Pareto-scheduling with family jobs or ND-agent on a parallel-batch machine to minimize the makespan and maximum cost

Yuan Gao[a], Jinjiang Yuan[a,*], C.T. Ng[b], T.C.E. Cheng[b]

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

Zhengzhou, Henan 450001, China

[b]Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University,

Hung Hom, Kowloon, Hong Kong

Abstract: We study Pareto-scheduling on an unbounded parallel-batch machine that can process any number of jobs simultaneously in a batch. The processing time of a batch is equal to the maximum processing time of the jobs in the batch. We consider two Pareto-scheduling problems. In one problem, the jobs are partitioned into families and the jobs from different families cannot be processed together in the same batch. We assume that the number of families is a constant. The objective is to minimize the makespan and the maximum cost. In the other problem, we have two agents A and B, where each agent $E \in \{A, B\}$ has its job set $\mathcal{J}_E$, called the E-jobs. Assuming that the job sets $\mathcal{J}_A$ and $\mathcal{J}_B$ are not necessarily disjoint, we call the agents ND agents. The objective is to minimize the makespan of the A-jobs and the maximum cost of the B-jobs. We provide polynomial-time algorithms to solve the two Pareto-scheduling problems.

Key words: Pareto-scheduling · parallel-batch machine · family jobs · ND agents · polynomial time

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

1 Introduction

As introduced in Lee et al. (1992), a parallel-batch machine is a production line that can process up to b jobs simultaneously in a batch, where b is the batch capacity. Suppose that there are n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ to be scheduled on an unbounded parallel-batch machine without preemption. The n jobs and the machine are all available from time 0. Each job J_j has a non-negative integer processing time p_j, and a regular and integer-valued objective function $f_j(\cdot)$. Thus we only need to consider the feasible schedules in which the batches are scheduled consecutively without idle time. The processing time of a batch $\mathcal{B}$ is equal to the maximum processing time of jobs in batch $\mathcal{B}$, i.e., $p(\mathcal{B}) = \max\{p_j : J_j \in \mathcal{B}\}$, and the completion times of the jobs in batch $\mathcal{B}$ are defined as the completion time of batch $\mathcal{B}$. This implies that a feasible schedule can be represented by a batch sequence $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k)$ in which the completion time of a job $J_j \in \mathcal{B}_i$ in σ is given by $C_j(\sigma) = p(\mathcal{B}_1) + p(\mathcal{B}_2) + \cdots + p(\mathcal{B}_i)$. Brucker et al. (1998) provided fruitful results on parallel-batch scheduling.

The *Pareto-scheduling problem* for minimizing two objective functions γ' and γ'' is formulated as follows: A feasible schedule σ of the n jobs is called a *Pareto-optimal schedule* if there exists no other feasible schedule π such that

$$(\gamma'(\pi), \gamma''(\pi)) \leq (\gamma'(\sigma), \gamma''(\sigma)) \text{ and } (\gamma'(\pi), \gamma''(\pi)) \neq (\gamma'(\sigma), \gamma''(\sigma)).$$

In this case, $(\gamma'(\sigma), \gamma''(\sigma))$ is called a *Pareto-optimal point* corresponding to σ. The goal of the Pareto-scheduling problem is to find all the Pareto-optimal points and, for each Pareto-optimal point, the corresponding Pareto-optimal schedule. Adopting the three-field notation for scheduling problems introduced by Graham et al. (1979), we denote the Pareto-scheduling problem for minimizing two objective functions γ' and γ'' as $\alpha|\beta|^{\#}(\gamma', \gamma'')$, where α represents the machine environment, and β represents the job characteristics or the feasibility conditions.

Corresponding to the Pareto-scheduling problem $\alpha|\beta|^{\#}(\gamma', \gamma'')$, there are two constrained scheduling problems, namely $\alpha|\beta|\gamma' : \gamma'' \leq Q''$ and $\alpha|\beta|\gamma'' : \gamma' \leq Q'$. In the literature, for an algorithm $\mathcal{A}(Q'')$ for problem $\alpha|\beta|\gamma' : \gamma'' \leq Q''$, if the schedule σ gen-

erated by $\mathcal{A}(Q'')$ is optimal for problem $\alpha|\beta|\gamma' : \gamma'' \leq Q''$ and also Pareto-optimal for problem $\alpha|\beta|^{\#}(\gamma', \gamma'')$, then σ is called an *optimal and Pareto-optimal schedule* for problem $\alpha|\beta|\gamma' : \gamma'' \leq Q''$ and $\mathcal{A}(Q'')$ is called an *optimal and Pareto-optimal algorithm* for problem $\alpha|\beta|\gamma' : \gamma'' \leq Q''$.

Pareto-scheduling has been a popular topic in scheduling research. For detailed discussion of the methodologies and development of Pareto-scheduling research, the reader may refer to Hoogeveen (2005), T'kindt and Billaut (2006), Perez-Gonzalez and Framinan (2014), and Agnetis et al. (2014).

In the Pareto-scheduling problem with families jobs, the jobs are from R different families $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_R$ that form a partition of $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. For each r with $1 \leq r \leq R$, we also write $\mathcal{J}_r = \{J_{r,1}, J_{r,2}, \ldots, J_{r,n_r}\}$ and $n_r = |\mathcal{J}_r|$. Jobs from different families cannot be processed together in the same batch. Geng and Yuan (2015a) presented an $O(n^{2R+1})$ time algorithm to solve the Pareto-scheduling problem p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, L_{\max})$.

Baker and Smith (2003), and Agnetis et al. (2004) introduced two-agent scheduling in which the two agents have disjoint job sets. We call this two-agent scheduling model as *CO-agent scheduling*. After their pioneering works, two-agent scheduling has been extensively studied in the literature. Agnetis et al. (2014) provided a detailed review of the research on two-agent scheduling. Some new complexity results can be found in Yuan (2016, 2017, 2018).

Agnetis et al. (2014) introduced a new model of two-agent scheduling called *ND-agent scheduling*. In this scheduling model, there are two agents A and B, where each agent $E \in \{A, B\}$ has its job set $\mathcal{J}_E$, called the E-jobs. The ND-agent assumption means that the two job sets $\mathcal{J}_A$ and $\mathcal{J}_B$ are not necessarily disjoint. Then there are three related disjoint job sets, namely $\mathcal{J}_{A'} = \mathcal{J}_A \setminus \mathcal{J}_B$ (called the pure A-jobs), $\mathcal{J}_{B'} = \mathcal{J}_B \setminus \mathcal{J}_A$ (called the pure B-jobs), and $\mathcal{J}_{AB} = \mathcal{J}_A \cap \mathcal{J}_B$ (called the AB-jobs), which form a partition of $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. For each $E \in \{A, B, A', B', AB\}$, we write $\mathcal{J}_E = \{J_{E,1}, J_{E,2}, \ldots, J_{E,n_E}\}$ and $n_E = |\mathcal{J}_E|$. In this paper we assume that the jobs from different sets of $\mathcal{J}_{A'}$, $\mathcal{J}_{B'}$, and $\mathcal{J}_{AB}$ cannot be processed together in the same batch. We call this case as the *incompatible group* (*IG* for short) assumption. Feng et al. (2013) presented an $O(n_A + n_B^4)$ time algorithm to solve the Pareto-scheduling problem p-batch$(+\infty)|$CO-agent, IG$|^{\#}(C_{\max}^A, L_{\max}^B)$.

For research on Pareto-scheduling, we only review the closely related works. He et

al. (2007) proved that problem p-batch$(+\infty)||^{\#}(C_{\max}, L_{\max})$ can be solved in $O(n^3)$ time. He et al. (2014) proved that problem p-batch$(+\infty)||^{\#}(C_{\max}, f_{\max})$ can be solved in $O(n^3 \log \sum p_j)$ time, which is polynomial but not strongly polynomial. Geng and Yuan (2015b) proved that problem p-batch$(+\infty)||^{\#}(C_{\max}, f_{\max})$ can be solved in $O(n^4)$ time, which is strongly polynomial. This also implies that problem p-batch$(+\infty)||f_{\max}$ can be solved in $O(n^4)$ time. Gao (2017) presented polynomial-time algorithms to solve problem p-batch$(+\infty)|(r_j, p_j)$-agreeable$|^{\#}(C_{\max}, f_{\max})$, where "$(r_j, p_j)$-agreeable" means that the jobs have agreeable release dates and processing times, i.e., $r_i < r_j$ implies $p_i \leq p_j$.

In this paper we study two Pareto-scheduling problems on an unbounded parallel-batch machine. The first problem is Pareto-scheduling with family jobs for minimizing the makespan and the maximum cost, denoted by

$$\text{p-batch}(+\infty)|R\text{-family}|^{\#}(C_{\max}, f_{\max}). \tag{1}$$

When $R = 1$, the problem in (1) reduces to problem p-batch$(+\infty)||^{\#}(C_{\max}, f_{\max})$. Furthermore, it is evident that the problem in (1) is a generalization of problem p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, L_{\max})$. So we borrow some techniques presented in Geng and Yuan (2015a) to deal with the problem in (1).

The second problem is ND-agent Pareto-scheduling with incompatible group for minimizing the makespan of the A-jobs and the maximum cost of the B-jobs, denoted by

$$\text{p-batch}(+\infty)|\text{ND-agent, IG}|^{\#}(C^A_{\max}, f^B_{\max}). \tag{2}$$

The problem in (2) is a generalization of problem p-batch$(+\infty)|$CO-agent, IG$|^{\#}(C^A_{\max}, L^B_{\max})$. So we borrow some techniques presented in Feng et al. (2013) to deal with the problem in (2).

We organize the rest of the paper as follows: In Section 2 we present a polynomial-time algorithm to solve the problem in (1) when the number of families R is a constant. In Section 3 we present a polynomial-time algorithm to solve the problem in (2). We conclude the paper and suggest future research topics in the last section.

2 Pareto-scheduling with family jobs

For the Pareto-scheduling problem p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, L_{\max})$, Geng and Yuan (2015a) presented an $O(n^{2R+1})$ time solution algorithm. In the following we borrow their

method to solve the Pareto-scheduling problem in (1), i.e., p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, f_{\max})$.

For each family $\mathcal{J}_r = \{J_{r,1}, J_{r,2}, \ldots, J_{r,n_r}\}$ with $1 \le r \le R$, we re-index its jobs in the shortest processing time (SPT) order so that $p_{r,1} \le p_{r,2} \le \cdots \le p_{r,n_r}$. A schedule $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m)$ is called an *SPT-batch schedule*, if for every two jobs $J_{r,i}$ and $J_{r,j}$ in the same family $\mathcal{J}_r$ with $1 \le r \le R$, $i < j$ implies $C_{r,i}(\sigma) \le C_{r,j}(\sigma)$. This means that in an SPT-batch schedule, the jobs of each family are batched and scheduled in the SPT order. Applying a similar analysis to that in the proof of Lemma 1 in Brucker et al. (1998), we derive the following result.

Lemma 2.1. *For each Pareto-optimal point of the problem in (1), there exists a corresponding Pareto-optimal schedule that is also an SPT-batch schedule.*

Lemma 2.1 shows that it suffices to consider SPT-batch schedules in our discussion. For each family $\mathcal{J}_r$ with $1 \le r \le R$, we further suppose that the n_r jobs in $\mathcal{J}_r$ have n'_r distinct processing times $p_{(r,1)} < p_{(r,2)} < \cdots < p_{(r,n'_r)}$. For each i with $1 \le i \le n'_r$, we define $\mathcal{J}_{(r,i)} = \{J_j \in \mathcal{J}_r : p_j = p_{(r,i)}\}$, $n_{(r,i)} = |\mathcal{J}_{(r,i)}|$, and $f_{(r,i)}(t) = \max\{f_j(t) : J_j \in \mathcal{J}_{(r,i)}\}$ for $t \ge 0$. By Lemma 2.1, we can regard the jobs of $\mathcal{J}_{(r,i)}$ as a composite job $J_{(r,i)}$ with the processing time $p_{(r,i)}$ and the cost function $f_{(r,i)}(t)$. We see that $\sum_{1 \le i \le n'_r} n_{(r,i)} = n_r$ and $f_{(r,i)}(t)$ can be calculated in $O(n_{(r,i)})$ time for fixed t. In fact, we do not need to calculate $f_{(r,i)}(t)$ for all $t \ge 0$ in our algorithm. Then we use $\mathcal{J}_{(r)}$ to denote the job set $\{J_{(r,1)}, \ldots, J_{(r,n'_r)}\}$ in the sequel.

For ease of exposition, for each family $\mathcal{J}_{(r)}$ with $1 \le r \le R$, we add a null job $J_{(r,0)}$ with $p_{(r,0)} = 0$ and $f_{(r,0)}(t) = -\infty$ for $t \ge 0$. For each i with $0 \le i \le n'_r$, let $\mathcal{J}_r^i = \{J_{(r,0)}, J_{(r,1)}, \ldots, J_{(r,i)}\}$. If a batch $\mathcal{B}$ consists of the jobs in $\mathcal{J}_{(r)}$, we call $\mathcal{B}$ an r-batch. In every schedule, for each r with $1 \le r \le R$, we assume that job $J_{(r,0)}$ forms a single r-batch starting at time 0.

To solve the problem in (1), we first consider its associated constrained scheduling problem

$$\text{p-batch}(+\infty)|R\text{-family}|C_{\max} : f_{\max} \le Q. \tag{3}$$

Corresponding to this problem, we consider the following two auxiliary problems.

- **Problem Aux**$(i_1, \ldots, i_R)$: This is the problem in (3) for the job set $\mathcal{J}_1^{i_1} \cup \cdots \cup \mathcal{J}_R^{i_R}$

with $(i_1, \ldots, i_R) \in X$, where $X = \{(i_1, \ldots, i_R) : 0 \le i_r \le n'_r,\ 1 \le r \le R\}$.

• **Problem Aux$^r(i_1, \ldots, i_R)$**: This is a restricted version of problem Aux$(i_1, \ldots, i_R)$ with $i_r \ge 1$ in which the last batch in every feasible schedule is an r-batch.

Based on Lemma 2.1, we present a dynamic programming algorithm DP(Q) to solve the problem in (3) in the following. Our algorithm will generate an optimal and Pareto-optimal schedule $\sigma(i_1, \ldots, i_R)$ for problem Aux$(i_1, \ldots, i_R)$. The makespan and maximum cost of $\sigma(i_1, \ldots, i_R)$ are denoted by $C(i_1, \ldots, i_R)$ and $f(i_1, \ldots, i_R)$, respectively. Our algorithm will also generate an optimal and Pareto-optimal schedule $\sigma^r(i_1, \ldots, i_R)$ for problem Aux$^r(i_1, \ldots, i_R)$. The makespan and maximum cost of $\sigma^r(i_1, \ldots, i_R)$ are denoted by $C^r(i_1, \ldots, i_R)$ and $f^r(i_1, \ldots, i_R)$, respectively.

Algorithm DP(Q): For the problem in (3), Aux$(i_1, \ldots, i_R)$, and Aux$^r(i_1, \ldots, i_R)$.

Step 1: The initial condition is $C(0, \ldots, 0) = 0$ and $f(0, \ldots, 0) = -\infty$.

Step 2: For each $(i_1, \ldots, i_R) \in X^+$, where $X^+ = \{(i_1, \ldots, i_R) \in X : i_1 + \cdots + i_R \ge 1\}$, the recursive functions are given by

$$C(i_1, \ldots, i_R) = \min\{C^r(i_1, \ldots, i_R) : 1 \le r \le R,\ i_r \ge 1\}, \tag{4}$$

and

$$\begin{aligned} & f(i_1, \ldots, i_R) \\ = \ & \min\{f^r(i_1, \ldots, i_R) : 1 \le r \le R,\ C^r(i_1, \ldots, i_R) = C(i_1, \ldots, i_R)\}, \end{aligned} \tag{5}$$

where for each $r \in \{1, 2, \ldots, R\}$ with $i_r \ge 1$, we have

$$C^r(i_1, \ldots, i_R) = C(i_1, \ldots, \bar{k}_r, \ldots, i_R) + p_{(r,i_r)}, \tag{6}$$

and

$$f^r(i_1, \ldots, i_R) = \max\{f(i_1, \ldots, \bar{k}_r, \ldots, i_R), \max_{\bar{k}_r+1 \le i \le i_r} f_{(r,i)}(C^r(i_1, \ldots, i_R))\}, \tag{7}$$

where $\bar{k}_r = \bar{k}_r(i_1, \ldots, i_R)$ is the index k_r defined by

$$\begin{cases} 0 \le k_r < i_r,\ \max_{k_r+1 \le i \le i_r} f_{(r,i)}(C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)}) \le Q \\ \text{so that } C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)} \text{ is as small as possible.} \end{cases}$$

Step 3: Output $C(n'_1, \ldots, n'_R)$ and $f(n'_1, \ldots, n'_R)$.

Note that if $C(n'_1, \ldots, n'_R) = +\infty$, then the problem in (3) is infeasible. Alternatively, if $C(n'_1, \ldots, n'_R) < +\infty$, then the final objective vector $(C_{\max}, f_{\max})$ of the problem in (3) is given by $(C(n'_1, \ldots, n'_R), f(n'_1, \ldots, n'_R))$ and the corresponding schedule is given by $\sigma(n'_1, \ldots, n'_R)$. Furthermore, Algorithm DP(Q) can also calculate all $C(i_1, \ldots, i_R)$ and $f(i_1, \ldots, i_R)$ with $(i_1, \ldots, i_R) \in X$, and all $C^r(i_1, \ldots, i_R)$ and $f^r(i_1, \ldots, i_R)$ with $(i_1, \ldots, i_R) \in X^+$ and $i_r \geq 1$. The corresponding schedules $\sigma(i_1, \ldots, i_R)$ and $\sigma^r(i_1, \ldots, i_R)$ can be found by backtracking.

Theorem 2.1. *The schedules $\sigma(i_1, \ldots, i_R)$ are optimal for problems Aux$(i_1, \ldots, i_R)$ with $(i_1, \ldots, i_R) \in X$, and the schedules $\sigma^r(i_1, \ldots, i_R)$ are optimal for problems Aux$^r(i_1, \ldots, i_R)$ with $(i_1, \ldots, i_R) \in X^+$ and $i_r \geq 1$.*

Proof. It suffices to show that $C(i_1, \ldots, i_R)$ and $C^r(i_1, \ldots, i_R)$ are correctly calculated by (4) and (6) in Algorithm DP(Q). The definition $C(0, \ldots, 0) = 0$ is obviously true. Hence, we may assume that $(i_1, \ldots, i_R) \in X^+$ and $C(i_1, \ldots, i_R) < +\infty$.

For an optimal schedule for problem Aux$(i_1, \ldots, i_R)$, assuming $C(i_1, \ldots, i_R)$, there exists an $r \in \{1, 2, \ldots, R\}$ with $i_r \geq 1$ so that the last batch is an r-batch. It follows that $C(i_1, \ldots, i_r, \ldots, i_R) = \min\{C^r(i_1, \ldots, i_R) : 1 \leq r \leq R, i_r \geq 1\}$. Thus, (4) calculates $C(i_1, \ldots, i_R)$ correctly.

For each $r \in \{1, 2, \ldots, R\}$ with $i_r \geq 1$ and $C^r(i_1, \ldots, i_R) < +\infty$, let $\sigma = (\mathcal{B}_1, \ldots, \mathcal{B}_k)$ be an optimal schedule for problem Aux$^r(i_1, \ldots, i_R)$, assuming $C^r(i_1, \ldots, i_R)$. Then the last batch $\mathcal{B}_k$ of σ is an r-batch, so $\mathcal{B}_k = \{J_{(r,k_r+1)}, \ldots, J_{(r,i_r)}\}$ for some k_r with $0 \leq k_r < i_r$. Thus $\sigma' = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{k-1})$ is a feasible schedule for problem Aux$(i_1, \ldots, k_r, \ldots, i_R)$. It follows that $C^r(i_1, \ldots, i_R) = C_{\max}(\sigma) = C_{\max}(\sigma') + p_{(r,i_r)} \geq C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)} \geq C^r(i_1, \ldots, i_r, \ldots, i_R)$. This implies that $C^r(i_1, \ldots, i_R) = C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)}$ and σ' is an optimal schedule for problem Aux$(i_1, \ldots, k_r, \ldots, i_R)$, assuming $C(i_1, \ldots, k_r, \ldots, i_R)$. Thus, (6) calculates $C^r(i_1, \ldots, i_R)$ correctly. The result follows. □

Lemma 2.2. *For each $(i_1, \ldots, i_R) \in X^+$ with $C(i_1, \ldots, i_R) < +\infty$, we have $C(i_1, \ldots, i_r - 1, \ldots, i_R) < C(i_1, \ldots, i_r, \ldots, i_R)$ for $i_r = 1, \ldots, n'_r$.*

Proof. We use $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m)$ to denote the schedule generated by Algorithm

DP(Q) for problem Aux$(i_1, \ldots, i_r, \ldots, i_R)$. Then we have $C_{\max}(\sigma) = C(i_1, \ldots, i_r, \ldots, i_R)$ and $f_{\max}(\sigma) \leq Q$. We may suppose that job $J_{(r,i_r)}$ belongs to some batch $\mathcal{B}_q$. Let $\sigma' = (\mathcal{B}_1, \ldots, \mathcal{B}_q \setminus \{J_{(r,i_r)}\}, \ldots, \mathcal{B}_m)$ be a new schedule obtained from σ by removing job $J_{(r,i_r)}$. We can easily observe that σ' is a feasible schedule for problem Aux$(i_1, \ldots, i_r - 1, \ldots, i_R)$. From the fact that $p_{(r,1)} < p_{(r,2)} < \cdots < p_{(r,i_r)}$, $J_{(r,i_r)}$ is the unique longest job in batch $\mathcal{B}_q$. It follows that $C(i_1, \ldots, i_r - 1, \ldots, i_R) \leq C_{\max}(\sigma') < C_{\max}(\sigma) = C(i_1, \ldots, i_r, \ldots, i_R)$. The result follows. □

Lemma 2.3. *For $(i_1, \ldots, i_R)$, $(i'_1, \ldots, i'_R) \in X$ with $(i_1, \ldots, i_R) \geq (i'_1, \ldots, i'_R)$, we have $\bar{k}_r(i_1, \ldots, i_R) \geq \bar{k}_r(i'_1, \ldots, i'_R)$ for $r = 1, 2, \ldots, R$.*

Proof. If $\bar{k}_r(i_1, \ldots, i_R) = n'_r$ or $\bar{k}_r(i'_1, \ldots, i'_R) = -\infty$, the result holds trivially. Hence, we may assume that $\bar{k}_r(i_1, \ldots, i_R) < n'_r$ and $\bar{k}_r(i'_1, \ldots, i'_R) \geq 0$. Then we have $i'_r \geq 1$ and $C^r(i'_1, \ldots, i'_R) \leq C^r(i_1, \ldots, i_R) < +\infty$. In the following it suffices to consider only the case that $(i'_1, \ldots, i'_R) = (i_1, \ldots, i_{r'} - 1, \ldots, i_R)$ for some $r' \in \{1, 2, \ldots, R\}$ with $i_{r'} \geq 1$. For our purpose, we first present a claim as follows:

Claim. Suppose that $i_r \geq 1$. Then, for every optimal SPT-batch schedule σ for problem Aux$^r(i_1, \ldots, i_R)$, assuming $C^r(i_1, \ldots, i_R)$, $\bar{k}_r(i_1, \ldots, i_R)$ is the index of the longest job in the last second r-batch of σ. Furthermore, $\bar{k}_r(i_1, \ldots, i_R)$ is the minimum of the index k_r so that $0 \leq k_r < i_r$ and $\max_{k_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$.

In order to prove the above claim, we use σ to denote an optimal SPT-batch schedule for problem Aux$^r(i_1, \ldots, i_R)$, assuming $C^r(i_1, \ldots, i_R)$. Let k'_r be the index of the longest job in the last second r-batch of σ. Then we have $0 \leq k'_r < i_r$ and $\max_{k'_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$. Let k''_r be the minimum of the index k_r so that $0 \leq k_r < i_r$ and $\max_{k_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$. This implies that $k'_r \geq k''_r$. If $k'_r \geq k''_r + 1$, from Lemma 2.2, we have $C(i_1, \ldots, k'_r, \ldots, i_R) > C(i_1, \ldots, k''_r, \ldots, i_R)$. Then $C^r(i_1, \ldots, i_R) = C(i_1, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r)} > C(i_1, \ldots, k''_r, \ldots, i_R) + p_{(r,i_r)} = C^r(i_1, \ldots, i_R)$, a contradiction. It follows that $k'_r = k''_r$. From the fact that $\sigma^r(i_1, \ldots, i_R)$ is an optimal SPT-batch schedule for problem Aux$^r(i_1, \ldots, i_R)$, assuming $C^r(i_1, \ldots, i_R)$, we have $\bar{k}_r(i_1, \ldots, i_R) = k'_r = k''_r$. The claim follows.

The above claim shows that the value $\bar{k}_r$ in Algorithm DP(Q) is unique. Set $k'_r = \bar{k}_r(i_1, \ldots, i_r, \ldots, i_R)$. Then $0 \leq k'_r < i_r$. We distinguish two cases as follows:

Case 1. $r = r'$. Since $i_r - 1 = i'_r \geq 1$, we have $i_r \geq 2$. If $k'_r = i_r - 1$, then $\bar{k}_r(i_1, \ldots, i_r - 1, \ldots, i_R) < i_r - 1 = k'_r$. If $k'_r < i_r - 1$, then the two jobs $J_{(r,i_r-1)}$ and $J_{(r,i_r)}$ belong to the last batch of $\sigma^r(i_1, \ldots, i_R)$. This implies that $\max_{k'_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$. From the fact that $p_{(r,i_r-1)} < p_{(r,i_r)}$, we have $\max_{k'_r+1 \leq i \leq i_r-1} f_{(r,i)}(C(i_1, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r-1)}) \leq Q$. From the above claim, we have $\bar{k}_r(i_1, \ldots, i_r - 1, \ldots, i_R) \leq \bar{k}_r(i_1, \ldots, i_r, \ldots, i_R)$.

Case 2. $r \neq r'$. We may suppose that $r' < r$. From Lemma 2.2, we have $C(i_1, \ldots, i_{r'} - 1, \ldots, k'_r, \ldots, i_R) < C(i_1, \ldots, i_{r'}, \ldots, k'_r, \ldots, i_R)$. By the implementation of Algorithm DP(Q), we have $\max_{k'_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, i_{r'}, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$. It follows that $\max_{k'_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, i_{r'} - 1, \ldots, k'_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$. From the above claim, we have $\bar{k}_r(i_1, \ldots, i_{r'} - 1, \ldots, i_R) \leq \bar{k}_r(i_1, \ldots, i_{r'}, \ldots, i_R)$. The lemma follows. □

From Lemma 2.3, we only need to check at most $\bar{k}_r(i_1, \ldots, i_R) - \bar{k}_r(i_1, \ldots, i_r - 1, \ldots, i_R) + 1$ indices k_r to find our required index $\bar{k}_r(i_1, \ldots, i_R)$ in each iteration of Algorithm DP(Q). Recall that, for each given time t, $f_{(r,i)}(t)$ can be calculated in $O(n_{(r,i)})$ time. Then the value $\max_{k_r+1 \leq i \leq i_r} f_{(r,i)}(C(i_1, \ldots, k_r, \ldots, i_R) + p_{(r,i_r)}) \leq Q$ can be checked in $O(n_r)$ time for each given k_r. Thus, given r and $i_{r'}$ with $1 \leq r' \leq R$ and $r' \neq r$, the n'_r value $\bar{k}_r(i_1, \ldots, i_R)$, $1 \leq i_r \leq n'_r$, can be calculated in $O(n_r^2)$ time.

The above discussion implies that Algorithm DP(Q) runs in $O(R \cdot \prod_{1 \leq r \leq R} n_r^2)$ time. Since $\sum_{r=1}^{R} n_r = n$, we have $\prod_{1 \leq r \leq R} n_r^2 \leq n^{2R} \cdot R^{-2R}$. Consequently, we have the following result.

Lemma 2.4. *The time complexity of Algorithm DP(Q) is $O(R \cdot \prod_{1 \leq r \leq R} n_r^2) = O(n^{2R})$.*

The following lemma is critical for our discussion.

Lemma 2.5. *Algorithm DP(Q) is optimal and Pareto-optimal for the problem in (3).*

Proof. We prove that, for each $(i_1, \ldots, i_R) \in X$ with $C(i_1, \ldots, i_R) < +\infty$, schedule $\sigma(i_1, \ldots, i_R)$ is optimal and Pareto-optimal for problem Aux$(i_1, \ldots, i_R)$, and for each $(i_1, \ldots, i_R) \in X^+$ with $i_r \geq 1$ and $C^r(i_1, \ldots, i_R) < +\infty$, schedule $\sigma^r(i_1, \ldots, i_R)$ is optimal and Pareto optimal for problem Aux$^r(i_1, \ldots, i_R)$. From Theorem 2.1, we only need to show the Pareto-optimality in the following.

For each $(i_1, \ldots, i_R) \in X$ with $C(i_1, \ldots, i_R) < +\infty$, let $\pi(i_1, \ldots, i_R)$ be an optimal and Pareto-optimal schedule for problem Aux$(i_1, \ldots, i_R)$, and for each $(i_1, \ldots, i_R) \in X^+$

with $i_r \geq 1$ and $C^r(i_1,\ldots,i_R) < +\infty$, let $\pi^r(i_1,\ldots,i_R)$ be an optimal and Pareto-optimal schedule for problem $\text{Aux}^r(i_1,\ldots,i_R)$. From Theorem 2.1, the makespans of $\pi(i_1,\ldots,i_R)$ and $\pi^r(i_1,\ldots,i_R)$ are $C(i_1,\ldots,i_R)$ and $C^r(i_1,\ldots,i_R)$, respectively. Thus, we only need to show that the maximum cost of $\pi(i_1,\ldots,i_R)$ is $f(i_1,\ldots,i_R)$ and the maximum cost of $\pi^r(i_1,\ldots,i_R)$ is $f^r(i_1,\ldots,i_R)$ by induction.

Note that $\pi(0,\ldots,0) = \sigma(0,\ldots,0)$. Then the maximum cost of $\pi(0,\ldots,0)$ is $f(0,\ldots,0) = -\infty$. Inductively, for $(i_1,\ldots,i_R) \in X^+$ with $i_r \geq 1$ and $C(i_1,\ldots,i_R) < +\infty$, we assume that the maximum cost of $\pi(i_1,\ldots,i'_r,\ldots,i_R)$ is $f(i_1,\ldots,i'_r,\ldots,i_R)$ for every i'_r with $0 \leq i'_r \leq i_r - 1$. Note that $\pi^r(i_1,\ldots,i_R)$ is an optimal SPT-batch schedule for problem $\text{Aux}^r(i_1,\ldots,i_R)$, assuming $C^r(i_1,\ldots,i_R)$. From Lemma 2.3, $k'_r = \bar{k}_r(i_1,\ldots,i_R)$ is the index of the longest job in the last second r-batch of π^r. We use π' to denote the schedule obtained from π^r by deleting the last batch $\mathcal{B} = \{J_{(r,k'_r+1)}, J_{(r,k'_r+2)},\ldots,J_{(r,n'_r)}\}$. From Theorem 2.1, we observe that schedule π' is optimal for problem $\text{Aux}(i_1,\ldots,k'_r,\ldots,i_R)$, assuming $C(i_1,\ldots,k'_r,\ldots,i_R)$.

By the induction hypothesis, we see that schedule $\pi(i_1,\ldots,k'_r,\ldots,i_R)$ is optimal and Pareto-optimal for problem $\text{Aux}(i_1,\ldots,k'_r,\ldots,i_R)$ with the maximum cost $f(i_1,\ldots,k'_r,\ldots,i_R)$. Then we have $f_{\max}(\pi') \geq f(i_1,\ldots,k'_r,\ldots,i_R)$. Since $\pi^r(i_1,\ldots,i_R)$ and $\sigma^r(i_1,\ldots,i_R)$ have the same last batch $\mathcal{B}$, we have $f_{\max}(\pi^r(i_1,\ldots,i_R)) \geq f^r(i_1,\ldots,i_R)$. Note that $\pi^r(i_1,\ldots,i_R)$ is optimal and Pareto-optimal for problem $\text{Aux}^r(i_1,\ldots,i_R)$ and $\sigma^r(i_1,\ldots,i_R)$ is optimal for problem $\text{Aux}^r(i_1,\ldots,i_R)$. Then we have $f_{\max}(\pi^r(i_1,\ldots,i_R)) \leq f^r(i_1,\ldots,i_R)$. Thus, $f_{\max}(\pi^r(i_1,\ldots,i_R)) = f^r(i_1,\ldots,i_R)$, i.e., the maximum cost of $\pi^r(i_1,\ldots,i_R)$ is $f^r(i_1,\ldots,i_R)$.

Now we consider $\pi = \pi(i_1,\ldots,i_R)$. Since π is optimal for problem $\text{Aux}(i_1,\ldots,i_R)$, there is an $r \in \{1,2,\ldots,R\}$ with $i_r \geq 1$ so that π is also optimal for problem $\text{Aux}^r(i_1,\ldots,i_R)$. Thus, $C_{\max}(\pi) = C(i_1,\ldots,i_R) = C^r(i_1,\ldots,i_R)$. Note that $\pi^r(i_1,\ldots,i_R)$ is optimal and Pareto-optimal for problem $\text{Aux}^r(i_1,\ldots,i_R)$ with $C_{\max}(\pi^r(i_1,\ldots,i_R)) = C^r(i_1,\ldots,i_R) = C_{\max}(\pi)$ and $f_{\max}(\pi^r(i_1,\ldots,i_R)) = f^r(i_1,\ldots,i_R)$. Then we have $f_{\max}(\pi) \geq f_{\max}(\pi^r(i_1,\ldots,i_R)) = f^r(i_1,\ldots,i_R)$. From (5), we have $f(i_1,\ldots,i_R) \leq f^r(i_1,\ldots,i_R) \leq f_{\max}(\pi)$. From the fact that π is optimal and Pareto-optimal for problem $\text{Aux}(i_1,\ldots,i_R)$ and $\sigma(i_1,\ldots,i_R)$ is optimal for problem $\text{Aux}(i_1,\ldots,i_R)$, we have $f_{\max}(\pi) \leq f(i_1,\ldots,i_R)$. Thus, $f_{\max}(\pi) = f(i_1,\ldots,i_R)$, i.e., the maximum cost of $\pi(i_1,\ldots,i_R)$ is $f(i_1,\ldots,i_R)$. The result follows. □

From Lemma 2.5, we present an solution algorithm for the problem in (1) as follows:

Pareto(DP(Q)): Initially set Q to be a sufficiently large number. Then we use Algorithm DP(Q) to solve the problem in (3) to obtain the first Pareto-optimal schedule σ_1 and the first Pareto-optimal point $(C_{\max}(\sigma_1), f_{\max}(\sigma_1))$. In general, if σ_i and $(C_{\max}(\sigma_i), f_{\max}(\sigma_i))$ have been generated, we reset $Q := f_{\max}(\sigma_i) - 1$ and run Algorithm DP(Q) for the corresponding problem in (3) to obtain the $(i+1)$-th Pareto-optimal schedule σ_{i+1} and the $(i+1)$-th Pareto-optimal point $(C_{\max}(\sigma_{i+1}), f_{\max}(\sigma_{i+1}))$. This procedure is repeated until we meet an index K so that σ_K and $(C_{\max}(\sigma_K), f_{\max}(\sigma_K))$ have been generated and Algorithm DP(Q) determines that the problem in (3) is infeasible for $Q = f_{\max}(\sigma_K) - 1$.

Theorem 2.2. *The problem in (1) can be solved by Pareto(DP(Q)) in $O(n^{3R+1})$ time.*

Proof. We mainly need to estimate the number of Pareto-optimal points of the problem in (1). We may suppose that $\sigma_1, \sigma_2, \ldots, \sigma_K$ are the schedules generated by Pareto(DP(Q)) for a given instance. From the implementation of Pareto(DP(Q)), we have

$$C_{\max}(\sigma_1) < C_{\max}(\sigma_2) < \cdots < C_{\max}(\sigma_K)$$

and

$$f_{\max}(\sigma_1) > f_{\max}(\sigma_2) > \cdots > f_{\max}(\sigma_K).$$

Let $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$. For each $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m) \in \mathcal{S}$, we set $\mathcal{B}^{\leq l}(\sigma) = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_l$ for $l = 1, 2, \ldots, m$. From Lemma 2.5, we observe that the subschedule $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_l)$ of σ is optimal and Pareto-optimal for the problem in (1) restricted to the job instance $\mathcal{B}^{\leq l}(\sigma)$.

For each $\sigma \in \mathcal{S}$, we call J_j a *key job* of σ if $f_j(\sigma) = f_{\max}(\sigma)$. The batch containing a key job is called a *key batch*. For each r with $1 \leq r \leq R$, we use $\mathcal{S}_r$ to denote the set of the schedules in $\mathcal{S}$ whose last key batch is an r-batch. Then $\mathcal{S}_r$, $r = 1, \ldots, R$, form a partition of $\mathcal{S}$. We first establish the following claim.

Claim. Let $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m)$ and $\sigma' = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{m'})$ be two schedules in $\mathcal{S}_r$ for some r. If $\mathcal{B}_l$ is a key batch of σ and $\mathcal{A}_j$ is a key batch of σ' so that $\mathcal{B}_l = \mathcal{A}_j$, then $\mathcal{B}^{\leq l}(\sigma) \neq \mathcal{A}^{\leq j}(\sigma')$.

In order to prove the above claim, we suppose to the contrary that $\mathcal{B}^{\leq l}(\sigma) = \mathcal{A}^{\leq j}(\sigma')$. From Lemma 2.5, for the same job instance $\mathcal{B}^{\leq l}(\sigma) = \mathcal{A}^{\leq j}(\sigma')$, $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_l)$ is optimal and Pareto-optimal for the problem in (3) with $Q = f_{\max}(\sigma)$ and $(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_j)$ is optimal and Pareto-optimal for the problem in (3) with $Q = f_{\max}(\sigma')$. Without loss of generality, we assume that $C_{\max}(\sigma) < C_{\max}(\sigma')$ and $f_{\max}(\sigma) > f_{\max}(\sigma')$.

From the Pareto-optimality of $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_l)$ and $(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_j)$, we have $C_{\mathcal{B}_l}(\sigma) < C_{\mathcal{A}_j}(\sigma')$. Note that $\mathcal{B}_l$ is a key batch of σ and $\mathcal{A}_j$ is a key batch of σ' so that $\mathcal{B}_l = \mathcal{A}_j$. By the regularity of function $f_j(\cdot)$ for each $j \in \{1, 2, \ldots, n\}$, we have $f_{\max}(\sigma) \leq f_{\max}(\sigma')$, a contradiction. The claim follows.

Given a family index r with $1 \leq r \leq R$ and a pair (l_r, l'_r) with $1 \leq l_r \leq l'_r \leq n'_r$, we use $\mathcal{S}_r(l_r, l'_r)$ to denote the set of the schedules σ in $\mathcal{S}_r$ whose last key batch, denoted by $\mathcal{B}_{m^*(\sigma)}$, is of the form $\{J_{(r,l_r)}, J_{(r,l_r+1)}, \ldots, J_{(r,l'_r)}\}$. Note that σ is an SPT-batch schedule. Then we have $\mathcal{B}^{\leq m^*(\sigma)}(\sigma) = \mathcal{J}_1^{i_1} \cup \cdots \cup \mathcal{J}_R^{i_R}$ for some $(i_1, \ldots, i_R) \in X$ with $i_r = l'_r \geq 1$. It follows that

$$\mathcal{S}_r(l_r, l'_r) \subseteq \{\sigma \in \mathcal{S}_r : \mathcal{B}^{\leq m^*(\sigma)}(\sigma) = \mathcal{J}_1^{i_1} \cup \cdots \cup \mathcal{J}_R^{i_R}, (i_1, \ldots, i_R) \in X, \, i_r = l'_r\}. \tag{8}$$

From the above claim and (8), we have

$$|\mathcal{S}_r(l_r, l'_r)| \leq |\{(i_1, \ldots, i_R) \in X, \, i_r = l'_r\}| = \textstyle\prod_{1 \leq s \leq R, s \neq r}(n'_s + 1). \tag{9}$$

From (9), we have $|\mathcal{S}_r| \leq \frac{n'_r(n'_r+1)}{2} \cdot \prod_{1 \leq s \leq R, s \neq r}(n'_s + 1) \leq \frac{n_r}{2} \cdot \prod_{1 \leq s \leq R}(n_s + 1)$. This implies that $|\mathcal{S}| = \sum_{1 \leq r \leq R} |\mathcal{S}_r| \leq \sum_{1 \leq r \leq R} \frac{n_r}{2} \cdot \prod_{1 \leq s \leq R}(n_s + 1) = \frac{n}{2} \cdot \prod_{1 \leq s \leq R}(n_s + 1) \leq \frac{n}{2}(1 + \frac{n}{R})^R = O(n^{R+1})$. From Lemma 2.4 and the implementation of Pareto(DP(Q)), the problem in (1) can be solved in $O(n^{3R+1})$ time. The result follows. □

From Theorem 2.2, we have the following result.

Corollary 2.1. *The problem in (1) with $R = 2$ can be solved in $O(n^7)$ time and the problem in (1) with $R = 1$ can be solved in $O(n^4)$ time.*

3 Pareto-scheduling with ND-agent

For the Pareto-scheduling problem p-batch$(+\infty)$|CO-agent, IG|$^{\#}(C^A_{\max}, L^B_{\max})$, Feng et al. (2013) presented an $O(n_A + n_B^4)$ time solution algorithm. In the following we borrow their

method to solve the Pareto-scheduling problem in (2), i.e., p-batch$(+\infty)$|ND-agent, IG|$^{\#}(C^A_{\max}, f^B_{\max})$, where the ND-agent assumption means that $\mathcal{J}_A$ and $\mathcal{J}_B$ are not necessarily disjoint. Let $\mathcal{J}_{A'} = \mathcal{J}_A \setminus \mathcal{J}_B$ (called the pure A-jobs), $\mathcal{J}_{B'} = \mathcal{J}_B \setminus \mathcal{J}_A$ (called the pure B-jobs), and $\mathcal{J}_{AB} = \mathcal{J}_A \cap \mathcal{J}_B$ (called the AB-jobs). For each $E \in \{A', B', AB\}$, if a batch $\mathcal{B}$ consists of the jobs in $\mathcal{J}_E$, we call $\mathcal{B}$ an E-batch. Furthermore, for each $E \in \{A', B', AB\}$, in $O(n_E \log n_E)$ time, we re-index the jobs in $\mathcal{J}_E$ in the SPT order so that $p_{E,1} \le p_{E,2} \le \cdots \le p_{E,n_E}$. Then we call $\sigma = (\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_m)$ an SPT-batch schedule, if for every two jobs $J_{E,i}$ and $J_{E,j}$ with $E \in \{A', B', AB\}$, $i < j$ implies $C_{E,i}(\sigma) \le C_{E,j}(\sigma)$. Applying a similar analysis to that in the proof of Lemma 1 in Brucker et al. (1998), we derive the following result.

Lemma 3.1. *For each Pareto-optimal point of the problem in (2), there exists a corresponding Pareto-optimal schedule that is also an SPT-batch schedule.*

Lemma 3.1 shows that it suffices to consider the SPT-batch schedules of the problem in (2) in the sequel. For ease of exposition, we add a null pure A-job $J_{A',0}$ with the processing time $p_{A',0} = 0$. This implies that $\mathcal{J}_{A'} \neq \emptyset$. We see that, for each Pareto-optimal point of the problem in (2), there exists a corresponding Pareto-optimal SPT-batch schedule σ so that the jobs in $\mathcal{J}_{A'}$ are processed as a single batch with the processing time $p_{A',n_{A'}}$ (since the batch capacity is unbounded), and the jobs in $\mathcal{J}_{AB}$ are processed before $\mathcal{J}_{A'}$. We call such σ an *effective schedule*. The *level* of an effective schedule σ, denoted by $l(\sigma)$, is defined as the maximum index $j \in \{0, 1, \ldots, n_{B'}\}$ such that the first j jobs in $\mathcal{J}_{B'}$ are scheduled before $\mathcal{J}_{A'}$ in σ. We next introduce some useful notation for a given effective schedule σ.

- σ^l is the partial schedule of σ restricted to the first $l(\sigma)$ pure B-jobs and all the AB-jobs. Equivalently, σ^l is the partial schedule of σ before the processing of $\mathcal{J}_{A'}$ in σ.
- $C_{\max}(\sigma^l)$ is the makespan of σ^l.
- $f^B_{\max}(\sigma^l)$ is the maximum cost of σ^l.
- $\overline{f}^B_{\max}(\sigma^l)$ is the maximum cost of the last $(n_{B'} - l(\sigma))$ pure B-jobs in σ. When $l(\sigma) = n_{B'}$, we define $\overline{f}^B_{\max}(\sigma^l) = -\infty$.

Lemma 3.2. *For each Pareto-optimal point of the problem in (2), there exists a corresponding effective schedule σ so that the partial schedule σ^l is a Pareto-optimal schedule for the problem in (1) in which the jobs to be scheduled are the jobs of σ^l.*

Proof. Let σ be an effective schedule corresponding to some given Pareto-optimal point. Since σ^l is a schedule of the first $l(\sigma)$ pure B-jobs and all the AB-jobs with the makespan $C_{\max}(\sigma^l)$ and the maximum cost $f^B_{\max}(\sigma^l)$, there exists a Pareto-optimal schedule π^l for the problem in (1) for the first $l(\sigma)$ pure B-jobs and all the AB-jobs so that $C_{\max}(\pi^l) \leq C_{\max}(\sigma^l)$ and $f_{\max}(\pi^l) \leq f^B_{\max}(\sigma^l)$. Let σ' be the schedule obtained from σ by replacing the partial schedule σ^l with π^l. Then we have $C_{\max}(\sigma'^l) = C_{\max}(\pi^l) \leq C_{\max}(\sigma^l)$ and $f^B_{\max}(\sigma'^l) = f_{\max}(\pi^l) \leq f^B_{\max}(\sigma^l)$. By the regularity of the cost functions of all the pure B-jobs, we conclude that $\overline{f}^B_{\max}(\sigma'^l) \leq \overline{f}^B_{\max}(\sigma^l)$. It follows that $C^A_{\max}(\sigma') = C_{\max}(\sigma'^l) + p_{A',n_{A'}} \leq C_{\max}(\sigma^l) + p_{A',n_{A'}} = C^A_{\max}(\sigma)$ and $f^B_{\max}(\sigma') = \max\{f^B_{\max}(\sigma'^l), \overline{f}^B_{\max}(\sigma'^l)\} \leq \max\{f^B_{\max}(\sigma^l), \overline{f}^B_{\max}(\sigma)\} = f^B_{\max}(\sigma)$. Consequently, σ' is an effective schedule corresponding to the given Pareto-optimal point so that $\sigma'^l = \pi^l$ is a Pareto-optimal schedule. The result follows. □

For each j with $0 \leq j \leq n_{B'}$, we use $P^B_j(C_{\max}, f_{\max})$ to denote the problem in (1), i.e., p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, f_{\max})$, in which the jobs to be scheduled are the first j pure B-jobs and all the AB-jobs (being regarded as two families of jobs), where

$$R = \begin{cases} 2, & \text{if } n_{AB} > 0 \text{ and } j > 0, \\ 1, & \text{otherwise.} \end{cases}$$

Moreover, we use $Q^B_j(t)$ to denote problem p-batch$(+\infty)||f^B_{\max}$ in which the jobs to be scheduled are the last $(n_{B'} - j)$ pure B-jobs, with the restriction that no jobs start earlier than time t. Note that Geng and Yuan (2015b) presented an $O(n^4)$ time algorithm to solve problem p-batch$(+\infty)||f_{\max}$. Thus, problem $Q^B_j(t)$ can be solved by the algorithm presented by Geng and Yuan (2015b) in $O(n^4_B)$ time. We call the algorithm the $f^B_{\max}$-batch algorithm.

If (C, F) is a Pareto-optimal point and σ is a Pareto-optimal schedule corresponding to (C, F), we call $(\sigma; (C, F))$ a *Pareto-optimal pair*. In our discussion, for each j with $0 \leq j \leq n_{B'}$, we use $\overrightarrow{\mathcal{Y}}_j$ to denote a set of Pareto-optimal pairs of problem $P^B_j(C_{\max}, f_{\max})$ that covers all the Pareto-optimal points. Set $\pi_j(t)$ as an optimal schedule for problem $Q^B_j(t)$. We set $\pi_{n_{B'}}(t)$ as a null schedule with $f^B_{\max}(\pi_{n_{B'}}(t)) = -\infty$. Furthermore, when $\mathcal{J}_{AB} = \emptyset$, we set $\overrightarrow{\mathcal{Y}}_0 = \{(\sigma; (C, F))\}$ in which σ is a null schedule with $C = 0$ and $F = -\infty$.

For each j with $0 \leq j \leq n_{B'}$ and any $(\sigma; (C, F)) \in \overrightarrow{\mathcal{Y}}_j$, we define a schedule-point pair

$\psi(\sigma;(C,F)) = (\sigma^*;(C^*,F^*))$ by setting $C^* = C + p_{A',n_{A'}}$, $F^* = \max\{F, f^B_{\max}(\pi_j(C^*))\}$, and $\sigma^* = (\sigma, \mathcal{J}_A, \pi_j(C^*))$. Then we define

$$\overrightarrow{\mathcal{Y}} = \{\psi(\sigma;(C,F)) : (\sigma;(C,F)) \in \overrightarrow{\mathcal{Y}}_j, 0 \le j \le n_{B'}\}. \tag{10}$$

From Lemma 3.2, we have the following result.

Lemma 3.3. *For each Pareto-optimal point (C^*, F^*) of the problem in (2), there is a corresponding effective schedule σ^* so that $(\sigma^*;(C^*,F^*)) \in \overrightarrow{\mathcal{Y}}$.*

From Lemma 3.3, we present the following algorithm to solve the problem in (2).

Pareto(ND-agent): For the problem in (2).

Step 1: For each j with $0 \le j \le n_{B'}$, use Pareto(DP(Q)) to solve problem $P^B_j(C_{\max}, f_{\max})$ and obtain $\overrightarrow{\mathcal{Y}}_j$.

Step 2: For each j with $0 \le j \le n_{B'}$ and any $(\sigma;(C,F)) \in \overrightarrow{\mathcal{Y}}_j$, use the $f^B_{\max}$-batch algorithm to solve problem $Q^B_j(C+p_{A',n_{A'}})$ and obtain an optimal schedule $\pi_j(C+p_{A',n_{A'}})$.

Step 3: Compute the set $\overrightarrow{\mathcal{Y}}$ by (10).

Step 4: Calculate the set $\overrightarrow{\mathcal{Y}}^*$ of non-dominated pairs in $\overrightarrow{\mathcal{Y}}$ and output $\overrightarrow{\mathcal{Y}}^*$.

From Corollary 2.1, Step 1 runs in $O(n_B^8)$ time. From the fact that the $f^B_{\max}$-batching algorithm runs in $O(n_B^4)$ time and the problem in (1) with $R = 2$ or $R = 1$ has at most $O(n_B^3)$ Pareto-optimal points, we conclude that Step 2 runs in $O(n_B^8)$ time. The total running time of Steps 3 and 4 is bounded by $O(n_B^4)$ time. Thus, the time complexity of Pareto(ND-agent) is given by $O(n_B^8)$. Note that we only need to calculate $p_{A',n_{A'}}$ since the jobs in $\mathcal{J}_{A'}$ are processed as a single batch. Thus, the preprocessing stage only needs $O(n_A + n_B \log n_B)$ time. Then we have the following result.

Theorem 3.1. *The problem in (2) is solvable in $O(n_A + n_B^8)$ time.*

Note that the Pareto-scheduling problem p-batch$(+\infty)$|CO-agent, IG|$^{\#}(C^A_{\max}, f^B_{\max})$ is a special version of the problem in (2). Thus, when we use Pareto(ND-agent) to solve problem p-batch$(+\infty)$|CO-agent, IG|$^{\#}(C^A_{\max}, f^B_{\max})$, the time complexity can be reduced. From Corollary 2.1, the problem in (1) with $R = 1$ can be solved in $O(n_B^4)$ time. This

implies that the running time of Step 1 reduces to $O(n_B^5)$. Furthermore, the problem in (1) with $R = 1$ has at most $O(n_B^2)$ Pareto-optimal points. Thus, the running time of Step 2 reduces to $O(n_B^7)$. Then we have the following result.

Corollary 3.1. *The Pareto-scheduling problem p-batch$(+\infty)|$CO-agent, IG$|^{\#}(C^A_{\max}, f^B_{\max})$ is solvable in $O(n_A + n_B^7)$ time.*

When $f_{\max} = L_{\max}$, the problem in (1) can be solved in $O(n^{2R+1})$ time. This implies that the above analysis can be simplified. We only highlight the major differences here.

For each j with $0 \le j \le n_{B'}$ and any $(\sigma; (C, F)) \in \overrightarrow{\mathcal{Y}}_j$, we define a schedule-point pair $\psi(\sigma; (C, F)) = (\sigma^*; (C^*, F^*))$ by setting $C^* = C + p_{A',n_{A'}}$, $F^* = \max\{F, C^* + L^B_{\max}(\pi_j(0))\}$, and $\sigma^* = (\sigma, \mathcal{J}_A, \pi_j(0))$ obtained by gluing σ, $\mathcal{J}_A$, and $\pi_j(0)$ together in that order. Furthermore, Step 2 of Pareto(ND-agent) can be re-written as "For each j with $0 \le j \le n_{B'}$, use the $L^B_{\max}$-batch algorithm to solve problem $Q^B_j(0)$ and obtain an optimal schedule $\pi_j(0)$.". By a similar analysis, we have the following result.

Corollary 3.2. *When $f_{\max} = L_{\max}$, the problem in (2) is solvable in $O(n_A + n_B^6)$ time.*

4 Conclusion

We study the two Pareto-scheduling problems p-batch$(+\infty)|R$-family$|^{\#}(C_{\max}, f_{\max})$ and p-batch$(+\infty)|$ND-agent, IG$|^{\#}(C^A_{\max}, f^B_{\max})$. We summarize in Table 1 the known complexity results related to our research, where α denotes the machine environment p-batch$(+\infty)$.

For further research, we suggest that close attention should be paid to the more complicated problem p-batch$(+\infty)|$ND-agent, IG$|^{\#}(f^A_{\max}, f^B_{\max})$, which has not been addressed as of now even for $(f^A_{\max}, f^B_{\max}) = (L^A_{\max}, L^B_{\max})$.

Acknowledgments

This research was supported in part by NSFC (11671368), NSFC (11771406), and NSFC (11571323), as well as by the Research Grants Council of Hong Kong under grant number PolyU 152207/17E.

Table 1: Summary of complexity results

Scheduling problem	Time complexity	Reference
$\alpha\|\|^{\#}(C_{\max}, L_{\max})$	$O(n^3)$	He et al. (2007)
$\alpha\|\|^{\#}(C_{\max}, f_{\max})$	$O(n^3 \log \sum p_j)$	He et al. (2014)
$\alpha\|\|^{\#}(C_{\max}, f_{\max})$	$O(n^4)$	Geng and Yuan (2015b)
$\alpha\|(r_j, p_j)$-agreeable$\|^{\#}(C_{\max}, L_{\max})$	$O(n^3)$	Gao (2017)
$\alpha\|(r_j, p_j)$-agreeable$\|^{\#}(C_{\max}, f_{\max})$	$O(\min\{n^3 \log(r_n + \sum p_j), n^4\})$	Gao (2017)
$\alpha\|R$-family$\|^{\#}(C_{\max}, L_{\max})$	$O(n^{2R+1})$	Geng and Yuan (2015a)
$\alpha\|R$-family$\|^{\#}(C_{\max}, f_{\max})$	$O(n^{3R+1})$	Theorem 2.2
$\alpha\|$CO-agent, IG$\|^{\#}(C^A_{\max}, L^B_{\max})$	$O(n_A + n_B^4)$	Feng et al. (2013)
$\alpha\|$CO-agent, IG$\|^{\#}(C^A_{\max}, f^B_{\max})$	$O(n_A + n_B^7)$	Corollary 3.1
$\alpha\|$ND-agent, IG$\|^{\#}(C^A_{\max}, f^B_{\max})$	$O(n_A + n_B^8)$	Theorem 3.1
$\alpha\|$ND-agent, IG$\|^{\#}(C^A_{\max}, L^B_{\max})$	$O(n_A + n_B^6)$	Corollary 3.2
$\alpha\|$ND-agent, IG$\|^{\#}(C^A_{\max}, C^B_{\max})$	$O(n)$	An easy observation

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

Agnetis A, Billaut JC, Gawiejnowicz S, Pacciarelli D, Soukhal A (2014) Multiagent Scheduling: Models and Algorithms. Springer, Berlin

Agnetis A, Mirchandani PB, Pacciarelli D, Pacifici A (2004) Scheduling problems with two competing agents. Operations Research 52:229-242

Baker KR, Smith JC (2003) A multiple-criterion model for machine scheduling. Journal of Scheduling 6:7-16

Brucker P, Gladky A, Hoogeveen H, Kovalyov MY, Potts CN, Taut-enhahn T, van de

Velde SL (1998) Scheduling a batching machine. Journal of Scheduling 1:31-54

Feng Q, Yuan JJ, Liu HL, He C (2013) A note on two-agent scheduling on an unbounded parallel-batching machine with makespan and maximum lateness objectives. Applied Mathematical Modelling 37:7071-7076

Gao Y (2017) Min-max scheduling of batch or drop-line jobs under agreeable release and processing. In Submission

Geng ZC, Yuan JJ (2015a) Pareto optimization scheduling of family jobs on a p-batch machine to minimize makespan and maximum lateness. Theoretical Computer Science 570:22-29

Geng ZC, Yuan JJ (2015b) A note on unbounded parallel-batch scheduling. Information Processing Letters 115:969-974

Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5:287-326

He C, Lin H, Yuan JJ, Mu YD (2014) Batching machine scheduling with bicriteria: maximum cost and makespan. Asia-Pacific Journal of Operational Research 31:1450025, http://dx.doi.org/10.1142/S0217595914500250

He C, Lin YX, Yuan JJ (2007) Bicriteria scheduling on a batching machine to minimize maximum lateness and makespan. Theoretical Computer Science 381:234-240

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

Lee CY, Uzsoy R, Martin-Vega LA (1992) Efficient algorithms for scheduling semiconductor burn-in operations. Operations Research 40:764-775

Perez-Gonzalez P, Framinan JM (2014) A common framework and taxonomy for multicriteria scheduling problem with interfering and competing jobs: Multi-agent scheduling problems. European Journal of Operational Research 235:1-16

T'kindt V, Billaut JC (2006) Multicriteria Scheduling: Theory, Models and Algorithms (second ed.). Springer, Berlin

Yuan JJ (2016) Complexities of some problems on multi-agent scheduling on a single machine. Journal of the Operations Research Society of China 4:379-384

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

Yuan JJ (2018) Complexities of four problems on two-agent scheduling. Optimization Letters, DOI 10.1007/s11590-017-1141-x