \cdots, \pi(k))$, which restricts the permutation π to its first k jobs.
- $\mathcal{J}_k^\pi = \{\pi(1), \pi(2), \cdots, \pi(k)\}$, which is the set of the first k jobs in the permutation π.
- $\mathcal{S}_k^\pi = \{j : C_j(\pi) \le C_k(\pi)\}$, which is the set of jobs completed by time $C_k(\pi)$ in π.

From the implementation of Pmtn-LS(π), we have

$$C_{\pi(k)}(\pi) = C(\mathcal{S}_{\pi(k)}^\pi) \le C(\mathcal{J}_k^\pi) = C_{\max}(\pi|_k) \text{ for all } k = 1, 2, \cdots, n. \tag{2}$$

Note that all the values in (2) can be obtained in $O(n \log n)$ time by running algorithm Pmtn-LS(π). For the case where $r_j = 0$ for all the jobs, all the values in (2) are given by $p_{\pi(1)} + p_{\pi(2)} + \cdots + p_{\pi(k)}$, $k = 1, 2, \cdots, n$, so they can be obtained in $O(n)$ time.

Lemma 2.4. *The Pmtn-LS schedule dominates all the feasible schedules, i.e., for every feasible schedule σ of the n jobs, there is a permutation π of the n jobs such that $C_j(\pi) \le C_j(\sigma)$, $j = 1, 2, \cdots, n$.*

Proof. For a given feasible schedule σ of the n jobs, let $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ be the permutation of the n jobs so that $C_{\pi(1)}(\sigma) < C_{\pi(2)}(\sigma) < \cdots < C_{\pi(n)}(\sigma)$. Given $j \in \{1, 2, \cdots, n\}$, let k be the index such that $\pi(k) = j$, i.e., j is the k-th job in the permutation π. Then $\mathcal{J}_k^\pi = \{\pi(1), \pi(2), \cdots, \pi(k)\}$. Since Pmtn-LS($\pi|_k$) schedules the jobs in $\mathcal{J}_k^\pi$ preemptively as early as possible, we have $C_{\max}(\pi|_k) \le \max\{C_{\pi(1)}(\sigma), C_{\pi(2)}(\sigma), \cdots, C_{\pi(k)}(\sigma)\} = C_{\pi(k)}(\sigma)$. From (2), we conclude that $C_j(\pi) = C_{\pi(k)}(\pi) \le C_{\max}(\pi|_k) \le C_{\pi(k)}(\sigma) = C_j(\sigma)$. This proves the lemma. □

Since we only consider regular scheduling objective functions, from Lemma 2.4, it suffices to consider the Pmtn-LS schedules in the rest of the paper.

Note that $C(\mathcal{J}_1^\pi) \le C(\mathcal{J}_2^\pi) \le \cdots \le C(\mathcal{J}_n^\pi)$ for every permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$. Moreover, from (2), we have $C_{\pi(k)}(\pi) \le C(\mathcal{J}_k^\pi)$ for $k = 1, 2, \cdots, n$, but we cannot guarantee the equalities because of the release dates. A permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ is called *completion-coinciding* if $C_{\pi(1)}(\pi) < C_{\pi(2)}(\pi) < \cdots < C_{\pi(n)}(\pi)$. For completion-coinciding permutations, we clearly have $\mathcal{S}_{\pi(k)}^\pi = \mathcal{J}_k^\pi$ and $C_{\pi(k)}(\pi) = C(\mathcal{S}_{\pi(k)}^\pi) = C(\mathcal{J}_k^\pi)$ for $k = 1, 2, \cdots, n$, so $C(\mathcal{J}_1^\pi) < C(\mathcal{J}_2^\pi) < \cdots < C(\mathcal{J}_n^\pi)$.

Not all the permutations are completion-coinciding. However, the following lemma enables us to consider completion-coinciding permutations if necessary.

Lemma 2.5. *For each permutation π, the new permutation π^* obtained from π by listing the n jobs in increasing order of their completion times in π is a completion-coinciding permutation such that Pmtn-LS(π) and Pmtn-LS(π^*) generate the same schedule, implying $C_j(\pi^*) = C_j(\pi)$ for $j = 1, 2, \cdots, n$.*

Proof. Let $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$. If π is a completion-coinciding permutation, we have nothing to do. Then we assume in the following that π is not completion-coinciding.

Let k be the largest index in $\{1, 2, \cdots, n\}$ such that $C_{\pi(k)}(\pi) < C_{\pi(i)}(\pi)$ for some index i with $i < k$. For our purpose, we may choose the index i such that $C_{\pi(i)}(\pi) = C(\mathcal{J}_k^\pi)$. From the choices of k and i and from (2), we have

$$C_{\pi(j)}(\pi) < C_{\pi(i)}(\pi) = C(\mathcal{J}_k^\pi) \text{ for } j \in \{1, 2, \cdots, k\} \setminus \{i\} \tag{3}$$

and

$$C_{\pi(i)}(\pi) < C_{\pi(k+1)}(\pi) < C_{\pi(k+1)}(\pi) < \cdots < C_{\pi(n)}(\pi). \tag{4}$$

If $r_{\pi(i)} < \max\{C_{\pi(j)} : j = i+1, i+2, \cdots, k\}$, then there is some index $j \in \{i+1, i+2, \cdots, k\}$ such that $r_{\pi(i)} < C_{\pi(j)}(\pi)$. Since $i < j$, job $\pi(i)$ takes precedence of job $\pi(j)$ in permutation π. From the implementation of algorithm Pmtn-LS(π), job $\pi(j)$ completes after job $\pi(i)$ in Pmtn-LS(π). This means that $C_{\pi(j)}(\pi) > C_{\pi(i)}(\pi)$, contradicting the inequality in (3). Consequently, we have

$$r_{\pi(i)} \geq \max\{C_{\pi(j)} : j = i+1, i+2, \cdots, k\}. \tag{5}$$

Let $\pi' = (\pi'(1), \pi'(2), \cdots, \pi'(n))$ such that

$$\pi'(j) = \begin{cases} \pi(j), & \text{for } j = 1, 2, \cdots, i-1, \\ \pi(j+1), & \text{for } j = i, i+1, \cdots, k-1, \\ \pi(i), & \text{for } j = k, \\ \pi(j), & \text{for } j = k+1, k+2, \cdots, n. \end{cases} \tag{6}$$

Then π' is the permutation obtained from π by shifting $\pi(i)$ to the position after $\pi(k)$. When we run algorithm Pmtn-LS(π'), from the first line in (6), the schedule for the $i-1$ jobs $\pi'(j) = \pi(j)$, $1 \leq j \leq i-1$, in Pmtn-LS(π') is the same as that in Pmtn-LS(π). From (5) and from the second line in (6), the schedule of the $k-i$ jobs $\pi'(j) = \pi(j+1)$, $i \leq j \leq k-1$, in Pmtn-LS(π') is the same as that in Pmtn-LS(π). So, just when the $k-1$ jobs $\pi'(1), \pi'(2), \cdots, \pi'(k-1)$ are scheduled in Pmtn-LS(π'), the time space occupied by job $\pi'(k) = \pi(i)$ in π is still idle. It follows that the schedule for job $\pi'(k)$ in Pmtn-LS(π') is certainly the same as that in Pmtn-LS(π). From the first three lines in (6), we know that $\mathcal{J}_k^{\pi'} = \mathcal{J}_k^\pi$, so $\mathcal{T}(\mathcal{J}_k^{\pi'}) = \mathcal{T}(\mathcal{J}_k^\pi)$. Thus, from the last line in (6), the schedule for the remaining $n-k$ jobs $\pi'(j) = \pi(j)$, $k+1 \leq j \leq n$, in Pmtn-LS(π') is the same as that in Pmtn-LS(π).

The above discussion reveals that π and π' satisfy the following three properties: (i) The two algorithms Pmtn-LS(π') and Pmtn-LS(π) generate the same schedule, (ii) $C_{\pi(k)}(\pi) < C(\mathcal{J}_k^\pi) < C_{\pi(k+1)}(\pi) < C_{\pi(k+2)}(\pi) < \cdots < C_{\pi(n)}(\pi)$, and (iii)

$C_{\pi'(k-1)}(\pi') < C(\mathcal{J}_k^{\pi'}) = C_{\pi'(k)}(\pi') < C_{\pi'(k+1)}(\pi') < \cdots < C_{\pi'(n)}(\pi')$. Consequently, by at most $n-1$ repetitions of the above procedure, we eventually obtain a permutation $\pi^* = (\pi^*(1), \pi^*(2), \cdots, \pi^*(n))$ of the n jobs in $\mathcal{J}$ such that Pmtn-LS(π^*) and Pmtn-LS(π) generate the same schedule and $C_{\pi^*(1)}(\pi^*) < C_{\pi^*(2)}(\pi^*) < \cdots < C_{\pi^*(n)}(\pi^*)$. Now we complete the proof by noting that π^* is obtained from π by listing the n jobs in increasing order of their completion times in π. □

Suppose that $\mathcal{J}'$ is a nonempty subset of $\mathcal{J} = \{1, 2, \cdots, n\}$. A job $x \in \mathcal{J}'$ is called *interfering* with respect to $\mathcal{J}'$ if $C(\mathcal{J}' \setminus \{x\}) < C(\mathcal{J}')$. Note that if $r_j = 0$ for all the jobs, then all the jobs in $\mathcal{J}'$ are interfering with respect to $\mathcal{J}'$. But in the general environment "r_j, pmtn", it is not the case.

Lemma 2.6. *Suppose that $\mathcal{J}'$ and $\mathcal{J}''$ are two subsets of $\mathcal{J}$ such that $\mathcal{J}' \subset \mathcal{J}''$. If $C(\mathcal{J}') < C(\mathcal{J}'')$, then there is an interfering job x with respect to $\mathcal{J}''$ such that $x \in \mathcal{J}'' \setminus \mathcal{J}'$.*

Proof. Note that $\mathcal{J}''$ is the disjoint union of $\mathcal{J}'$ and $\mathcal{J}'' \setminus \mathcal{J}'$. Let π be a permutation of the jobs in $\mathcal{J}''$ such that the jobs in $\mathcal{J}'$ are listed before the jobs in $\mathcal{J}'' \setminus \mathcal{J}'$. Then the maximum completion time of the jobs in $\mathcal{J}'$ is given by $\max\{C_j(\pi) : j \in \mathcal{J}'\} = C(\mathcal{J}')$ and the maximum completion time of the jobs in $\mathcal{J}''$ is given by $\max\{C_j(\pi) : j \in \mathcal{J}''\} = C(\mathcal{J}'')$. Let x be the last completed job in π, i.e., $C_x(\pi) = C(\mathcal{J}'')$. Then $C(\mathcal{J}'' \setminus \{x\}) < C(\mathcal{J}'')$. Since $C(\mathcal{J}') < C(\mathcal{J}'')$, we have $x \in \mathcal{J}'' \setminus \mathcal{J}'$. Thus, x is a required interfering job with respect to $\mathcal{J}''$. The lemma follows. □

Since the original proposition is equivalent to its converse negative proposition, Lemma 2.6 can be equivalently stated in the following form:

Lemma 2.6′. *Suppose that $\mathcal{J}'$ and $\mathcal{J}''$ are two subsets of $\mathcal{J}$ such that $\mathcal{J}' \subset \mathcal{J}''$. If every job in $\mathcal{J}'' \setminus \mathcal{J}'$ is not interfering with respect to $\mathcal{J}''$, then $C(\mathcal{J}') = C(\mathcal{J}'')$.*

Lemma 2.7. *Let $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ be a completion-coinciding permutation of the jobs in $\mathcal{J}$, and let x and k be two indices with $1 \le x < k \le n$. Let π' be the new permutation obtained from π by shifting $\pi(x)$ to the position just after $\pi(k)$, i.e.,*

$$\pi' = (\pi(1), \cdots, \pi(x-1), \pi(x+1), \cdots, \pi(k), \pi(x), \pi(k+1), \cdots, \pi(n)).$$

Then we have

(i) $C_{\pi(x)}(\pi') = C_{\pi'(k)}(\pi') \le C_{\pi(k)}(\pi)$,

(ii) $C_j(\pi') \le C_j(\pi)$ for every job $j \in \mathcal{J} \setminus \{\pi(x)\}$,

(iii) $C_j(\pi') = C_j(\pi)$ for every job $j \in \{\pi(k+1), \pi(k+2), \cdots, \pi(n)\}$, and

(iv) if $\pi(x)$ is an interfering job with respect to $\mathcal{J}_k^{\pi}$, then $C_{\pi(k)}(\pi') < C_{\pi(x)}(\pi') = C_{\pi(k)}(\pi)$.

Proof. Recall that $\mathcal{J}_h^{\pi} = \{\pi(1), \pi(2), \cdots, \pi(h)\}$ for $h \in \{1, 2, \cdots, n\}$. Since π is a completion-coinciding permutation of the jobs in $\mathcal{J}$, we have

$$C_{\pi(h)}(\pi) = C_{\max}(\pi|_h) = C(\mathcal{J}_h^{\pi}) \text{ for } h \in \{1, 2, \cdots, n\}. \tag{7}$$

The definitions of π, π', x, and k imply that $\pi(x) = \pi'(k)$ and $\mathcal{J}_k^{\pi'} = \mathcal{J}_k^{\pi}$. From (2) and (7), we have

$$C_{\pi(x)}(\pi') = C_{\pi'(k)}(\pi') \leq C_{\max}(\pi'|_k) = C(\mathcal{J}_k^{\pi'}) = C(\mathcal{J}_k^{\pi}) = C_{\pi(k)}(\pi). \tag{8}$$

This proves (i).

Now let $j \in \mathcal{J} \setminus \{\pi(x)\}$. Then there are some $h, h' \in \{1, 2, \cdots, n\}$ such that $j = \pi(h) = \pi'(h')$. Note that $h \neq x$ and $h' \neq k$ since $j \neq \pi(x)$ and $j \neq \pi'(k)$. By the definitions of π and π', we have $\mathcal{J}_{h'}^{\pi'} \subseteq \mathcal{J}_h^{\pi}$ (in fact, if $h \leq x - 1$ or $h \geq k + 1$, we have $h = h'$ and $\mathcal{J}_{h'}^{\pi'} = \mathcal{J}_h^{\pi}$, and if $x + 1 \leq h \leq k$, we have $h = h' + 1$ and $\mathcal{J}_{h'}^{\pi'} = \mathcal{J}_h^{\pi} \setminus \{\pi(x)\}$). This implies that $C(\mathcal{J}_{h'}^{\pi'}) \leq C(\mathcal{J}_h^{\pi})$. From (2) and (7) again, we have

$$C_j(\pi') = C_{\pi'(h')}(\pi') \leq C_{\max}(\pi'|_{h'}) = C(\mathcal{J}_{h'}^{\pi'}) \leq C(\mathcal{J}_h^{\pi}) = C_{\max}(\pi|_h) = C_{\pi(h)}(\pi) = C_j(\pi).$$

This proves (ii).

The result in (iii) follows from (7) and the fact that, for $h = k + 1, k + 2, \cdots, n$, we have $C_{\pi(h)}(\pi') = C(\mathcal{S}_{\pi(h)}^{\pi'}) = C(\mathcal{J}_h^{\pi})$.

To prove (iv), we assume that $\pi(x)$ is an interfering job with respect to $\mathcal{J}_k^{\pi}$. Then we have

$$C_{\max}(\pi'|_{k-1}) = C(\mathcal{J}_{k-1}^{\pi'}) = C(\mathcal{J}_k^{\pi} \setminus \{\pi(x)\}) < C_{\pi(k)}(\pi), \tag{9}$$

where the first equality follows from (2). From (8) and (9), the only possibility is that $C_{\pi(k)}(\pi') < C_{\pi'(k)}(\pi') = C_{\max}(\pi'|_k) = C_{\pi(k)}(\pi)$, or equivalently, $C_{\pi(k)}(\pi') < C_{\pi(x)}(\pi') = C_{\pi(k)}(\pi)$. This proves the lemma. □

The above results on Pmtn-LS schedules enable us to deploy a new method to deal with preemptive scheduling problems (with job release dates): Using $C(\mathcal{J}')$ with $\mathcal{J}' \subseteq \mathcal{J}$ as a parameter, it suffices to consider permutations of the jobs in $\mathcal{J}'$. In most cases, we need not obtain $C(\mathcal{J}')$ by generating a Pmtn-LS schedule of $\mathcal{J}'$; instead, we use the lemmas established in this subsection repeatedly. As a result, we do not consider the release dates directly in the sequel.

2.3 Lateness-like criteria and EDD-like permutations

Consider multi-agent scheduling in which $f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}$ are the scheduling objective functions of the m agents $\{1, 2, \cdots, m\}$, respectively. The scheduling criterion of minimizing $f_{\max}^{(i)}$ is called *lateness-like* if $f_{\max}^{(i)}$ is regular and there is a permutation, denoted by $O_i = (i1, i2, \cdots, in_i)$, of the n_i jobs in $\mathcal{J}^{(i)}$ such that, for every time instant τ, we have

$$f_{i1}^{(i)}(\tau) \geq f_{i2}^{(i)}(\tau) \geq \cdots \geq f_{in_i}^{(i)}(\tau). \tag{10}$$

In this case, $f_{\max}^{(i)}$ is also called a *lateness-like objective function* and O_i is an optimal permutation for problem $1|r_j, \text{pmtn}|f_{\max}^{(i)}$. This in fact follows from the idea for solving problem $1|r_j, \text{pmtn}|f_{\max}$ in Baker et al. [4]. We call O_i an *EDD-like permutation* of agent i, $i = 1, 2, \cdots, m$.

Given an EDD-like permutation of agent i, for each job $j \in \mathcal{J}^{(i)}$, we use $j^{(s)}$ to denote the *position index* of job j in O_i, i.e., $j = ij^{(s)}$.

The lateness-like criterion and EDD-like permutation are so named because the problem $1|r_j, \text{pmtn}|L_{\max}$, which minimizes the maximum lateness, has an optimal permutation in which the jobs are sequenced in the EDD order. If $f_{\max}^{(i)} = L_{\max}^{(i)}$, by setting $O_i = (i1, i2, \cdots, in_i)$ as the EDD order of the jobs in $\mathcal{J}^{(i)}$, i.e., $d_{i1} \leq d_{i2} \leq \cdots \leq d_{in_i}$, then the relations in (10) hold obviously. This is the intuition behind the definitions of the lateness-like criterion and EDD-like permutation.

Some common lateness-like scheduling criteria are minimizing $C_{\max}, F_{\max}, L_{\max}, T_{\max}$, and $WC_{\max}$. For $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, Table 3 depicts the corresponding EDD-like permutation O_i and the time complexity for generating O_i.

Table 3: EDD-like permutations O_i with respect to $f_{\max}^{(i)}$.

$f_{\max}^{(i)}$	$O_i = (i1, i2, \cdots, in_i)$	Time Complexity
$C_{\max}^{(i)}$	An arbitrary order	$O(n)$
$F_{\max}^{(i)}$	$r_{i1} \leq r_{i2} \leq \cdots \leq r_{in_i}$	$O(n \log n)$
$L_{\max}^{(i)}$	$d_{i1}^{(i)} \leq d_{i2}^{(i)} \leq \cdots \leq d_{in_i}^{(i)}$	$O(n \log n)$
$T_{\max}^{(i)}$	$d_{i1}^{(i)} \leq d_{i2}^{(i)} \leq \cdots \leq d_{in_i}^{(i)}$	$O(n \log n)$
$WC_{\max}^{(i)}$	$w_{i1}^{(i)} \geq w_{i2}^{(i)} \geq \cdots \geq w_{in_i}^{(i)}$	$O(n \log n)$

We can observe from Table 3 that, for $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, the corresponding EDD-like permutations O_i can be obtained in at most $O(n \log n)$ time. This time complexity is usually dominated by the final time complexity for solving a multi-agent problem. Thus, when a lateness-like criterion of minimizing $f_{\max}^{(i)}$ is considered and its concrete form is uncertain, we always assume that the EDD-like permutation O_i is given in advance.

3 Constrained multi-agent scheduling problems

Given an $(m-1)$-vector $(X_1, X_2, \cdots, X_{m-1})$, we consider the following two constrained multi-agent scheduling problems

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(m)} : f_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1, \tag{11}$$

$$1|r_j, \text{pmtn}, \text{CO}|f_{\max}^{(m)} : f_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1. \tag{12}$$

Then the problem in (12) is a subproblem of the problem in (11). We take the convention in this section that, for a time τ, a job $j \in \mathcal{J}$ and an agent $i \in \{1, 2, \cdots, m\}$,

$$f_j^{(i)}(\tau) = -\infty \text{ if } j \notin \mathcal{J}^{(i)}. \tag{13}$$

A job $j \in \mathcal{J}$ is called *legal* at time τ if $\tau \geq r_j + p_j$ and $f_j^{(i)}(\tau) \leq X_i$ for all $i \in \{1, 2, \cdots, m-1\}$. Hence, if no job is legal at time $C(\mathcal{J})$, then the problem in (11)

is infeasible. Subsequently, we design algorithms that always check if a job $j \in \mathcal{J}$ is legal at time $C(\mathcal{J})$. Since we clearly have $C(\mathcal{J}) \geq r_j + p_j$, to check the legality of job j at time $C(\mathcal{J})$, we only need to check the relations $f_j^{(i)}(C(\mathcal{J})) \leq X_i$ for all $i \in \{1, 2, \cdots, m-1\}$. Generally, we have the following lemma to determine the last job in an optimal permutation.

Lemma 3.1. *Suppose that the problem in (11) is feasible and let x be a legal job at time $C(\mathcal{J})$ such that $f_x^{(m)}(C(\mathcal{J}))$ is as small as possible. Then there is an optimal schedule (permutation) $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ for the problem such that $\pi(n) = x$.*

Proof. To prove the lemma, we assume that $\pi' = (\pi'(1), \pi'(2), \cdots, \pi'(n))$ is an optimal completion-coinciding permutation for the problem. If $\pi'(n) = x$, we are done by setting $\pi = \pi'$. Suppose that $\pi'(n) \neq x$. Then we set π as the permutation obtained from π' by shifting x to the last position of the permutation. From Lemma 2.7, we have $C_x(\pi) \leq C_{\pi'(n)}(\pi') = C(\mathcal{J})$ and $C_j(\pi) \leq C_j(\pi')$ for all $j \in \{1, 2, \cdots, n\} \setminus \{x\}$. Since x is legal at time $C(\mathcal{J})$, π is certainly a feasible permutation for the problem such that $\pi(n) = x$. Now, $f_j^{(m)}(C_j(\pi)) \leq f_j^{(m)}(C_j(\pi')) \leq f_{\max}^{(m)}(\pi')$ for all $j \in \mathcal{J}^{(m)} \setminus \{x\}$ and, from the choice of x, we have $f_x^{(m)}(C_x(\pi)) \leq f_x^{(m)}(C(\mathcal{J})) \leq f_{\pi'(n)}^{(m)}(C(\mathcal{J})) = f_{\pi'(n)}^{(m)}(C_{\pi'(n)}(\pi')) \leq f_{\max}^{(m)}(\pi')$. Consequently, π is an optimal schedule with $\pi(n) = x$. The lemma follows. □

We notice that in a feasible Pmtn-LS schedule determined by the permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ with $\pi(n) = x$, the maximum completion time of the jobs in $\mathcal{J}$ is $C(\mathcal{J})$ and the jobs in $\mathcal{J} \setminus \{x\}$ occupy the time space $\mathcal{T}(\mathcal{J} \setminus \{x\})$, so job x is processed in the first p_x units of time in the time space $[r_x, C(\mathcal{J})] \setminus \mathcal{T}(\mathcal{J} \setminus \{x\})$, which is sufficient for processing job x because of the feasibility of the permutation. This further implies that assigning x as the last job in an optimal permutation does not affect the subsequent steps to find the final optimal solution.

The above discussion enables us to present the following algorithm.

Algorithm 3.1. For the problem in (11) with given $X_1, X_2, \cdots, X_{m-1}$.

Step 1. Set $\mathcal{J} := \{1, 2, \cdots, n\}$ and $k := n$. Moreover, sort the n jobs in $\mathcal{J}$ in the ERD order and denote the ERD order of the jobs by $\vec{\mathcal{J}}$.

Step 2. Calculate $C(\mathcal{J})$ by using the ERD order $\vec{\mathcal{J}}$ and set $\tau := C(\mathcal{J})$.

Step 3. Generate the set $\mathcal{L}(\tau)$ that consists of all the jobs $j \in \mathcal{J}$ legal at time τ, i.e.,

$$f_j^{(i)}(\tau) \leq X_i \text{ for all } i \in \{1, 2, \cdots, m-1\}. \tag{14}$$

• If $\mathcal{L}(\tau) \neq \emptyset$, pick a job $x \in \mathcal{L}(\tau)$ such that $f_x^{(m)}(\tau)$ is as small as possible. Then go to Step 4.

• If $\mathcal{L}(\tau) = \emptyset$, i.e., no job in $\mathcal{J}$ is legal at time τ, then terminate the algorithm and output *infeasibility*.

Step 4. Define $\pi(k) = x$ and do the following:

- If $k \geq 2$, then set $\mathcal{J} := \mathcal{J} \setminus \{x\}$, $\vec{\mathcal{J}} := \vec{\mathcal{J}} \setminus \{x\}$ and $k := k - 1$, and go to Step 2.
- If $k = 1$, then go to Step 5.

Step 5. Output the optimal permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ and, if necessary, run algorithm Pmtn-LS(π) to obtain the final schedule and calculate $f_{\max}^{(m)}(\pi)$.

Theorem 3.1. *Algorithm 3.1 solves the problem in (11) in $O(mn^2)$ time.*

Proof. The correctness of Algorithm 3.1 follows from Lemma 3.1. Algorithm 3.1 has n iterations. Step 1 runs in $O(n \log n)$ time, which is the time used to sort the jobs in the ERD order $\vec{\mathcal{J}}$. Note that Step 1 is not included in the iterations of the algorithm. Step 2 runs in $O(n)$ time, which follows from Lemma 2.3 since the ERD order $\vec{\mathcal{J}}$ is given in advance. Step 3 runs in $O(mn)$ time, which is dominated by the time used for checking the $O(mn)$ inequalities in (14). Step 4 runs in $O(n)$ time. Thus, the time complexity of each iteration of the algorithm is given by $O(mn)$. It follows that the overall running time of Algorithm 3.1 is $O(mn^2)$. □

For the constrained CO-agent scheduling problem in (12), since $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \cdots, \mathcal{J}^{(m)}$ form a partition of $\mathcal{J} = \{1, 2, \cdots, n\}$, the running time of Step 3 in Algorithm 3.1 is $O(n)$. Then the time complexity of Algorithm 3.1 reduces to $O(n^2)$ in this case. Consequently, we have the following result.

Theorem 3.2. *The problem in (12) is solvable in $O(n^2)$ time.*

The result in Theorem 3.2 seems the best possible since, up to now, the best time complexity for solving problem $1||f_{\max}$ is $O(n^2)$.

When $r_j = 0$ for all jobs and all the m criteria are of max-lateness form, the following two special problems can be more efficiently solved:

$$1|\text{ND}|L_{\max}^{(m)} : L_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1, \tag{15}$$

$$1|\text{CO}|L_{\max}^{(m)} : L_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1, \tag{16}$$

Theorem 3.3. *The problem in (15) is solvable in $O(mn + n \log n)$ time and the problem in (16) is solvable in $O(n \log n)$ time.*

Proof. For convenience, for a job j and an agent i with $j \notin \mathcal{J}^{(i)}$, we define $d_j^{(i)} = +\infty$. This takes the same rule as the convention in (13). Moreover, we define $X^{(m)} = +\infty$.

For the ND-agent version, the restriction "$L_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1$" requires that the completion time C_j of each job j should satisfy $C_j - d_j^{(i)} \leq X_i$ for $i = 1, 2, \cdots, m-1$, which induces a deadline

$$\bar{d}_j = \min\{X_i + d_j^{(i)} : i = 1, 2, \cdots, m\}$$

on job j, $j = 1, 2, \cdots, n$. The deadlines of the n jobs can be determined in $O(mn)$ time. So the ND-agent problem in (15) reduces to problem $1|\bar{d}_j|L_{\max}^{(m)}$ in $O(mn)$ time.

For the CO-agent version, we only need to calculate the deadlines by setting

$$\bar{d}_j = X_i + d_j^{(i)} \text{ for the unique } i \text{ with } j \in \mathcal{J}^{(i)}.$$

Then the deadlines of the n jobs can be determined in $O(n)$ time and the CO-agent problem in (16) reduces to problem $1|\bar{d}_j|L_{\max}^{(m)}$ in $O(n)$ time.

Note that $1|\bar{d}_j|L_{\max}^{(m)}$ is in fact a subproblem of problem $1|\bar{d}_j|L_{\max}$ on the n jobs $\{1, 2, \cdots, n\}$. For $t \in (0, +\infty)$ and $j \in \{1, 2, \cdots, n\}$, we define $f_j(t) = t - d_j$ if $t \le \bar{d}_j$ and $f_j(t) = +\infty$ if $t > \bar{d}_j$. Then problem $1|\bar{d}_j|L_{\max}$ is the same as problem $1||f_{\max}$. Lawler's algorithm for solving this special problem can be stated as follows:

(A1): *Starting at time $\tau := \sum_{j=1}^{n} p_j$, schedule an unscheduled and legal job j, i.e., $\bar{d} \ge \tau$, with the maximum due date d_j in the interval $[\tau - p_j, \tau]$. Reset $\tau := \tau - p_j$ and repeat the above procedure. If there are no unscheduled and legal jobs in some iteration with $\tau > 0$, then the problem is infeasible.*

For this special problem $1|\bar{d}_j|L_{\max}$, the time complexity of algorithm A1 can be reduced to $O(n \log n)$ by introducing a simple data structure in the following way:

At each decision point τ of algorithm A1, a list $\vec{\mathcal{J}}$ is used to sort the unscheduled and illegal jobs in nondecreasing order of their deadlines, and another list $\vec{\mathcal{S}}$ is used to sort the unscheduled and legal jobs in nondecreasing order of their due dates. Initially, $\vec{\mathcal{J}}$ sorts all the jobs and $\vec{\mathcal{S}}$ is empty. We set $\tau := \sum_{j=1}^{n} p_j$ and start the implementation of algorithm A1. If some job in $\vec{\mathcal{J}}$ is legal at time τ, then the last job in $\vec{\mathcal{J}}$, say, j, is legal at time τ, i.e., $\bar{d}_j \ge \tau$. Then we delete j from $\vec{\mathcal{J}}$ and insert j in the job list $\vec{\mathcal{S}}$ by a binary search so that the jobs in the updated $\vec{\mathcal{S}}$ are still listed in the EDD order. Whence $\vec{\mathcal{J}}$ and $\vec{\mathcal{S}}$ have been generated in the current iteration, and $\vec{\mathcal{S}}$ is nonempty, we pick the last job of $\vec{\mathcal{S}}$, say, i, delete i from $\vec{\mathcal{S}}$, and schedule job i in the time interval $[\tau - p_j, \tau]$. After this, we set $\tau := \tau - p_i$ and proceed to the next iteration.

Now the time complexity $O(n \log n)$ follows from the fact that deleting an item from a sorted list or inserting an item in a sorted list by a binary search takes $O(\log n)$ time.

The above discussion implies that the problem in (15) is solvable in $O(mn + n \log n)$ time and the problem in (16) is solvable in $O(n \log n)$ time. The result follows. □

Remark: If $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ for all $i = 1, 2, \cdots, m$, we can slightly modify the proof of Theorem 3.3 to establish the following result.

Theorem 3.4. *Suppose that $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ for all $i = 1, 2, \cdots, m$. Then problem*

$$1|ND|f_{\max}^{(m)} : f_{\max}^{(i)} \le X_i \ \forall i = 1, 2, \cdots, m-1$$

is solvable in $O(mn + n \log n)$ time and problem

$$1|CO|f_{\max}^{(m)} : f_{\max}^{(i)} \le X_i \ \forall i = 1, 2, \cdots, m-1$$

is solvable in $O(n \log n)$ time.

4 Pareto multi-agent scheduling problems

Let m, the number of agents, be a fixed number. Let $f_{\max}^{(i)}$ be the scheduling objective function of agent $i \in \{1, 2, \cdots, m\}$. We assume in this section that each criterion of minimizing $f_{\max}^{(i)}$ is lateness-like. Let $O_i = (i1, i2, \cdots, in_i)$ be an EDD-like permutation of the n_i jobs in $\mathcal{J}^{(i)}$, $i = 1, 2, \cdots, m$.

4.1 The ND-agent version

We consider the following Pareto ND-agent scheduling problem

$$1|r_j, \text{pmtn, ND}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}). \tag{17}$$

Recall that when we state that π is a schedule of $\mathcal{J} = \{1, 2, \cdots, n\}$, we mean that π refers to a permutation of the n jobs in $\mathcal{J}$ and we also refer to the Pmtn-LS schedule determined by the permutation π.

For a given schedule π, a job h_i is called the *bottleneck job* of agent i under π if $h_i \in \mathcal{J}^{(i)}$ and h_i is the last job (in terms of completion time) in the schedule such that

$$f_{h_i}^{(i)}(C_{h_i}(\pi)) = \max_{j \in \mathcal{J}^{(i)}} f_j^{(i)}(C_j(\pi)) = f_{\max}^{(i)}(\pi).$$

In this case, we also say that h_i is a *bottleneck job* under schedule π. Note that each agent has a unique bottleneck job under π, but distinct agents may have a common bottleneck job. Thus there may be repetitions in the sequence $h_1, h_2, \cdots, h_m$. Sometimes it is useful to re-order the m agents by a permutation $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(m))$ such that the bottleneck jobs are listed in nondecreasing order of their completion times in π, i.e., $C_{h_{\sigma(1)}}(\pi) \leq C_{h_{\sigma(2)}}(\pi) \leq \cdots \leq C_{h_{\sigma(m)}}(\pi)$. The bottleneck jobs have the following useful property.

Lemma 4.1. *Let iq be the bottleneck job of agent i under schedule π. Then $C_{iq'}(\pi) < C_{iq}(\pi)$ for all $q' = 1, 2, \cdots, q-1$, where $C_{iq}(\pi)$ is the completion time of job iq in π.*

Proof. Suppose to the contrary that there is some $q' \in \{1, 2, \cdots, q-1\}$ such that $C_{iq'}(\pi) > C_{iq}(\pi)$. Then we have

$$f_{\max}^{(i)}(\pi) \geq f_{iq'}^{(i)}(C_{iq'}(\pi)) \geq f_{iq'}^{(i)}(C_{iq}(\pi)) \geq f_{iq}^{(i)}(C_{iq}(\pi)) = f_{\max}^{(i)}(\pi),$$

where the second and third inequalities follow from the fact that all the scheduling objective functions are regular and lateness-like. Then we have $f_{\max}^{(i)}(\pi) = f_{iq'}^{(i)}(C_{iq'}(\pi))$, so iq is not the last completed job, assuming $f_{\max}^{(i)}(\pi)$ in π. This contradicts the assumption that iq is the bottleneck job of agent i under π. The lemma follows. □

A *schedule-configuration* of the problem in (17) is defined as a pair $(\sigma, \mathbf{Q})$, where $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(m))$ is a permutation of the m agents $\{1, 2, \cdots, m\}$, and

$$\mathbf{Q} = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & \cdots & q_{1,m-1} \\ q_{2,1} & q_{2,2} & \cdots & \cdots & q_{2,m-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ q_{m,1} & q_{m,2} & \cdots & \cdots & q_{m,m-1} \end{pmatrix}$$

is an $m \times (m-1)$-matrix of nonnegative integers such that

$$0 \le q_{s,1} \le q_{s,2} \le \cdots \le q_{s,m-1} \le n_{\sigma(s)} \text{ for } s = 1, 2, \cdots, m, \tag{18}$$

and

$$q_{1,1}, q_{2,2}, \cdots, q_{m-1,m-1} \ge 1. \tag{19}$$

For convenience, we denote the matrix $\mathbf{Q}$ as $\mathbf{Q} = (q_{s,t})_{m\times(m-1)}$ in the sequel.

To grasp the essence of a schedule-configuration $(\sigma, \mathbf{Q})$, we can regard that the s-th row of $\mathbf{Q}$ is associated with agent $\sigma(s)$ for $s = 1, 2, \cdots, m$. In the s-th row $(q_{s,1}, q_{s,2}, \cdots, q_{s,m-1})$, each entry $q_{s,t}$ has two meanings: it refers to the $q_{s,t}$-th job $\sigma(s)q_{s,t}$ of agent $\sigma(s)$ and the set of the first $q_{s,t}$ jobs $\mathcal{J}_{q_{s,t}}^{O_{\sigma(s)}} = \{\sigma(s)1, \sigma(s)2, \cdots, \sigma(s)q_{s,t}\}$ of agent $\sigma(s)$. Thus, from the s-th row $(q_{s,1}, q_{s,2}, \cdots, q_{s,m-1})$, we can generate a sequence of $m-1$ subsets of the job set of agent $\sigma(s)$

$$(\mathcal{J}_{q_{s,1}}^{O_{\sigma(s)}}, \mathcal{J}_{q_{s,2}}^{O_{\sigma(s)}}, \cdots, \mathcal{J}_{q_{s,m-1}}^{O_{\sigma(s)}}) \tag{20}$$

such that

$$\mathcal{J}_{q_{s,1}}^{O_{\sigma(s)}} \subseteq \mathcal{J}_{q_{s,2}}^{O_{\sigma(s)}} \subseteq \cdots \subseteq \mathcal{J}_{q_{s,m-1}}^{O_{\sigma(s)}}. \tag{21}$$

By taking each sequence in (20) as a row, we obtain the following $m \times (m-1)$-matrix

$$\mathcal{J}^{(\sigma,\mathbf{Q})} = \begin{pmatrix} \mathcal{J}_{q_{1,1}}^{O_{\sigma(1)}} & \mathcal{J}_{q_{1,2}}^{O_{\sigma(1)}} & \cdots & \cdots & \mathcal{J}_{q_{1,m-1}}^{O_{\sigma(1)}} \\ \mathcal{J}_{q_{2,1}}^{O_{\sigma(2)}} & \mathcal{J}_{q_{2,2}}^{O_{\sigma(2)}} & \cdots & \cdots & \mathcal{J}_{q_{2,m-1}}^{O_{\sigma(2)}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{J}_{q_{m,1}}^{O_{\sigma(m)}} & \mathcal{J}_{q_{m,2}}^{O_{\sigma(m)}} & \cdots & \cdots & \mathcal{J}_{q_{m,m-1}}^{O_{\sigma(m)}} \end{pmatrix},$$

which consists of $m \times (m-1)$ subsets of $\mathcal{J}$.

Now, for a schedule-configuration $(\sigma, \mathbf{Q})$, we define $m-1$ subsets of $\mathcal{J} = \{1, 2, \cdots, n\}$ by setting for each $t = 1, 2, \cdots, m-1$

$$\mathcal{J}_t^{(\sigma,\mathbf{Q})} = \mathcal{J}_{q_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{q_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{q_{m,t}}^{O_{\sigma(m)}}. \tag{22}$$

Note that $\mathcal{J}_t^{(\sigma,\mathbf{Q})}$ is the union of the m subsets in the t-th column of matrix $\mathcal{J}^{(\sigma,\mathbf{Q})}$. From (21) and (22), we have

$$\mathcal{J}_1^{(\sigma,\mathbf{Q})} \subseteq \mathcal{J}_2^{(\sigma,\mathbf{Q})} \subseteq \cdots \subseteq \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}, \tag{23}$$

so

$$C(\mathcal{J}_1^{(\sigma,\mathbf{Q})}) \leq C(\mathcal{J}_2^{(\sigma,\mathbf{Q})}) \leq \cdots \leq C(\mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}). \tag{24}$$

Given a schedule-configuration $(\sigma, \mathbf{Q})$, the principal diagonal vector of matrix $\mathbf{Q}$, i.e., $(q_{1,1}, q_{2,2}, \cdots, q_{m-1,m-1})$, plays an important role in our analysis. For each $t \in \{1, 2, \cdots, m-1\}$, let $h_{\sigma(t)}$ be the $q_{t,t}$-th job of agent $\sigma(t)$, i.e.,

$$h_{\sigma(t)} = \sigma(t)q_{t,t}, \text{ or equivalently, } h_{\sigma(t)}^{(\sigma(t))} = q_{t,t}. \tag{25}$$

Then we obtain a vector

$$\mathbf{h}(\sigma, \mathbf{Q}) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$$

of $m-1$ job indices. We call $\mathbf{h}(\sigma, \mathbf{Q})$ the *principal vector* induced by $(\sigma, \mathbf{Q})$. Since $1 \leq q_{t,t} \leq n_{\sigma(t)}$ for all $t = 1, 2, \cdots, m-1$, we see that the principal vector $\mathbf{h}(\sigma, \mathbf{Q})$ induced by $(\sigma, \mathbf{Q})$ is well-defined.

The purposes of the $m-1$ sets $\mathcal{J}_1^{(\sigma,\mathbf{Q})} \subseteq \mathcal{J}_2^{(\sigma,\mathbf{Q})} \subseteq \cdots \subseteq \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}$ and the principal vector $\mathbf{h}(\sigma, \mathbf{Q}) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$ are to calculate the following $m-1$ values

$$X_{\sigma(t)}(\sigma, \mathbf{Q}) = f_{h_{\sigma(t)}}^{(\sigma(t))}(C(\mathcal{J}_t^{(\sigma,\mathbf{Q})})), \quad t = 1, 2, \cdots, m-1. \tag{26}$$

Finally, we define $X_{\sigma(m)}(\sigma, \mathbf{Q})$ as the optimal value of the constrained scheduling problem

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(\sigma(m))} : f_{\max}^{(\sigma(t))} \leq X_{\sigma(t)}(\sigma, \mathbf{Q}) \ \forall t = 1, 2, \cdots, m-1. \tag{27}$$

With the $m-1$ values in (26) and the value $X_{\sigma(m)}(\sigma, \mathbf{Q})$ in hand, we obtain an m-vector

$$\mathbf{X}(\sigma, \mathbf{Q}) = (X_1(\sigma, \mathbf{Q}), X_2(\sigma, \mathbf{Q}), \cdots, X_m(\sigma, \mathbf{Q})). \tag{28}$$

We call $\mathbf{X}(\sigma, \mathbf{Q})$ the *objective vector* of the schedule-configuration $(\sigma, \mathbf{Q})$.

The reason for the above elaboration is to come up with a well-designed procedure for solving the Pareto multi-agent scheduling problem in (17). The idea underpinning this procedure is as follows:

– Lemma 2.2 tells us that, if we can find a set of m-vectors $\mathcal{X}$ including all the Pareto-optimal points, then we can solve problem $1|r_j, \text{pmtn}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ by solving, for each $(X_1, \cdots, X_m) \in \mathcal{X}$, the constrained scheduling problem

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(k)} : f_{\max}^{(i)} \leq X_i \ \forall i \neq k,$$

where $k \in \{1, 2, \cdots, m\}$. For efficiency consideration, we naturally require that the size of $\mathcal{X}$ be bounded by a polynomial in n.

– To find such a set $\mathcal{X}$, upon studying the structure of the Pareto-optimal schedules, we find that there are only a polynomial number of patterns for the bottleneck jobs and their completion times.

– In a fixed pattern of the Pareto-optimal schedules, the bottleneck jobs $h_1, h_2, \cdots, h_m$ of the m agents and their completion times $C_{h_1}, C_{h_2}, \cdots, C_{h_m}$ are determined. Then we put the vector $(X_1, X_2, \cdots, X_m)$ into $\mathcal{X}$, where $X_i = f_{h_i}^{(i)}(C_{h_i})$ for $i = 1, 2, \cdots, m$.

– Although the Pareto-optimal schedules are unknown in advance, we find that the schedule-configurations in fact cover all the patterns of the Pareto-optimal schedules. We can show that, for each Pareto-optimal point $(X_1, X_2, \cdots, X_m)$, there must be a schedule-configuration $(\sigma, \mathbf{Q})$ such that the objective vector of $(\sigma, \mathbf{Q})$ is given by $\mathbf{X}(\sigma, \mathbf{Q}) = (X_1, X_2, \cdots, X_m)$.

– As a result, we can set $\mathcal{X}$ as the set of objective vectors of the schedule-configurations, i.e.,

$$\mathcal{X} = \{\mathbf{X}(\sigma, \mathbf{Q}) : (\sigma, \mathbf{Q}) \text{ is a schedule-configuration}\}.$$

– The set $\mathcal{X}$ constructed in the above way has a size polynomial in n since there are a total of $O(m!(n_1 n_2 \cdots n_m)^{m-1}) = O(n^{m^2-m})$ schedule-configurations, which is exactly what we are looking for.

– In summary, to solve the problem in (17), we enumerate all the schedule-configurations, generate the vector set $\mathcal{X} = \{\mathbf{X}(\sigma, \mathbf{Q}) : (\sigma, \mathbf{Q}) \text{ is a schedule-configuration}\}$, and finally pick the Pareto-optimal points in $\mathcal{X}$. This completes the discussion of the rationale for the proposed procedure.

Lemma 4.2. *Suppose that the EDD-like permutations $O_1, O_2, \cdots, O_m$ are given in advance. Then, for each schedule-configuration $(\sigma, \boldsymbol{Q})$, the objective vector $\boldsymbol{X}(\sigma, \boldsymbol{Q})$ of $(\sigma, \boldsymbol{Q})$ can be obtained in $O(n^2)$ time. For the case where $r_j = 0$ for all the jobs and $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ for all the agents, the time complexity reduces to $O(n \log n)$.*

Proof. Given a schedule-configuration $(\sigma, \mathbf{Q})$, we calculate some necessary items.

The principal vector $\mathbf{h}(\sigma, \mathbf{Q}) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$ can be obtained in a constant time since m is a fixed number and $h_{\sigma(t)} = \sigma(t) q_{t,t}$ for $t = 1, 2, \cdots, m-1$.

For each pair (s, t) with $s \in \{1, 2, \cdots, m\}$ and $t \in \{1, 2, \cdots, m-1\}$, the job set $\mathcal{J}_{q_{s,t}}^{O_{\sigma(s)}}$ can be obtained in $O(n)$ time since we have a total of n jobs and the permutation $O_{\sigma(s)}$ of agent $\sigma(s)$ is given in advance.

From the definition $\mathcal{J}_t^{(\sigma,\mathbf{Q})} = \mathcal{J}_{q_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{q_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{q_{m,t}}^{O_{\sigma(m)}}$ in (22), the $m-1$ sets $\mathcal{J}_1^{(\sigma,\mathbf{Q})}, \mathcal{J}_2^{(\sigma,\mathbf{Q})}, \cdots, \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}$ can be obtained in $O(mn) = O(n)$ time.

To calculate the $m-1$ values $C(\mathcal{J}_1^{(\sigma,\mathbf{Q})}), C(\mathcal{J}_2^{(\sigma,\mathbf{Q})}), \cdots, C(\mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})})$, we generate a permutation π of the n jobs in which the n jobs are listed in the order

$$\mathcal{J}_1^{(\sigma,\mathbf{Q})},\ \mathcal{J}_2^{(\sigma,\mathbf{Q})} \setminus \mathcal{J}_1^{(\sigma,\mathbf{Q})},\ \cdots,\ \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})} \setminus \mathcal{J}_{m-2}^{(\sigma,\mathbf{Q})},\ \mathcal{J} \setminus \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}.$$

Since $\mathcal{J}_1^{(\sigma,\mathbf{Q})} \subseteq \mathcal{J}_2^{(\sigma,\mathbf{Q})} \subseteq \cdots \subseteq \mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}$ as described in (23), we can generate π in $O(n)$ time. By running algorithm Pmtn-LS(π) in $O(n \log n)$ time, we obtain the $m-1$ values $C(\mathcal{J}_1^{(\sigma,\mathbf{Q})}), C(\mathcal{J}_2^{(\sigma,\mathbf{Q})}), \cdots, C(\mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})})$.

Given $\mathbf{h}(\sigma, \mathbf{Q}) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$ and $(C(\mathcal{J}_1^{(\sigma,\mathbf{Q})}), C(\mathcal{J}_2^{(\sigma,\mathbf{Q})}), \cdots, C(\mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})}))$, for each $t = 1, 2, \cdots, m-1$, the value $X_{\sigma(t)}(\sigma, \mathbf{Q}) = f_{h_{\sigma(t)}}^{(\sigma(t))}(C(\mathcal{J}_t^{(\sigma,\mathbf{Q})}))$ defined in (26) can be calculated in a constant time.

The total time complexity for the above calculations is $O(n \log n)$. But since $X_{\sigma(m)}(\sigma, \mathbf{Q})$ is the optimal value of the constrained scheduling problem in (27), from Theorem 3.1, the value $X_{\sigma(m)}(\sigma, \mathbf{Q})$ is available in $O(mn^2) = O(n^2)$ time. Consequently, the objective vector $\mathbf{X}(\sigma, \mathbf{Q})$ can be calculated in $O(n^2)$ time.

When $r_j = 0$ for all the jobs and $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ for all the agents, from Theorem 3.3, the value $X_{\sigma(m)}(\sigma, \mathbf{Q})$ can be determined in $O(mn + n \log n) = O(n \log n)$ time. Then $\mathbf{X}(\sigma, \mathbf{Q})$ can be calculated in $O(n \log n)$ time. □

The following lemma is critical for our discussion. To enhance the readability of the paper, we present the proof in the Appendix.

Lemma 4.3. *For each Pareto-optimal point $(X_1, X_2, \cdots, X_m)$ of the Pareto scheduling problem $1|r_j, pmtn, ND|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ in (17), there exists a schedule-configuration $(\sigma, \boldsymbol{Q}^*)$ such that the objective vector of $(\sigma, \boldsymbol{Q}^*)$ is given by $\boldsymbol{X}(\sigma, \boldsymbol{Q}^*) = (X_1, X_2, \cdots, X_m)$.*

Proof. See the Appendix. □

Remark: To help the reader understand Lemma 4.3, we consider the special problem $1|\text{CO}|(L_{\max}^{(1)}, L_{\max}^{(2)}, \cdots, L_{\max}^{(m)})$ and let $(X_1, X_2, \cdots, X_m)$ be a Pareto-optimal point of this problem. Same as the case where $m = 2$ studied in Agnetis et al. [1], there must be a Pareto-optimal schedule π corresponding to $(X_1, X_2, \cdots, X_m)$ such that, for each agent i, the n_i jobs of agent i are scheduled in the EDD order $O_i = (i1, i2, \cdots, in_i)$ in π, $i = 1, 2, \cdots, m$. This implies that, for each $j \in \{1, 2, \cdots, n\}$, the set of the first j jobs in π is of the form

$$\mathcal{J}_j^{\pi} = \{\pi(1), \pi(2), \cdots, \pi(j)\} = \mathcal{J}_{j_1}^{O_1} \cup \mathcal{J}_{j_2}^{O_2} \cup \cdots \cup \mathcal{J}_{j_m}^{O_m}, \tag{29}$$

where $j_i \in \{0, 1, \cdots, n_i\}$ for $i = 1, 2, \cdots, m$, such that $j_1 + j_2 + \cdots + j_m = j$.

Now let h_i be the bottleneck job of agent i under π, $i = 1, 2, \cdots, m$, and let $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(m))$ be a permutation of the m agents $\{1, 2, \cdots, m\}$ such that $C_{h_{\sigma(1)}}(\pi) < C_{h_{\sigma(2)}}(\pi) < \cdots < C_{h_{\sigma(m)}}(\pi)$. By setting

$$q_{s,t} = \max\{q : \mathcal{J}_q^{O_{\sigma(s)}} \subseteq \mathcal{J}_{h_{\sigma(t)}}^{\pi}\}, \quad s = 1, 2, \cdots, m, \ t = 1, 2, \cdots, m-1,$$

we obtain a schedule-configuration $(\sigma, \mathbf{Q}^*)$, where $\mathbf{Q}^* = (q_{s,t})_{m \times (m-1)}$. From (29), it is observed that

$$\mathcal{J}_{h_{\sigma(t)}}^{\pi} = \mathcal{J}_{q_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{q_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{q_{m,t}}^{O_{\sigma(m)}} = \mathcal{J}_t^{(\sigma, \mathbf{Q}^*)}, \quad t = 1, 2, \cdots, m-1.$$

Consequently, we have

$$C_{h_{\sigma(t)}}(\pi) = C(\mathcal{J}_t^{(\sigma, \mathbf{Q}^*)}), \quad t = 1, 2, \cdots, m-1,$$

so

$$X_{\sigma(t)} = L_{h_{\sigma(t)}}^{(\sigma(t))}(\pi) = X_{\sigma(t)}(\sigma, \mathbf{Q}^*), \quad t = 1, 2, \cdots, m-1. \tag{30}$$

Note that $X_{\sigma(m)}(\sigma, \mathbf{Q}^*)$ is the optimal value of problem

$$1|\text{CO}|L_{\max}^{(\sigma(m))} : L_{\max}^{(\sigma(t))} \leq X_{\sigma(t)}(\sigma, \mathbf{Q}^*) \ \forall t = 1, 2, \cdots, m-1$$

and $(X_1, X_2, \cdots, X_m)$ is a Pareto-optimal point. From Lemma 2.1 or Lemma 2.2, X_m is the optimal value of problem

$$1|\text{CO}|L_{\max}^{(\sigma(m))} : L_{\max}^{(\sigma(t))} \leq X_{\sigma(t)} \ \forall t = 1, 2, \cdots, m-1.$$

It follows from (30) that $X_{\sigma(m)} = X_{\sigma(m)}(\sigma, \mathbf{Q}^*)$. Consequently,

$$\mathbf{X}(\sigma, \mathbf{Q}^*) = (X_1(\sigma, \mathbf{Q}^*), X_2(\sigma, \mathbf{Q}^*), \cdots, X_m(\sigma, \mathbf{Q}^*)) = (X_1, X_2, \cdots, X_m).$$

This shows that Lemma 4.3 is valid for problem $1|\text{CO}|(L_{\max}^{(1)}, L_{\max}^{(2)}, \cdots, L_{\max}^{(m)})$.

From the above discussion, we have the following algorithm for solving the Pareto ND-agent scheduling problem in (17).

Algorithm 4.1. For the problem $1|r_j, \text{pmtn}, \text{ND}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ in (17).

Step 0. (*Preprocessing*) Generate the EDD-like permutations $O_1, O_2, \cdots, O_m$ in advance.

Step 1. Generate the set Γ that consists of all the schedule-configurations $(\sigma, \mathbf{Q})$.

Step 2. For each schedule-configuration $(\sigma, \mathbf{Q}) \in \Gamma$, calculate its objective vector $\mathbf{X}(\sigma, \mathbf{Q}) = (X_1(\sigma, \mathbf{Q}), X_2(\sigma, \mathbf{Q}), \cdots, X_m(\sigma, \mathbf{Q}))$. Set

$$\mathcal{X} := \{\mathbf{X}(\sigma, \mathbf{Q}) : (\sigma, \mathbf{Q}) \text{ is a schedule-configuration}\}$$

and $\mathcal{X}^* := \emptyset$. Here $\mathcal{X}^*$ can be interpreted as the set of Pareto-optimal points determined currently.

Step 3. Pick a vector $\mathbf{X} = (X_1, X_2, \cdots, X_m) \in \mathcal{X}$ and do the following:

(3.1) For k from 1 to m, do the following:

- Solve the constrained scheduling problem

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(k)} : f_{\max}^{(i)} \leq X_i \ \forall i \neq k$$

and let X_k^* be the optimal value. If the problem is infeasible, then we define $X_k^* = +\infty$.

(3.2) If $(X_1^*, X_2^*, \cdots, X_m^*) \neq (X_1, X_2, \cdots, X_m)$, then set $\mathcal{X} := \mathcal{X} \setminus \{\mathbf{X}\}$. Go to Step 4.

(3.3) If $(X_1^*, X_2^*, \cdots, X_m^*) = (X_1, X_2, \cdots, X_m)$, then let $\pi(\mathbf{X})$ be an arbitrary schedule obtained in Step (3.1). Set $\mathcal{X} := \mathcal{X} \setminus \{\mathbf{X}\}$ and $\mathcal{X}^* := \mathcal{X}^* \cup \{\mathbf{X}\}$. Go to Step 4.

Step 4. If $\mathcal{X} \neq \emptyset$, return to Step 3. If $\mathcal{X} = \emptyset$, then output $\mathcal{X}^*$ together with the schedules $\pi(\mathbf{X})$ for $\mathbf{X} \in \mathcal{X}^*$ and terminate the algorithm.

Theorem 4.1. *Suppose that all the m scheduling criteria of minimizing $f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}$ are lateness-like and the EDD-like permutations $O_1, O_2, \cdots, O_m$ are given in advance. Then Algorithm 4.1 solves the Pareto scheduling problem $1|r_j, pmtn, ND|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ in (17) in $O((n_1 n_2 \cdots n_m)^{m-1} n^2) = O(n^{m^2-m+2})$ time.*

Proof. From Lemma 4.3, each Pareto-optimal point must be the objective vector of some schedule-configuration. Thus, the vector set $\mathcal{X}$ generated in Step 2 includes all the Pareto-optimal points. From Lemma 2.2, we can run Algorithm 3.1 m times to determine whether or not a vector $\mathbf{X} = (X_1, X_2, \cdots, X_m)$ in $\mathcal{X}$ is Pareto-optimal. Such tasks are done in Step 3. Hence, Algorithm 4.1 solves the problem in (17).

To analyze the time complexity, recall that m is a fixed number. To generate a schedule-configuration $(\sigma, \mathbf{Q})$, we have $m!$ (a constant) choices for the permutation $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(m))$ of the m agents and at most

$$(n_1+1)^{m-1}(n_2+1)^{m-1}\cdots(n_m+1)^{m-1} = O((n_1n_2\cdots n_m)^{m-1})$$

choices for the $m \times (m-1)$-matrix $\mathbf{Q} = (q_{s,t})_{m\times(m-1)}$. Thus, we have $|\Gamma| = O((n_1n_2\cdots n_m)^{m-1})$ in Step 1. This further implies that $|\mathcal{X}| = O((n_1n_2\cdots n_m)^{m-1})$ in Step 2. From Lemma 4.2, each vector $\mathbf{X}(\sigma, \mathbf{Q}) \in \mathcal{X}$ can be obtained from $(\sigma, \mathbf{Q})$ in $O(n^2)$ time. Thus, Step 2 runs in $O((n_1n_2\cdots n_m)^{m-1}n^2)$ time. Step 3 has $|\mathcal{X}| = O((n_1n_2\cdots n_m)^{m-1})$ iterations. In each iteration, Step (3.1) solves the m constrained scheduling problems

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(k)} : f_{\max}^{(i)} \le X_i \ \forall i \ne k$$

for $k = 1, 2, \cdots, m$. From Theorem 3.1, each of the problems is solvable in $O(n^2)$ time. Thus, Step 3 runs in $O((n_1n_2\cdots n_m)^{m-1}n^2)$ time. It follows that Algorithm 4.1 runs in $O((n_1n_2\cdots n_m)^{m-1}n^2) = O(n^{m^2-m+2})$ time. □

If $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, then the EDD-like permutation O_i can be generated in $O(n \log n)$ time. From Theorem 4.1, we have the following result.

Theorem 4.2. *If $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ for all the agents $i \in \{1, 2, \cdots, m\}$, then Algorithm 4.1 solves the problem $1|r_j, pmtn, ND|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ in (17) in $O((n_1n_2\cdots n_m)^{m-1}n^2) = O(n^{m^2-m+2})$ time.*

If $r_j = 0$ for all the jobs and $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, by putting Theorem 3.4 in the time complexity analysis of Algorithm 4.1, we deduce the following result.

Theorem 4.3. *If $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, then problem*

$$1|ND|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$$

is solvable in $O((n_1n_2\cdots n_m)^{m-1}n\log n) = O(n^{m^2-m+1}\log n)$ time.

Proof. Note that Algorithm 4.1 certainly solves this subproblem correctly. Since $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$, the EDD-like permutations $O_1, O_2, \cdots, O_m$ can be generated in $O(mn \log n) = O(n \log n)$ time in Step 0.

From Lemma 4.2, Step 2 runs in $O((n_1n_2\cdots n_m)^{m-1}n\log n)$ time. From Theorem 3.4, Step 3 runs in $O((n_1n_2\cdots n_m)^{m-1}n\log n)$ time. Consequently, Algorithm 4.1 solves the problem in $O((n_1n_2\cdots n_m)^{m-1}n\log n) = O(n^{m^2-m+1}\log n)$ time. □

4.2 The CO-agent version

We now consider the Pareto CO-agent scheduling problem

$$1|r_j, \text{pmtn, CO}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}), \tag{31}$$

in which each function $f_{\max}^{(i)}$ is lateness-like. Note that $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \cdots, \mathcal{J}^{(m)}$ form a partition of $\mathcal{J} = \{1, 2, \cdots, n\}$ with $|\mathcal{J}^{(i)}| = n_i$. Then $n_1 + n_2 + \cdots + n_m = n$.

From the discussion in Section 4.1, the problem in (31) is solvable in $O((n_1 n_2 \cdots n_m)^{m-1} n^2)$ time. We improve the time complexity by a small trick. To this end, we re-consider the constrained CO-agent scheduling problem

$$1|r_j, \text{pmtn, CO}|f_{\max}^{(m)} : f_{\max}^{(i)} \leq X_i \ \forall i = 1, 2, \cdots, m-1, \tag{32}$$

where each function $f_{\max}^{(i)}$ is lateness-like. Recall that, for each agent i, the EDD-like permutation $O_i = (i1, i2, \cdots, in_i)$ is given in advance. Similar to Lemma 3.1, we have the following lemma to determine the last job in an optimal permutation.

Lemma 4.4. *Suppose that the problem in (32) is feasible. Then there is an optimal permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$ for the problem such that $\pi(n) = x$, where*

- *$x = in_i$ if there is some $i \in \{1, 2, \cdots, m-1\}$ such that $f_{i,n_i}^{(i)}(C(\mathcal{J})) \leq X_i$, and*
- *$x = mn_m$ otherwise.*

Proof. Since the constrained CO-agent scheduling problem in (32) is a special version of the constrained ND-agent scheduling problem in (11), the result follows directly from Lemma 3.1. □

Recalling that for $1 \leq i \leq m$ and $1 \leq q \leq n_i$, we have $\mathcal{J}_q^{O_i} = \{i1, i2, \cdots, iq\}$. We now introduce some new notation. Given an m-vector $(x_1, x_2, \cdots, x_m)$ with $x_i \in \{0, 1, \cdots, n_i\}$ for each $i \in \{1, 2, \cdots, m\}$, we define

$$\mathcal{J}(x_1, x_2, \cdots, x_m) = \mathcal{J}_{x_1}^{O_1} \cup \mathcal{J}_{x_2}^{O_2} \cup \cdots \cup \mathcal{J}_{x_m}^{O_m} \tag{33}$$

and

$$C(x_1, x_2, \cdots, x_m) = C(\mathcal{J}_{x_1}^{O_1} \cup \mathcal{J}_{x_2}^{O_2} \cup \cdots \cup \mathcal{J}_{x_m}^{O_m}). \tag{34}$$

Then $C(0, 0, \cdots, 0) = 0$ and $C(n_1, n_2, \cdots, n_m) = C(\mathcal{J})$.

Each value $C(x_1, x_2, \cdots, x_m)$ can be determined in $O(n \log n)$ time, as stated in Section 2.2. Thus, all the values $C(x_1, x_2, \cdots, x_m)$ can be determined in $O(n_1 n_2 \cdots n_m n \log n)$ time.

Remark: With the help of an ERD order $\vec{\mathcal{J}}$ of $\mathcal{J}$ as in Algorithm 3.1, the time complexity for calculating all the values $C(x_1, x_2, \cdots, x_m)$ can be reduced to $O(n_1 n_2 \cdots n_m n)$. But the $O(n_1 n_2 \cdots n_m n \log n)$ time complexity is sufficient for our subsequent discussion.

Now we use the following algorithm to solve the problem in (32).

Algorithm 4.2. For the problem in (32).

Step 1. Set $(x_1, x_2, \cdots, x_m) := (n_1, n_2, \cdots, n_m)$, $\tau := C(x_1, x_2, \cdots, x_m)$, and $k := n$.

Step 2. Check the condition

$$f_{ix_i}^{(i)}(\tau) \leq X_i \text{ for all } i = 1, 2, \cdots, m-1$$

to find a job ix_i legal at time τ.

• If some job ix_i with $x_i \geq 1$ and $1 \leq i \leq m-1$ is legal at time τ, then set $\pi(k) := ix_i$. Go to Step 3.

• Otherwise, do the following:

– If $x_m \geq 1$, then set $\pi(k) := mx_m$. Go to Step 3.

– If $x_m = 0$, then terminate the algorithm and output *infeasibility*.

Step 3. Suppose that $\pi(k) := ix_i$ for some $i \in \{1, 2, \cdots, m\}$. Do the following:

• If $k \geq 2$, then set $\tau := C(x_1, \cdots, x_{i-1}, x_i - 1, x_{i+1}, \cdots, x_m)$, $x_i := x_i - 1$, and $k := k - 1$, and go to Step 2.

• If $k = 1$, then go to Step 4.

Step 4. Output the permutation $\pi = (\pi(1), \pi(2), \cdots, \pi(n))$, which is optimal for the problem under consideration.

Lemma 4.5. *Given all the permutations $O_1, O_2, \cdots, O_m$ and all the values $C(x_1, x_2, \cdots, x_m)$ in advance, Algorithm 4.2 solves the problem in (32) with lateness-like criteria in $O(mn)$ time.*

Proof. The correctness of Algorithm 4.2 is implied in Lemma 4.4. To analyze the time complexity, we notice that Algorithm 4.2 has n iterations. In each iteration, Step 2 runs in $O(m)$ time and Step 3 runs in a constant time (since all the values $C(x_1, x_2, \cdots, x_m)$ are given in advance). Consequently, Algorithm 4.2 runs in $O(mn)$ time. □

Lemma 4.6. *Given all the permutations $O_1, O_2, \cdots, O_m$ and all the values $C(x_1, x_2, \cdots, x_m)$ in advance, for each schedule-configuration $(\sigma, \boldsymbol{Q})$, the objective vector $\boldsymbol{X}(\sigma, \boldsymbol{Q})$ of $(\sigma, \boldsymbol{Q})$ can be obtained in $O(n)$ time.*

Proof. Similar to the proof of Lemma 4.2, for a schedule-configuration $(\sigma, \mathbf{Q})$, the principal vector $\mathbf{h}(\sigma, \mathbf{Q}) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$ can be obtained in a constant time. But in the current status, for $t = 1, 2, \cdots, m-1$, we have

$$\mathcal{J}_t^{(\sigma,\mathbf{Q})} = \mathcal{J}_{q_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{q_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{q_{m,t}}^{O_{\sigma(m)}} = \mathcal{J}(x_1(t), x_2(t), \cdots, x_m(t)),$$

where $x_i(t) = q_{\sigma^{-1}(i),t}$ for $i = 1, 2, \cdots, m$. This implies that the $m-1$ values $C(\mathcal{J}_1^{(\sigma,\mathbf{Q})}), C(\mathcal{J}_2^{(\sigma,\mathbf{Q})}), \cdots, C(\mathcal{J}_{m-1}^{(\sigma,\mathbf{Q})})$ have been given in advance. Consequently, for $t = 1, 2, \cdots, m-1$, the value $X_{\sigma(t)}(\sigma, \mathbf{Q}) = f_{h_{\sigma(t)}}^{(\sigma(t))}(C(\mathcal{J}_t^{(\sigma,\mathbf{Q})}))$ defined in (26) can be calculated in a constant time.

Finally, we note that $X_{\sigma(m)}(\sigma, \mathbf{Q})$ is the optimal value of the constrained scheduling problem

$$1|r_j, \text{pmtn}, \text{CO}|f_{\max}^{(\sigma(m))} : f_{\max}^{(\sigma(t))} \leq X_{\sigma(t)}(\sigma, \mathbf{Q}) \ \forall t = 1, 2, \cdots, m-1.$$

From Lemma 4.5, the value $X_{\sigma(m)}(\sigma, \mathbf{Q})$ can be obtained in $O(mn) = O(n)$ time. Consequently, the objective vector $\mathbf{X}(\sigma, \mathbf{Q})$ can be calculated in $O(n)$ time. □

Algorithm 4.3. For the problem $1|r_j, \text{pmtn}, \text{CO}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ in (31).

Step 0. (*Preprocessing*) Calculate all the values $C(x_1, x_2, \cdots, x_m)$ for $0 \leq x_i \leq n_i$ $\forall i = 1, 2, \cdots, m$.

Step 1. Generate the set Γ that consists of all the schedule-configurations $(\sigma, \mathbf{Q})$.

Step 2. For each schedule-configuration $(\sigma, \mathbf{Q}) \in \Gamma$, calculate its objective vector $\mathbf{X}(\sigma, \mathbf{Q}) = (X_1(\sigma, \mathbf{Q}), X_2(\sigma, \mathbf{Q}), \cdots, X_m(\sigma, \mathbf{Q}))$. Set

$$\mathcal{X} := \{\mathbf{X}(\sigma, \mathbf{Q}) : (\sigma, \mathbf{Q}) \text{ is a schedule-configuration}\}$$

and $\mathcal{X}^* := \emptyset$. Here $\mathcal{X}^*$ can be interpreted as the set of Pareto-optimal points determined currently.

Step 3. Pick a vector $\mathbf{X} = (X_1, X_2, \cdots, X_m) \in \mathcal{X}$ and do the following:

(3.1) For k from 1 to m, do the following:

- Solve the constrained scheduling problem

$$1|r_j, \text{pmtn}, \text{CO}|f_{\max}^{(k)} : f_{\max}^{(i)} \leq X_i \ \forall i \neq k$$

and let X_k^* be the optimal value. If the problem is infeasible, then we define $X_k^* = +\infty$.

(3.2) If $(X_1^*, X_2^*, \cdots, X_m^*) \neq (X_1, X_2, \cdots, X_m)$, then set $\mathcal{X} := \mathcal{X} \setminus \{\mathbf{X}\}$. Go to Step 4.

(3.3) If $(X_1^*, X_2^*, \cdots, X_m^*) = (X_1, X_2, \cdots, X_m)$, then let $\pi(\mathbf{X})$ be an arbitrary schedule obtained in Step (3.1). Set $\mathcal{X} := \mathcal{X} \setminus \{\mathbf{X}\}$ and $\mathcal{X}^* := \mathcal{X}^* \cup \{\mathbf{X}\}$. Go to Step 4.

Step 4. If $\mathcal{X} \neq \emptyset$, return to Step 3. If $\mathcal{X} = \emptyset$, then output $\mathcal{X}^*$ together with the schedules $\pi(\mathbf{X})$ for $\mathbf{X} \in \mathcal{X}^*$ and terminate the algorithm.

From the discussion in Section 4.1, it is obvious that Algorithm 4.3 solves the problem in (31) correctly. As for the time complexity, we notice the following facts:

– Step 0 runs in $O(n_1 n_2 \cdots n_m n \log n)$ time. This follows from the fact that each value $C(\mathcal{J}')$ with $\mathcal{J}' \subseteq \mathcal{J}$ can be obtained in $O(n \log n)$ time, as stated in Section 2.2.

– Step 1 runs in $O((n_1 n_2 \cdots n_m)^{m-1})$ time. This follows from the fact that the number of schedule-configurations, i.e. $|\Gamma|$, is upper bounded by $O((n_1 n_2 \cdots n_m n)^{m-1})$.

– Step 2 runs in $O((n_1 n_2 \cdots n_m)^{m-1} n)$ time. This follows from Lemma 4.6.

– Step 3 runs in $O((n_1 n_2 \cdots n_m)^{m-1} n)$ time. This follows from Lemma 4.5.

The above discussion leads to the following result.

Theorem 4.4. *Suppose that the objective functions* $f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)}$ *are lateness-like and the EDD-like permutations* $O_1, O_2, \cdots, O_m$ *are given in advance. Then Algorithm 4.3 solves problem* $1|r_j, pmtn, CO|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ *in* $O((n_1 n_2 \cdots n_m)^{m-1} n)$ *time.*

As a consequence of Theorem 4.4, we have the following result.

Theorem 4.5. *If* $f_{\max}^{(i)} \in \{C_{\max}^{(i)}, F_{\max}^{(i)}, L_{\max}^{(i)}, T_{\max}^{(i)}, WC_{\max}^{(i)}\}$ *for all* $i = 1, 2, \cdots, m$, *then problem* $1|r_j, pmtn, CO|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ *is solvable in* $O((n_1 n_2 \cdots n_m)^{m-1} n)$ *time.*

When $m = 2$, Theorem 4.5 implies that problem $1|r_j, \text{pmtn}, \text{CO}|(L_{\max}^{(1)}, L_{\max}^{(2)})$ is solvable in $O(n_1 n_2 n)$ time. This is an improvement over the time complexity $O(n_1 n_2 n \log n)$ established in Yuan et al. [31].

5 NP-hardness when m is arbitrary

In this section we assume that m, the number of agents, is arbitrary. Hoogeveen [14] showed that problem $1|\text{ID}|f_{\max}^{(1)} + f_{\max}^{(2)} + \cdots + f_{\max}^{(m)}$ is strongly NP-hard. We show that the problem remains strongly NP-hard even when $p_j = 1$ for all the jobs.

Recently Agnetis [1] showed that problem $1|\text{ND}, p_j = 1|\sum_{i=1}^{m} \lambda_i C_{\max}^{(i)}$ is strongly NP-hard. In their proof, they defined that $\lambda_i = 1$ for all the agent i. Then problem $1|\text{ND}, p_j = 1|\sum_{i=1}^{m} C_{\max}^{(i)}$ is also strongly NP-hard. This result can be used to show the following results related to our research.

Theorem 5.1. *The three scheduling problems* $1|ID, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$, $1|ID, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$, *and* $1|ID, p_j = 1|\sum_{i=1}^{m} T_{\max}^{(i)}$ *are strongly* NP*-hard.*

Proof. The reductions used in our proof are very easy, so we only present an informal description.

We first consider problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$. Beginning with an instance I of problem $1|\text{ND}, p_j = 1|\sum_{i=1}^{m} C_{\max}^{(i)}$, we obtain an instance I' of problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$ by replacing each agent i with an agent i, $1 \le i \le m$, and setting $w_j^{(i)} = 1$ if $j \in \mathcal{J}^{(i)}$ and $w_j^{(i)} = 0$ otherwise for $1 \le j \le n$ and $1 \le i \le m$. It can be observed that the reduction is done in polynomial time under unary encoding and, under every schedule of the n jobs, the objective value of instance I' is the same as that of instance I. Consequently, problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$ is strongly NP-hard.

We next consider problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$. Beginning with an instance I of problem $1|\text{ND}, p_j = 1|\sum_{i=1}^{m} C_{\max}^{(i)}$, we obtain an instance I'' of problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$ by replacing each agent i with an agent i, $1 \le i \le m$, and setting $d_j^{(i)} = 0$ if $j \in \mathcal{J}^{(i)}$ and $d_j^{(i)} = n$ otherwise for $1 \le j \le n$ and $1 \le i \le m$. It can be observed that the reduction is done in polynomial time under unary encoding and, under every

schedule of the n jobs, the objective value of instance I'' is the same as that of instance I. Consequently, problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$ is strongly NP-hard.

The above discussion for problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$ is still valid for problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} T_{\max}^{(i)}$. Then problem $1|\text{ID}, p_j = 1|\sum_{i=1}^{m} T_{\max}^{(i)}$ is also strongly NP-hard. This completes the proof. □

Yuan [28] showed that problem $1|\text{CO}|\sum_{i=1}^{m} \lambda_i L_{\max}^{(i)}$ is strongly NP-hard by a reduction from the strongly NP-hard problem $1||\sum w_j T_j$ proved in Lawler [18]. The reduction in Yuan [28] also implies that problem $1|\text{CO}|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ is strongly NP-hard. By modifying the reduction in Yuan [28], we present the following stronger complexity result.

Theorem 5.2. *Problem* $1|CO, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ *is strongly* NP*-hard.*

Proof. Suppose that we are given an instance I of problem $1||\sum w_j T_j$, in which there are n jobs $\{1, 2, \cdots, n\}$ with each job j having a processing time $p_j > 0$, a due date $d_j \geq 0$, and a weight $w_j > 0$, where all the values p_j, d_j, and w_j are integers. We construct an instance I' of problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ as follows:

• We have $m = n$ competing agents $1, 2, \cdots, n$. Each agent $i \in \{1, 2, \cdots, n\}$ has p_i jobs $i1, i2, \cdots, ip_i$ with a common processing time 1 and a common due date d_i. Moreover, the weight of agent i is given by $\lambda_i = w_i$. Then we have a total of $p_1 + p_2 + \cdots + p_n$ jobs in I'.

Essentially, I' is obtained from I by regarding each job i (of processing time p_i and due date d_i) in I as p_i jobs (of processing time 1 and due date d_i) of agent i in I' and taking w_i as the weight of agent i in I'.

Under unary encoding, the sizes of both I and I' are given by

$$O(n + \sum_{i=1}^{n} p_i + \sum_{i=1}^{n} d_i + \sum_{i=1}^{n} w_i).$$

Thus, the above construction can be done in polynomial time under unary encoding.

In every schedule π (without artificial idle times) of instance I', the maximum tardiness $T_{\max}^{(i)}(\pi)$ of agent i is determined by the completion time of the last job of agent i in π. Then we may regard the processing of the p_i jobs of agent i in π as the processing of job i with preemption in π with $T_i(\pi) = T_{\max}^{(i)}(\pi)$. This means that problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ on instance I' is equivalent to problem $1|\text{pmtn}|\sum w_j T_j$ on instance I. Since all the jobs are released at time 0, $1|\text{pmtn}|\sum w_j T_j$ is equivalent to $1||\sum w_j T_j$. Therefore, problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ on instance I' is equivalent to problem $1||\sum w_j T_j$ on instance I. From the strong NP-hardness of problem $1||\sum w_j T_j$ established in Lawler [18], we conclude that problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i T_{\max}^{(i)}$ is strongly NP-hard. This completes the proof. □

Cheng et al. [9] showed that problem $1|\text{CO}|\sum_{i=1}^{m} WC_{\max}^{(i)}$ is strongly NP-hard. Similar to the proof of Theorem 5.2, by using problem $1|\text{CO}|\sum_{i=1}^{m} WC_{\max}^{(i)}$ for the reduction and

replacing each job j with processing p_j by p_j unit-length jobs belonging to the same agent as j, we deduce the following result.

Theorem 5.3. *Problem* $1|CO, p_j = 1|\sum_{i=1}^{m} WC_{\max}^{(i)}$ *is strongly* NP*-hard.*

6 Conclusions

In this paper we study multi-agent scheduling with release dates and preemption on a single machine, where the scheduling objective function of each agent to be minimized is regular and of the max-form and the multi-agent aspect is ND-agent, ID-agent, or CO-agent. We show that the constrained multi-agent scheduling problems are solvable in polynomial time, and the weighted-sum and Pareto multi-agent scheduling problems are solvable in polynomial time if the number of agents is a fixed number and the scheduling criteria are lateness-like. We also show that the weighted-sum and Pareto multi-agent scheduling problems are strongly NP-hard even when $r_j = 0$ and $p_j = 1$ for all the jobs.

Comparing with the known results in the literature, our research advances multi-criteria scheduling research in the following aspects: (i) Our research on problem $1|r_j, \text{pmtn}, \text{ND}|(f_{\max}^{(1)}, f_{\max}^{(2)}, \cdots, f_{\max}^{(m)})$ with lateness-like criteria is innovative. The technique we develop to address this problem goes beyond the methodology presented in the literature. (ii) From Tables 1 and 2, for the CSP in the "r_j, pmtn" environment, the time complexity of our algorithms matches the corresponding time complexity of the algorithms for solving their special versions with $r_j = 0$ in the literature. (iii) From Tables 1 and 2, for the WSP and PSP in the "r_j, pmtn" environment, the time complexity of our algorithms improves the existing results in the literature. (iv) Our NP-hardness results for CO-agent scheduling are the first such results for the special version with $p_j = 1$ for all the jobs. Future research should address the following questions.

Cheng et al. [9] showed that problem $1|\text{CO}|\sum_{i=1}^{m} L_{\max}^{(i)}$ is at least NP-hard in the ordinary sense and the computational complexity status of the problem (whether it is strongly NP-hard or solvable in pseudo-polynomial time) is still open. Moreover, to the best of our knowledge, the computational complexity of the following related problems is still open:

- Problem $1|\text{CO}|\sum_{i=1}^{m} f_{\max}^{(i)}$, where m is fixed.
- Problem $1|\text{ID}|\sum_{i=1}^{m} f_{\max}^{(i)}$, where m is fixed.
- Problem $1|\text{ND}|\sum_{i=1}^{m} f_{\max}^{(i)}$, where m is fixed.
- Problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} \lambda_i L_{\max}^{(i)}$.
- Problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} L_{\max}^{(i)}$.
- Problem $1|\text{CO}, p_j = 1|\sum_{i=1}^{m} T_{\max}^{(i)}$.

Acknowledgements

We thank an associate editor and five anonymous referees for their constructive comments and suggestions on earlier versions of our paper. Yuan was supported in part by NSFC under grant numbers 11671368 and 11771406. Cheng was supported in part by The Hong Kong Polytechnic University under the Fung Yiu King - Wing Hang Bank Endowed Professorship in Business Administration.

Appendix: Proof of Lemma 4.3

Proof. Suppose that π is a Pareto-optimal schedule corresponding to $(X_1, X_2, \cdots, X_m)$. From Lemma 2.5, we can also regard π as a completion-coinciding permutation of $\mathcal{J} = \{1, 2, \cdots, n\}$. Then $\mathcal{S}^{\pi}_{\pi(k)} = \mathcal{J}^{\pi}_{k}$ for $k = 1, 2, \cdots, n$ and $C_{\pi(1)}(\pi) < C_{\pi(2)}(\pi) < \cdots < C_{\pi(n)}(\pi)$. Let h_i be the bottleneck job of agent i under π, $i = 1, 2, \cdots, m$ and let $\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(m))$ be a permutation of the m agents $\{1, 2, \cdots, m\}$ such that

$$C_{h_{\sigma(1)}}(\pi) \leq C_{h_{\sigma(2)}}(\pi) \leq \cdots \leq C_{h_{\sigma(m)}}(\pi). \tag{35}$$

Note that some equalities in (35) may hold since different agents may share a common bottleneck job.

For our purpose, we choose the Pareto-optimal schedule π such that the following two conditions are satisfied:

(C1) $\sum_{s=1}^{m} C_{h_s}(\pi)$ is as small as possible, and

(C2) subject to condition (C1), $\sum_{s=1}^{m} h_s^{(s)}$ is as large as possible.

Recall that $j^{(s)}$ is the index of job j in O_s, i.e., $j = sj^{(s)}$. We define two matrices $\mathbf{Q} = (q_{s,t})_{m\times m}$ and $\tilde{\mathbf{Q}} = (\tilde{q}_{s,t})_{m\times m}$ in the following way.

Definition of $\mathbf{Q} = (q_{s,t})_{m\times m}$: For $1 \leq s \leq m$ and $1 \leq t \leq m$, we set $q_{s,t}$ as the maximum index in $\{0, 1, \cdots, n_{\sigma(s)}\}$ such that all the jobs in $\mathcal{J}^{O_{\sigma(s)}}_{q_{s,t}} = \{\sigma(s)1, \sigma(s)2, \cdots, \sigma(s)q_{s,t}\}$ are completed by time $C_{h_{\sigma(t)}}(\pi)$ in π, i.e.,

$$q_{s,t} = \max\{q : \mathcal{J}^{O_{\sigma(s)}}_{q} \subseteq \mathcal{S}^{\pi}_{h_{\sigma(t)}}\}. \tag{36}$$

It is observed that

$$0 \leq q_{s,1} \leq q_{s,2} \leq \cdots \leq q_{s,m} \leq n_{\sigma(s)} \text{ for each } s \text{ with } 1 \leq s \leq m. \tag{37}$$

From the choices of these $q_{s,t}$ in (36), we further have, for $s, t \in \{1, 2, \cdots, m\}$,

$$\mathcal{J}^{O_{\sigma(s)}}_{q_{s,t}} \subseteq \mathcal{S}^{\pi}_{h_{\sigma(t)}} \tag{38}$$

and

$$\sigma(s), q_{s,t} + 1 \notin \mathcal{S}^{\pi}_{h_{\sigma(t)}}. \tag{39}$$

Definition of $\tilde{\mathbf{Q}} = (\tilde{q}_{s,t})_{m\times m}$: The matrix $\tilde{\mathbf{Q}} = (\tilde{q}_{s,t})_{m\times m}$ is obtained from $\mathbf{Q}$ by setting

$$\tilde{q}_{s,t} = \begin{cases} \min\{q_{s,t}, h_{\sigma(s)}^{(\sigma(s))}\}, & \text{if } t \le s, \\ q_{s,t}, & \text{if } t > s. \end{cases} \tag{40}$$

From (37) and (40), it is clear that

$$0 \le \tilde{q}_{s,1} \le \tilde{q}_{s,2} \le \cdots \le \tilde{q}_{s,m} \le n_{\sigma(s)} \text{ for each } s \text{ with } 1 \le s \le m. \tag{41}$$

Since $h_{\sigma(t)}$ is the bottleneck job of agent $\sigma(t)$ and $h_{\sigma(t)} \in \mathcal{S}^{\pi}_{h_{\sigma(t)}}$, from Lemma 4.1 and the choice of $q_{s,t}$, we have

$$q_{t,t} \ge h_{\sigma(t)}^{(\sigma(t))} \ge 1 \text{ for } t = 1, 2, \cdots, m. \tag{42}$$

From (40), the expressions in (42) also imply that

$$\tilde{q}_{t,t} = h_{\sigma(t)}^{(\sigma(t))} \ge 1 \text{ for } t = 1, 2, \cdots, m. \tag{43}$$

Let $\mathbf{Q}^* = (\tilde{q}_{s,t})_{m\times(m-1)}$ be the $m \times (m-1)$-matrix obtained from $\tilde{\mathbf{Q}}$ by deleting the last column, i.e.,

$$\mathbf{Q}^* = \begin{pmatrix} \tilde{q}_{1,1} & \tilde{q}_{1,2} & \cdots & \cdots & \tilde{q}_{1,m-1} \\ \tilde{q}_{2,1} & \tilde{q}_{2,2} & \cdots & \cdots & \tilde{q}_{2,m-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \tilde{q}_{m,1} & \tilde{q}_{m,2} & \cdots & \cdots & \tilde{q}_{m,m-1} \end{pmatrix}.$$

From (41) and (43), $\mathbf{Q}^*$ satisfies the two conditions in (18) and (19). Therefore, we have the following claim.

Claim 1. $(\sigma, \mathbf{Q}^*)$ is a schedule-configuration.

Now we set $\mathbf{h}(\sigma, \mathbf{Q}^*) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$. Since $(\sigma, \mathbf{Q}^*)$ is a schedule-configuration, the relations in (43) further imply the following claim.

Claim 2. $\mathbf{h}(\sigma, \mathbf{Q}^*) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$ is the principal vector induced by the schedule-configuration $(\sigma, \mathbf{Q}^*)$.

We borrow the notation in (22) and set $\mathcal{J}_t^{(\sigma,\mathbf{Q})} = \mathcal{J}_{q_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{q_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{q_{m,t}}^{O_{\sigma(m)}}$ and $\mathcal{J}_t^{(\sigma,\tilde{\mathbf{Q}})} = \mathcal{J}_{\tilde{q}_{1,t}}^{O_{\sigma(1)}} \cup \mathcal{J}_{\tilde{q}_{2,t}}^{O_{\sigma(2)}} \cup \cdots \cup \mathcal{J}_{\tilde{q}_{m,t}}^{O_{\sigma(m)}}$ for $t = 1, 2, \cdots, m$. Note that $t = m$ is allowed here. From the definition of $\mathbf{Q}^*$, we have

$$\mathcal{J}_t^{(\sigma,\mathbf{Q}^*)} = \mathcal{J}_t^{(\sigma,\tilde{\mathbf{Q}})} \text{ for } t = 1, 2, \cdots, m-1. \tag{44}$$

Since $\tilde{q}_{s,t} \le q_{s,t}$ for all $s, t \in \{1, 2, \cdots, m\}$, from (38), we further have

$$\mathcal{J}_t^{(\sigma,\tilde{\mathbf{Q}})} \subseteq \mathcal{J}_t^{(\sigma,\mathbf{Q})} \subseteq \mathcal{S}^{\pi}_{h_{\sigma(t)}} \text{ for all } t = 1, 2, \cdots, m.$$

We next prove the following long and critical claim.

Claim 3. For each $t \in \{1, 2, \cdots, m\}$, every job (if any) in $\mathcal{S}^{\pi}_{h_{\sigma(t)}} \setminus \mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_t$ is not interfering with respect to $\mathcal{S}^{\pi}_{h_{\sigma(t)}}$.

Suppose to the contrary that the result in Claim 3 is violated. Let t be the minimum index in $\{1, 2, \cdots, m\}$ such that there is some interfering job $x \in \mathcal{S}^{\pi}_{h_{\sigma(t)}} \setminus \mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_t$ with respect to $\mathcal{S}^{\pi}_{h_{\sigma(t)}}$. Then we have

$$C_{h_{\sigma(t)}}(\pi) = C(\mathcal{S}^{\pi}_{h_{\sigma(t)}}) > C(\mathcal{S}^{\pi}_{h_{\sigma(t)}} \setminus \{x\}) \geq C(\mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_t). \tag{45}$$

The minimality of t implies that there are no interfering jobs in $\mathcal{S}^{\pi}_{h_{\sigma(s)}} \setminus \mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_s$ with respect to $\mathcal{S}^{\pi}_{h_{\sigma(s)}}$ for $s = 1, 2, \cdots, t-1$. From Lemma 2.6′, we have

$$C_{h_{\sigma(s)}}(\pi) = C(\mathcal{S}^{\pi}_{h_{\sigma(s)}}) = C(\mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_s) \text{ for } s = 1, 2, \cdots, t-1. \tag{46}$$

From (45) and (46), we further have

$$h_{\sigma(t)} \notin \{h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(t-1)}\}. \tag{47}$$

We may assume that $x = \sigma(s)x_s$ for every $s \in \{1, 2, \cdots, m\}$ with $x \in \mathcal{J}^{(\sigma(s))}$, i.e., x is the x_s-th job of agent $\sigma(s)$ if $x \in \mathcal{J}^{(\sigma(s))}$. Note that the relation $x \notin \mathcal{J}^{(\sigma,\tilde{\mathbf{Q}})}_t$ can be equivalently written as

$$x = \sigma(s)x_s \notin \mathcal{J}^{O_{\sigma(s)}}_{\tilde{q}_{s,t}} \text{ for } s \in \{1, 2, \cdots, m\} \text{ with } x \in \mathcal{J}^{(\sigma(s))}. \tag{48}$$

Then we have

$$x_s \geq \tilde{q}_{s,t} + 1 \text{ for } s \in \{1, 2, \cdots, m\} \text{ with } x \in \mathcal{J}^{(\sigma(s))}. \tag{49}$$

The inequalities in (49) can be further enhanced. To this end, we define l as the maximum index in $\{t, t+1, \cdots, m\}$ such that $h_{\sigma(t)} = h_{\sigma(l)}$. From (35), we have

$$h_{\sigma(t)} = h_{\sigma(t+1)} = \cdots = h_{\sigma(l)}. \tag{50}$$

From (47) and the relations $C_{h_{\sigma(1)}}(\pi) \leq C_{h_{\sigma(2)}}(\pi) \leq \cdots \leq C_{h_{\sigma(m)}}(\pi)$ in (35), we have $C_{h_{\sigma(t-1)}}(\pi) < C_{h_{\sigma(t)}}(\pi)$. The choice of l further implies that $C_{h_{\sigma(l+1)}}(\pi) > C_{h_{\sigma(t)}}(\pi)$. Then we have

$$C_{h_{\sigma(1)}}(\pi) \leq \cdots \leq C_{h_{\sigma(t-1)}}(\pi) < C_{h_{\sigma(t)}}(\pi) < C_{h_{\sigma(l+1)}}(\pi) \leq \cdots \leq C_{h_{\sigma(m)}}(\pi), \tag{51}$$

where $C_{h_{\sigma(0)}}(\pi) = 0$ and $C_{h_{\sigma(m+1)}}(\pi) = +\infty$.

If $s \in \{1, 2, \cdots, t-1\}$, then the definition of $\tilde{q}_{s,t}$ in (40) implies that $\tilde{q}_{s,t} = q_{s,t}$. From (48), we have $x \notin \mathcal{J}^{O_{\sigma(s)}}_{q_{s,t}}$. Then the definition of $q_{s,t}$ in (36) implies that $x_s \geq q_{s,t} + 2$.

If $s \in \{l+1, l+2, \cdots, m\}$, from (51), we have $C_{h_{\sigma(s)}}(\pi) > C_{h_{\sigma(t)}}(\pi)$. This implies that $h_{\sigma(s)} \notin \mathcal{S}^{\pi}_{h_{\sigma(t)}}$, so $q_{s,t} < h^{(\sigma(s))}_{\sigma(s)}$. From the definition of $\tilde{q}_{s,t}$ in (40), we have $\tilde{q}_{s,t} = \min\{q_{s,t}, h^{(\sigma(s))}_{\sigma(s)}\} = q_{s,t}$. Again, we have $x_s \geq q_{s,t} + 2$.

If $s \in \{t, t+1, \cdots, l\}$, then $h_{\sigma(t)} = h_{\sigma(s)}$ is the bottleneck job of agent $\sigma(s)$ under π. From (42), we have $h_{\sigma(s)}^{(\sigma(s))} \leq q_{s,s} = q_{s,t}$, where the equality follows from the fact that $h_{\sigma(t)} = h_{\sigma(s)}$. Then $\tilde{q}_{s,t} = \min\{q_{s,t}, h_{\sigma(s)}^{(\sigma(s))}\} = h_{\sigma(s)}^{(\sigma(s))}$.

From the above discussion, the inequalities in (49) can be enhanced as follows: For $s \in \{1, 2, \cdots, m\}$ with $x \in \mathcal{J}^{(\sigma(s))}$, we have

$$\begin{cases} x_s \geq q_{s,t} + 2, & \text{if } s \notin \{t, t+1, \cdots, l\}, \\ x_s \geq h_{\sigma(s)}^{(\sigma(s))} + 1, & \text{if } s \in \{t, t+1, \cdots, l\}. \end{cases} \tag{52}$$

Let π^* be the new permutation of the n jobs in $\mathcal{J}$ obtained from permutation π by shifting x to the position just after $h_{\sigma(t)}$. Since x is an interfering job with respect to $\mathcal{S}_{h_{\sigma(t)}}^{\pi}$, from Lemma 2.7, we have

$$\begin{cases} C_{h_{\sigma(t)}}(\pi^*) < C_x(\pi^*) = C_{h_{\sigma(t)}}(\pi), & \\ C_j(\pi^*) \leq C_j(\pi), & \text{if } j \neq x, \\ C_j(\pi^*) = C_j(\pi), & \text{if } C_j(\pi) > C_{h_{\sigma(t)}}(\pi). \end{cases} \tag{53}$$

From the relation $C_j(\pi^*) \leq C_j(\pi)$ for $j \neq x$ in (53), together with the regularity of the scheduling objective functions, we have

$$f_j^{(s)}(C_j(\pi^*)) \leq f_{\max}^{(s)}(\pi) \text{ for } j \in \mathcal{J}^{(s)} \setminus \{x\} \text{ and } s = 1, 2, \cdots, m. \tag{54}$$

We now prove the following inequalities: For $s \in \{1, 2, \cdots, m\}$ with $x \in \mathcal{J}^{(\sigma(s)}$, we have

$$\begin{cases} f_x^{(\sigma(s))}(C_x(\pi^*)) < f_{\max}^{(\sigma(s))}(\pi), & \text{if } s \in \{1, 2, , \cdots, t-1\}, \\ f_x^{(\sigma(s))}(C_x(\pi^*)) \leq f_{\max}^{(\sigma(s))}(\pi), & \text{if } s \in \{t, t+1, \cdots, m\}. \end{cases} \tag{55}$$

Recall the relation $C_x(\pi^*) = C_{h_{\sigma(t)}}(\pi)$ in (53).

If $s \notin \{t, t+1, \cdots, l\}$, from (52), we have $x_s \geq q_{s,t} + 2$. Let y_s be the $(q_{s,t}+1)$-th job of agent $\sigma(s)$. Then $C_{y_s}(\pi) > C_{h_{\sigma(t)}}(\pi)$. From the lateness-like property, we have

$$f_x^{(\sigma(s))}(C_{h_{\sigma(t)}}(\pi)) \leq f_{y_s}^{(\sigma(s))}(C_{h_{\sigma(t)}}(\pi)) \leq f_{y_s}^{(\sigma(s))}(C_{y_s}(\pi)) \leq f_{\max}^{(\sigma(s))}(\pi). \tag{56}$$

Thus, the inequalities in (55) hold for $s \in \{l+1, l+2, \cdots, m\}$.

For $s \in \{1, 2, , \cdots, t-1\}$, we have $f_{y_s}^{(\sigma(s))}(C_{y_s}(\pi)) < f_{\max}^{(\sigma(s))}(\pi)$ since $C_{y_s}(\pi) > C_{h_{\sigma(t)}}(\pi) = C_{h_{\sigma(s)}}(\pi)$ and $h_{\sigma(s)}$ is the bottleneck job of agent $\sigma(s)$ under π. This means that the last inequality in (56) must be strict. Thus, the inequalities in (55) hold for $s \in \{1, 2, , \cdots, t-1\}$.

For $s \in \{t, t+1, \cdots, l\}$, from (52), we have $x_s \geq h_{\sigma(s)}^{(\sigma(s))} + 1$. From the lateness-like property again, we have $f_x^{(\sigma(s))}(C_{h_{\sigma(t)}}(\pi)) \leq f_{h_{\sigma(s)}}^{(\sigma(s))}(C_{h_{\sigma(t)}}(\pi)) = f_{\max}^{(\sigma(s))}(\pi)$, where the equality follows from the fact that $h_{\sigma(t)} = h_{\sigma(s)}$ is the bottleneck job of agent $\sigma(s)$ under π. This proves the inequalities in (55).

Since σ is a permutation of the m agents $\{1, 2, \cdots, m\}$, a weakened version of (55) can be written as: For $s \in \{1, 2, \cdots, m\}$ with $x \in \mathcal{J}^{(\sigma(s))}$, we have

$$f_x^{(s)}(C_x(\pi^*)) \leq f_{\max}^{(s)}(\pi). \tag{55'}$$

From (54) and (55'), we have $f_{\max}^{(s)}(\pi^*) \leq f_{\max}^{(s)}(\pi)$ for every agent s. But since π is a Pareto-optimal schedule, we must have

$$f_{\max}^{(s)}(\pi^*) = f_{\max}^{(s)}(\pi) = X_s \text{ for all } s = 1, 2, \cdots, m. \tag{57}$$

Thus, we conclude from (57) that π^* is also a Pareto-optimal schedule corresponding to $(X_1, X_2, \cdots, X_m)$.

For each $s \in \{1, 2, \cdots, m\}$, we use h_s^* to denote the bottleneck job of agent s under schedule π^*. Then $h_{\sigma(s)}^*$ is the bottleneck job of agent $\sigma(s)$ under π^*. For each job j with $j \neq x$ and $C_j(\pi) > C_{h_{\sigma(s)}}(\pi)$, we have $f_j^{(\sigma(s))}(C_j(\pi^*)) \leq f_j^{(\sigma(s))}(C_j(\pi)) < f_{\max}^{(\sigma(s))}(\pi) = f_{\max}^{(\sigma(s))}(\pi^*)$, where the first inequality follows from the relation $C_j(\pi^*) \leq C_j(\pi)$ for $j \neq x$ in (53) and the second inequality follows from the fact that the bottleneck job $h_{\sigma(s)}$ is the last job, assuming the value $f_{\max}^{(\sigma(s))}(\pi)$ in π. Thus, we have

$$j \neq h_{\sigma(s)}^* \text{ if } j \neq x \text{ and } C_j(\pi) > C_{h_{\sigma(s)}}(\pi). \tag{58}$$

For $s \in \{1, 2, \cdots, t-1\}$, the relation $f_x^{(\sigma(s))}(C_x(\pi^*)) < f_{\max}^{(\sigma(s))}(\pi)$ in (55) implies that $x \neq h_{\sigma(s)}^*$. From the relation $C_{h_{\sigma(s)}}(\pi) = C(\mathcal{S}_{h_{\sigma(s)}}^{\pi}) = C(\mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})})$ in (46), together with the fact that $x \notin \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}$, we have

$$\begin{aligned}
& \max\{C_j(\pi^*) : j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}\} \\
\geq\ & C(\mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}) \ = \ C_{h_{\sigma(s)}}(\pi) \\
=\ & \max\{C_j(\pi) : j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}\} \\
\geq\ & \max\{C_j(\pi^*) : j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}\},
\end{aligned}$$

where the last inequality follows from the relation $C_j(\pi^*) \leq C_j(\pi)$ for $j \neq x$ in (53). This implies that $C_{h_{\sigma(s)}}(\pi) = \max\{C_j(\pi^*) : j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}\}$. For every job $j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}$ with $j \neq h_{\sigma(s)}$, we have $j \neq x$, so $C_j(\pi^*) \leq C_j(\pi) < C_{h_{\sigma(s)}}(\pi)$. Thus, the only possibility is that $C_{h_{\sigma(s)}}(\pi^*) = \max\{C_j(\pi^*) : j \in \mathcal{J}_s^{(\sigma,\tilde{\mathbf{Q}})}\} = C_{h_{\sigma(s)}}(\pi)$. From (58), together with the fact that $x \neq h_{\sigma(s)}^*$, we conclude that

$$C_{h_{\sigma(s)}^*}(\pi^*) = C_{h_{\sigma(s)}}(\pi) \text{ and } h_{\sigma(s)}^* = h_{\sigma(s)} \text{ for } s = 1, 2, \cdots, t-1. \tag{59}$$

For $s \in \{l+1, l+2, \cdots, m\}$, from (53), we have $C_x(\pi^*) = C_{h_{\sigma(t)}}(\pi) < C_{h_{\sigma(s)}}(\pi) = C_{h_{\sigma(s)}}(\pi^*)$ and $C_j(\pi^*) = C_j(\pi)$ for all the jobs j with $C_j(\pi) > C_{h_{\sigma(s)}}(\pi)$. Thus, from (58) directly, we have

$$C_{h_{\sigma(s)}^*}(\pi^*) = C_{h_{\sigma(s)}}(\pi) \text{ and } h_{\sigma(s)}^* = h_{\sigma(s)} \text{ for } s = l+1, l+2, \cdots, m. \tag{60}$$

For $s \in \{t, t+1, \cdots, l\}$, we have $C_x(\pi^*) = C_{h_{\sigma(s)}}(\pi)$. From (58) directly, we have

$$C_{h^*_{\sigma(s)}}(\pi^*) \leq C_{h_{\sigma(s)}}(\pi) = C_x(\pi^*) \text{ for } s = t, t+1, \cdots, l. \tag{61}$$

If any one of the inequalities in (61) is strict, then from (59), (60), and (61), we have $\sum_{s=1}^m C_{h^*_s}(\pi^*) = \sum_{s=1}^m C_{h^*_{\sigma(s)}}(\pi^*) < \sum_{s=1}^m C_{h_{\sigma(s)}}(\pi) = \sum_{s=1}^m C_{h_s}(\pi)$. This contradicts the choice of π under condition (C1), since π^* is also a Pareto-optimal schedule corresponding to $(X_1, X_2, \cdots, X_m)$. Consequently, we have

$$C_{h^*_{\sigma(s)}}(\pi^*) = C_{h_{\sigma(s)}}(\pi) = C_x(\pi^*) \text{ and } h^*_{\sigma(s)} = x \text{ for } s = t, t+1, \cdots, l. \tag{62}$$

Now we have $\sum_{s=1}^m C_{h^*_s}(\pi^*) = \sum_{s=1}^m C_{h_s}(\pi)$. Thus, (C1) is satisfied by both π and π^*. But then, by combining (59), (60), and (62), we have $\sum_{s=1}^m h_s^{*(s)} - \sum_{s=1}^m h_s^{(s)} = \sum_{s=1}^m h_{\sigma(s)}^{*(\sigma(s))} - \sum_{s=1}^m h_{\sigma(s)}^{(\sigma(s))} = \sum_{s=t}^l (x_s - h_{\sigma(s)}^{(\sigma(s))}) > 0$, where the unique inequality follows from the relation $x_s \geq h_{\sigma(s)}^{(\sigma(s))} + 1$ in (52). This means $\sum_{s=1}^m h_s^{*(s)} > \sum_{s=1}^m h_s^{(s)}$, which contradicts the choice of π under condition (C2). Claim 3 follows.

Claim 3 states that, for each $t \in \{1, 2, \cdots, m\}$, $\mathcal{J}_t^{(\sigma, \tilde{\mathbf{Q}})}$ is a subset of $\mathcal{S}^\pi_{h_{\sigma(t)}}$ such that no jobs in $\mathcal{S}^\pi_{h_{\sigma(t)}} \setminus \mathcal{J}_t^{(\sigma, \tilde{\mathbf{Q}})}$ are interfering with respect to $\mathcal{S}^\pi_{h_{\sigma(t)}}$. By using Lemma 2.6′, we have

$$C_{h_{\sigma(t)}}(\pi) = C(\mathcal{S}^\pi_{h_{\sigma(t)}}) = C(\mathcal{J}_t^{(\sigma, \tilde{\mathbf{Q}})}) \text{ for } t \in \{1, 2, \cdots, m\}. \tag{63}$$

Recall from Claim 2 that the principal vector induced by $(\sigma, \mathbf{Q}^*)$ is given by $\mathbf{h}(\sigma, \mathbf{Q}^*) = (h_{\sigma(1)}, h_{\sigma(2)}, \cdots, h_{\sigma(m-1)})$. Thus, for each $t \in \{1, 2, \cdots, m-1\}$, the entry $X_{\sigma(t)}(\sigma, \mathbf{Q}^*)$ of the objective vector $\mathbf{X}(\sigma, \mathbf{Q}^*)$ is given by

$$\begin{aligned}
& X_{\sigma(t)}(\sigma, \mathbf{Q}^*) \\
= \; & f_{h_{\sigma(t)}}^{(\sigma(t))}(C(\mathcal{J}_t^{(\sigma, \mathbf{Q}^*)})) && \Leftarrow \text{ from (26)} \\
= \; & f_{h_{\sigma(t)}}^{(\sigma(t))}(C(\mathcal{J}_t^{(\sigma, \tilde{\mathbf{Q}})})) && \Leftarrow \text{ from (44)} \\
= \; & f_{h_{\sigma(t)}}^{(\sigma(t))}(C_{h_{\sigma(t)}}(\pi)) && \Leftarrow \text{ from (63)} \\
= \; & f_{\max}^{(\sigma(t))}(\pi) && \Leftarrow \text{ from the definition of } h_{\sigma(t)} \\
= \; & X_{\sigma(t)}. && \Leftarrow \text{ from the definition of } \pi
\end{aligned} \tag{64}$$

Recall that $X_{\sigma(m)}(\sigma, \mathbf{Q}^*)$ is the optimal value of problem

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(\sigma(m))} : f_{\max}^{(\sigma(t))} \leq X_{\sigma(t)}(\sigma, \mathbf{Q}) \; \forall t = 1, 2, \cdots, m-1$$

in (27). From (64), this problem is identical to

$$1|r_j, \text{pmtn}, \text{ND}|f_{\max}^{(\sigma(m))} : f_{\max}^{(\sigma(t))} \leq X_{\sigma(t)} \; \forall t = 1, 2, \cdots, m-1. \tag{65}$$

Since $(X_1, X_2, \cdots, X_m)$ is a Pareto-optimal point, from Lemma 2.1 or Lemma 2.2, the optimal value of the problem in (65) is just $X_{\sigma(m)}$. Then we have

$$X_{\sigma(m)}(\sigma, \mathbf{Q}^*) = X_{\sigma(m)}. \tag{66}$$

It follows from (64) and (66) that

$$\mathbf{X}(\sigma, \mathbf{Q}^*) = (X_1(\sigma, \mathbf{Q}^*), X_2(\sigma, \mathbf{Q}^*), \cdots, X_m(\sigma, \mathbf{Q}^*)) = (X_1, X_2, \cdots, X_m).$$

This proves the lemma. □

References

[1] A. Agnetis, J.C. Billaut, S. Gawiejnowicz, D. Pacciarelli and A. Soukhal, *Multiagent Scheduling - Models and Algorithms*, Springer, 2014.

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

[3] A. Agnetis, D. Pacciarelli and A. Pacifici, Multi-agent single machine scheduling, *Annals of Operations Research*, **150**(2007), 3-15.

[4] K.R. Baker, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints, *Operations Research*, **26**(1983), 111-120.

[5] K.R. Baker and J.C. Smith, A multiple-criterion model for machine scheduling, *Journal of Scheduling*, **6**(2003), 7-16.

[6] P. Brucker, *Scheduling Algorithms*, Springer-Verlag, Berlin, 2007.

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

[8] T.C.E. Cheng, C.T. Ng and J.J. Yuan, Multi-agent scheduling on a single machine to minimize total weighted number of tardy jobs, *Theoretical Computer Science*, **362**(2006), 273-281.

[9] T.C.E. Cheng, C.T. Ng and J.J. Yuan, Multi-agent scheduling on a single machine with max-form criteria, *European Journal of Operational Research*, **188**(2008), 603-609.

[10] J. Du and J.Y-T. Leung, Minimizing total tardiness on one machine is NP-hard, *Mathematics of Operations Research*, **15**(1990), 483-495.

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

[12] Y. Gao, J.J. Yuan, C.T. Ng and T.C.E. Cheng, A further study on two-agent parallel-batch scheduling with release dates and deteriorating jobs to minimize the makespan, *European Journal of Operational Research*, **273**(2019), 74-81.

[13] C. He and J.Y.-T. Leung, Two-agent scheduling of time-dependent jobs, *Journal of Combinatorial Optimization*, **34**(2017), 362-377.

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

[15] H. Hoogeveen, Multicritia scheduling, *European Journal of Operational Research*, **167**(2005), 592-623.

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

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

[18] E.L. Lawler, A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, *Annals of Discrete Mathematics*, **1** (1977), 331-342.

[19] J.Y.-T. Leung, M. Pinedo and G.H. Wan, Competitive two-agent scheduling and its applications, *Operations Research*, **58**(2010), 458-469.

[20] S.S. Li, T.C.E. Cheng, C.T. Ng and J.J. Yuan, Two-agent scheduling on a single sequential and compatible batching machine, *Naval Research Logistics*, **64**(2017), 628-641.

[21] R.T. Nelson, R.K. Sarin and R.L. Daniels, Scheduling with multiple performance measures: The one-machine case, *Management Science*, **32**(1986), 464-479.

[22] C.T. Ng, T.C.E. Cheng and J.J. Yuan, A note on the complexity of the problem of two-agent scheduling on a single machine, *Journal of Combinatorial Optimization*, **12**(2006), 387-394.

[23] D. Oron, D. Shabtay and G. Steiner, Single machine scheduling with two competing agents and equal job processing times, *European Journal of Operational Research*, **244**(2015), 86-99.

[24] F. Sadi and A. Soukhal, Complexity analyses for multi-agent scheduling problems with a global agent and equal length jobs, *Discrete Optimization*, **23**(2017), 93-104.

[25] F. Sadi, A. Soukhal and J.-C. Billaut, Solving multi-agent scheduling problems on parallel machines with a global objective function, *RAIRO - Operations Research*, **48**(2014), 255-269.

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

[27] V. T'Kindt and J.-C. Billaut, *Multicrireria Scheduling: Theory, Models and Algorithms*, Springer, Berlin, 2006.

[28] J.J. Yuan, Complexities of some problems on multi-agent scheduling on a single machine, *Journal of the Operations Research Society of China*, **4**(2016), 379-384.

[29] J.J. Yuan, Multi-agent scheduling on a single machine with a fixed number of competing agents to minimize the weighted sum of number of tardy jobs and makespans, *Journal of Combinatorial Optimization*, **34**(2017), 433-440.

[30] J.J. Yuan, Complexities of four problems on two-agent scheduling, *Optimization Letters*, **12**(2018), 763-780.

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

[32] J.J. Yuan, W.P. Shang and Q. Feng, A note on the scheduling with two families of jobs, *Journal of Scheduling*, **8**(2005), 537-542.