An improved algorithm on unbounded parallel-batching scheduling to minimize maximum cost and makespan *

Cheng He[a]†, Jing Wu[a], Jinglei Xu[b], Junling Wang[a]

[a]School of Science, Henan University of Technology, Zhengzhou, Henan 450001, China

[b]Faculty of International Studies, Henan Normal University, Xinxiang, Henan 453007, China

Abstract: This paper studies the bicriteria problem of scheduling n jobs on a parallel-batching machine to minimize maximum cost and makespan simultaneously. A parallel-batching machine is a machine that can handle up to b jobs in a batch. The jobs in a batch start and complete respectively at the same time and the processing time of a batch is equal to the largest processing time of jobs in the batch. We consider the unbounded case. For the above bicriteria scheduling problem, we present an $O(n^3)$-time algorithm, which improved the best known $O(n^4)$-time algorithm, and the time complexity is the same as the special case in which maximum cost is maximum lateness. Meanwhile, our algorithm can also solve the single-criterion unbounded parallel-batching scheduling problem to minimize maximum cost in $O(n^3)$ time, which improved the best known $O(n^4)$-time algorithm.

Key word: Bicriteria scheduling, Parallel-batching, Maximum cost, Makespan, Pareto optimal solutions

1 Introduction

Multicriteria scheduling problems have been extensively studied since last two decade (see e.g. [1, 3, 4, 19, 17, 21]), where multi-objective functions may represent different interests of multiple decision-makers. On the other hand, scheduling on batching machines is also a notable direction in scheduling theory (see e.g. [2]). For the batch scheduling, this paper only considers the parallel-batching (rather than serial-batching, denoted s-*batch* in short) model. A parallel-batching machine (denoted p-*batch* in short) is a machine that can handle at most b jobs in a batch, where b is the batch capacity. In particular, the

*This work was supported by NSFHN (No. 232400420003) and STAPHNOS (No. 2020-70) and IFPHNUT (No. 2020ZKCJ08).

†Corresponding author. Email address: hech202@163.com.

unbounded model is denoted by $b \geq n$, and the bounded model is denoted by $b < n$. The jobs in a batch start and complete respectively at the same time and the processing time of a batch is equal to the largest processing time of jobs in the batch. This model is motivated by the applications of semiconductor industry, aircraft industry, steel working industry, glass manufacturing industry and other areas. So it is worthwhile to study the multicriteria scheduling on parallel-batching machines. A parallel-batching machine is generally considered as a bottleneck because of high equipment cost and high energy consumption. Therefore, the machine occupancy time $C_{\max}$ is usually taken into account. On the other hand, assume that each job has a cost with respect to its completion time, but the final decision cost is determined by the maximum cost $f_{\max}$ of all jobs. For example, each job has a storage cost that is decided by its storage specification, and the final storage cost is decided by the highest storage specification. This paper studies the bicriteria scheduling problem of minimizing maximum cost $f_{\max}$ and makespan $C_{\max}$ simultaneously on a parallel-batching machine. Following the three-field notation scheme of Graham et al. [8], our problem is denoted by $1|\text{p-}batch, b \geq n|(f_{\max}, C_{\max})$. The task of the simultaneous optimization is to identify all Pareto optimal points and the Pareto optimal schedule corresponding to each Pareto optimal point.

For the bicriteria scheduling (non-batching) problem, Hoogeveen [18] showed that the problem of minimizing two maximum cost criteria, i.e., $1||(f_{\max}, g_{\max})$, is solvable in $O(n^4)$ time, where $f_{\max}$ and $g_{\max}$ are two general maximum-cost objective functions. For the bicriteria serial-batching scheduling problem, He et al. [12] solved problem $1|\text{s-}batch, b \geq n|(f_{\max}, C_{\max})$ in $O(n^2)$ time. He et al. [15] improved problem $1|\text{s-}batch, b \geq n|(f_{\max}, C_{\max})$ from $O(n^5)$ to time $O(n^3)$. He et al. [11] presented an $O(n^4)$-time algorithm for hierarchical minimization of two maximum costs on a bounded serial-batching machine $1|\text{s-}batch, b < n|Lex(f_{\max}, g_{\max})$. He et al. [10] solved a class of simultaneous optimization scheduling problems with two agents on an unbounded serial-batching machine respectively in polynomial time. For the bicriteria parallel-batching scheduling problem, Feng et al. [6] showed that two-agent scheduling problem $1|\text{p-}batch, b \geq n, inco|(L^A_{\max}, C^B_{\max})$ can be solved in $O(n_B + n_A^4)$ time, where "inco" means that the jobs of two agents A and B cannot be put into a common batch, i.e., the jobs are *incompatible* (short for *inco*) (Otherwise, the jobs are compatible (short for *co*)), n_X is the number of jobs of agent X ($X = A$ or B). For problem $1|\text{p-}batch, b < n, co, p_j^A = p|(L^A_{\max}, C^B_{\max})$ (where $p_j^A = p$ means that all jobs of agent A have equal processing time p), Feng et al. [5] gave an algorithm with $O(n_A n_B n)$ time. He et al. [16] gave an $O(n_B^2 + n_A^3 n_B + n_A n_B \log n_B)$-time algorithm for problem $1|\text{p-}batch, b < n, co, p_j^A = p|(f^A_{\max}, C^B_{\max})$. Moreover, the algorithm can solve problem $1|\text{p-}batch, b < n, co, p_j^A = p|(L^A_{\max}, C^B_{\max})$ in $O(n_B^2 + n_A^2 n_B + n_A n_B \log n_B)$ time, which improved the $O(n_A n_B n)$ time bound of Feng et al. [6]. He et al. [9] showed that hierarchical optimization with two maximum costs on an unbounded parallel-batching machine $1|\text{p-batch}, b \geq n|Lex(f_{\max}, g_{\max})$ can be solved $O(n^4)$ time. He et al. [13] presented an $O(n^3)$-time algorithm for problem $1|\text{p-batch}, b \geq n|(L_{\max}, C_{\max})$. For problem $1|\text{p-batch}, b \geq n|(f_{\max}, C_{\max})$, Geng and Yuan [7] improved our $O(n^3 \log P)$-time algorith-

m (see [14]) to $O(n^4)$-time algorithm, where P is the sum of the processing times of all jobs. In the paper, we show that problem $1|\text{p-batch}, b \geq n|(f_{\max}, C_{\max})$ can be solved in $O(n^3)$ time. Therefore, our algorithm not only improves the known $O(n^4)$-time bound in [7], but also obtains the same time bound as the special case with $f_{\max} = L_{\max}$, i.e., $1|\text{p-batch}, b \geq n|(L_{\max}, C_{\max})$ in [13]. Thus our algorithm removes the gap between our problem and problem $1|\text{p-batch}, b \geq n|(L_{\max}, C_{\max})$. As a by-product, our algorithm can also solve the single-criterion batch scheduling problem $1|\text{p-batch}, b \geq n|f_{\max}$ in $O(n^3)$ time, which improved the known $O(n^4)$-time algorithm in [7].

The paper has two innovations: (1) We judge the total number of iterations by the most moving steps of all jobs in the entire algorithm. (2) In each iteration, we check the "feasibility" of jobs in each batch backwards by the determined static completion time, although the completion time of the current job may have been changed. And each job is checked at most twice, which ensures that the running time in each iteration is at most $O(n)$ time. However, in literature [7], the algorithm checks the "feasibility" of jobs in each batch forwards and the running time of each iteration is at most $O(n^2)$ time.

The paper is organized as follows. In Section 2 we state some preliminaries. Section 3 is dedicated to a polynomial-time algorithm for our problem. Section 4 is focus on computational complexity experiments and results comparison. In Section 5 we give a short summary.

2 Preliminaries

Suppose that we are given a job set $\mathcal{J}$ that contains n independent jobs $J_1, J_2, \cdots, J_n$, i.e., $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$. They are to be scheduled on a single batching machine that is continuously available from time zero onwards and that can handle any number of jobs at the same time, i.e., the batch capacity $b \geq n$. Job J_j has a processing time p_j and a cost function f_j $(j = 1, 2, \cdots, n)$, where $f_j(t)$ denotes the cost incurred if the job is completed at time t, which is a nondecreasing function of t, for $j = 1, 2, \cdots, n$. Given a schedule σ, we use $C_j(\sigma)$ to denote the completion time of job J_j in σ, and $f_j(\sigma) = f_j(C_j(\sigma))$ and $f_{\max}(\sigma) = \max_{j=1}^{n} f_j(C_j(\sigma))$ are defined as the cost of job J_j in σ and the maximum cost of the jobs in σ, respectively. Without loss of generality, we assume that the job parameters are integers.

For problems of minimizing a regular objective function without jobs' release dates, there must be an optimal solution in which the batches are processed contiguously from time zero onwards. So we restrict our attention to the solutions with this property. Thus, a schedule σ is a sequence of batches $\sigma = (B_1, B_2, \cdots, B_r)$, where each batch B_l $(l = 1, 2 \cdots, r)$ is a set of jobs. The processing time of batch B_l is $p(B_l) = \max_{J_j \in B_l}\{p_j\}$ and its completion time is $C(B_l) = \sum_{q=1}^{l} p(B_q)$. Note that the completion time of job J_j in σ is $C_j(\sigma) = C(B_l)$ for each $J_j \in B_l$ and $1 \leq l \leq r$. When there is no ambiguity, we abbreviate $C_j(\sigma)$ to C_j.

• A feasible schedule σ is Pareto optimal with respect to the performance criteria $f_{\max}$ and $C_{\max}$ if there is no feasible schedule π such that both $f_{\max}(\pi) \leq f_{\max}(\sigma)$ and $C_{\max}(\pi) \leq C_{\max}(\sigma)$ hold, where at least one of the inequalities is strict. $(f_{\max}(\sigma), C_{\max}(\sigma))$ is called Pareto optimal point corresponding to Pareto optimal schedule σ.

The following theorem provides a general approach, the so-called ε-constraint approach, for finding Pareto optimal schedules.

Theorem 2.1. ([17]) Let y be the optimal value of constraint problem $\alpha|f \leq \widehat{x}|g$ (where $\widehat{x}$ is a upper bound of f), and let x be the optimal value of the problem $\alpha|g \leq y|f$. Then (x, y) is a Pareto optimal point for $\alpha||(f, g)$.

Suppose that problem $\alpha|g \leq y|f$ is efficiently solvable. Then the other method to find Pareto optimal schedules is unilateral ε-constraint approach in the following.

Unilateral ε-constraint approach ([17]): Let (x^i, y^i) be a previously obtained point. Then solve the problem $\alpha|g < y^i|f$ (or $\alpha|g \leq y^i - 1|f$ if g is integral) and obtain a schedule π^{i+1} which give the next point $(x^{i+1}, y^{i+1}) = (f(\pi^{i+1}), g(\pi^{i+1}))$. Repeat the above process until the problem $\alpha|g < y^i|f$ is infeasible.

Theorem 2.2. ([17]) Let (x^1, y^1) be the first point obtained by Unilateral ε-constraint approach, where y^1 be an upper bound of g. Then all Pareto optimal schedules of problem $\alpha||(f, g)$ can by obtained by Unilateral ε-constraint approach.

3 Minimizing maximum cost and makespan

Assume that the jobs have been re-indexed according to the well-known SPT rule so that $p_1 \leq p_2 \leq \cdots \leq p_n$ (a tie is broken arbitrarily).

Lemma 3.1. [2] With any regular objective function f, there exists an optimal schedule $(B_1, B_2, \cdots, B_r)$ for the scheduling problem $1|p\text{-}batch, b \geq n|f$ such that $B_l = \{J_{i_l}, J_{i_l+1}, \cdots, J_{i_{l+1}-1}\}$ and $1 = i_1 < i_2 < \cdots < i_r < i_{r+1} = n + 1$.

Lemma 3.1 shows that an optimal schedule is specified by the jobs that start each batch, since the complete schedule can be formed by the SPT rule. We refer to such a schedule as *SPT-batch schedule*.

The same property holds for bicriteria scheduling problems.

Lemma 3.2. Let (f, c) be a Pareto optimal point of $1|p\text{-}batch, b \geq n|(f_{\max}, C_{\max})$. Then there exists an optimal SPT-batch schedule corresponding to (f, c).

By Lemma 3.2, we restrict ourselves to the SPT-batch schedules in the following.

Let $\sigma = (B_1, B_2, \cdots, B_r)$ be an SPT-batch schedule. Note that n jobs form at most n batches in any feasible schedule. For convenience, we insert $n - r$ empty batches with processing times 0 before the first batch of schedule σ such that schedule σ contains exactly n batches, denote the new schedule by σ', i.e., $\sigma' = (B'_1, B'_2, \cdots, B'_n)$ with $B'_1 =$

$B'_2 = \cdots = B'_{n-r} = \emptyset$ and $B'_{n-r+i} = B_i$ for $1 \le i \le r$. It is clear that $f_{\max}(\sigma') = f_{\max}(\sigma)$ and $C_{\max}(\sigma') = C_{\max}(\sigma)$. Therefore, the introduction of empty batch has no effect on our problem. Without loss of generality, we can assume that any an SPT-batch schedule $\sigma := (B_1, B_2, \cdots, B_n)$ in the following, where the first $n - r$ batches are empty batches, the last r batches are nonempty batches and $1 \le r \le n$.

Definition 3.1. A family of sets $S = \{S_1, S_2, \cdots, S_n\}$ is *an SPT-partition* of $\mathcal{J}$ means that sets $S_1, S_2, \cdots, S_n$ are an partition of $\mathcal{J}$ (i.e., these sets are disjoint and their union is $\mathcal{J}$, where maybe there is an i $(1 \le i \le n-1)$ such that $S_1 = S_2 = \cdots = S_i = \emptyset$ and $S_k \ne \emptyset$ for $i + 1 \le k \le n)$ and $p_j \le p_k$ for any $J_j \in S_l$ and $J_k \in S_{l+1}$, where $1 \le l \le n - 1$.

By Theorem 2.2, we only need to solve a series of constraint problems $1|p\text{-}batch, b \ge n, f_{\max} \le f|C_{\max}$ (P_f for short) for $\underline{f} \le f \le \overline{f}$ in order to solve our problem $1|p\text{-}batch, b \ge n, |(f_{\max}, C_{\max})$, where $\underline{f}$ and $\overline{f}$ are the lower bound and the upper bound of $f_{\max}$, respectively. We introduce an important definition that generalizes the idea for problem $1|r_j, p_j = p|(L_{\max}, C_{\max})$ in [20].

Definition 3.2. Let $\sigma = (B_1, B_2, \cdots, B_n)$ be any SPT-batch feasible schedule of problem P_f. Then an SPT-partition $S = \{S_1, S_2, \cdots, S_n\}$, with $S_i = \{J_{\eta_{i-1}+1}, J_{\eta_{i-1}+2}, \cdots, J_{\eta_i}\}$ $(1 \le i \le n)$, of $\mathcal{J}$ is called *a Candidate Set Family* (CSF for short) of problem P_f means that $f_{\eta_i}(C(B_l)) > f$ for any $1 \le i \le n$ and $i < l \le n$, i.e., the job J_{η_i} cannot be assigned to any batch whose index is larger than i in any SPT-batch feasible schedule of problem P_f.

Definition 3.3. A SPT-batch schedule $\sigma = (B_1, B_2, \cdots, B_n)$ *obeys a given CSF* $S = \{S_1, S_2, \cdots, S_n\}$ of problem P_f if all jobs in batch B_i come from $\bigcup_{i \le k \le n} S_k$ for $1 \le i \le n$ and the first $n - r$ batches are empty batches and the last r batches are nonempty batches, where $1 \le r \le n$.

For any given integer f and a CSF $S = \{S_1, S_2, \cdots, S_n\}$ of problem P_f, where $S_i = \{J_{\eta_{i-1}+1}, J_{\eta_{i-1}+2}, \cdots, J_{\eta_i}\}$ $(1 \le i \le n)$, let $\Psi(f)$ denote the set composed of the SPT-batch schedules of problem P_f and $\Psi(S, f)$ denote the set composed of the SPT-batch schedules which obey CSF S of problem P_f. From Definition 3.2 and Definition 3.3, we have Lemma 3.3 and Corollary 3.4.

Lemma 3.3. $\Psi(f) \subseteq \Psi(S, f)$.

Proof. Let $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(f)$. If $\sigma \notin \Psi(S, f)$, then there is $J_i \in B_l$ and $J_i \in S_k$ with $k < l$. Suppose that $J_{\eta_k} \in B_m$. Then $m \ge l > k$ by SPT-property and $f_{\eta_k}(C(B_m)) = f_{\eta_k}(\sigma) \le f_{\max}(\sigma) \le f$ by $\sigma \in \Psi(f)$, which contradicts that S is a CSF of problem P_f. So $\sigma \in \Psi(S, f)$, i.e., $\Psi(f) \subseteq \Psi(S, f)$. □

Corollary 3.4. Let $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(S, f)$. Then $p(B_i) \ge p_{\eta_i}$ for $1 \le i \le n$.

Proof. Assuming $B_i = \{J_{\kappa_{i-1}+1}, J_{\kappa_{i-1}+2}, \cdots, J_{\kappa_i}\}$. From $\sigma \in \Psi(S, f)$, we know that σ obeys CSF S of problem P_f. So the jobs in B_i $(1 \le i \le n)$ can only come from set $\bigcup_{i \le k \le n} S_k$. Further, $\kappa_i \ge \eta_i$ by SPT-property. Thus $p(B_i) = p_{\kappa_i} \ge p_{\eta_i}$ for $1 \le i \le n$. □

Let $|\sigma|$ denote the number of the empty batches contained in schedule σ throughout the paper. Let f^q be any given integer and $S^q := \{S_1^q, S_2^q, \cdots, S_n^q\}$ be a CSF of problem P_{f^q}. Now we solve problem P_{f^q} by the following Algorithm, which will be used as a central subroutine when solving problem $1|p-batch, b \geq n|(f_{\max}, C_{\max})$.

Algorithm $\mathcal{A}_{f^q}(S^q)$

Step 1 Let $k := 1$ and $\Gamma^k := \{\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k\} := S^q$, i.e., $\Gamma_i^k = S_i^q$ for $1 \leq i \leq n$. Let $\delta := (B_1, B_2, \cdots, B_n)$ with $B_i = S_i^q$ for $1 \leq i \leq n$ and let $B_i := \{J_{\xi_{i-1}+1}, J_{\xi_{i-1}+2}, \cdots, J_{\xi_i}\}$ for $1 \leq i \leq n$.

Step 2 Let schedule $\sigma^k := (B_1^k, B_2^k, \cdots, B_n^k)$ with $B_i^k := \Gamma_i^k$ for $1 \leq i \leq n$ and $|\sigma^k| := l_k$ (clearly, $|\delta| := l_k$ and $\xi_1 = \cdots = \xi_{l_k} = 0$ and $\xi_{l_{k+1}} \geq 1$). Let $t_i := C(B_i^k)$ for $1 \leq i \leq n$ and $l := n$ and $h := \xi_l = n$.

Step 3 If $f_h(t_l) \leq f^q$, then let $h := h - 1$ and go to Step 4. If $f_h(t_l) > f^q$, then go to Step 5.

Step 4 If $h > \xi_{l-1}$, then go back to Step 3. Otherwise $h = \xi_{l-1}$, let $l := l - 1$. If $l = l_k$, i.e., $h = 0$, then go to Step 6; otherwise $l > l_k$ and go back to Step 3.

Step 5 If $h = \xi_l$, then update δ and let $k := k + 1$ and $\Gamma^k := \{\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k\}$ with $\Gamma_i^k := B_i$ for $1 \leq i \leq n$ and $\sigma^k := \delta$ (maybe $\Gamma^k = \Gamma^{k-1}$ and $\sigma^k = \sigma^{k-1}$) and $k_q := k$ and stop. Otherwise let $B_l := \{J_{h+1}, J_{h+2}, \cdots, J_{\xi_l}\}$ and $l := l - 1$.

(5.1) If $l = l_k$, then let $B_l := \{J_1, J_2, \cdots, J_h\}$ and $\xi_l := h$ and go to Step 6.

(5.2) If $l > l_k$, then let $B_l := B_l \bigcup \{J_{\xi_l+1}, J_{\xi_l+2}, \cdots, J_h\}$ and $\xi_l := h$ and go back to Step 3.

Step 6 Update δ. If $\delta = \sigma^k$ (i.e., $f_{\max}(\sigma^k) \leq f^q$ holds), then we get schedule σ^k, let $\pi_q := \sigma^k$ and $k_q := k$ and stop. Otherwise $\delta \neq \sigma^k$, let $k := k+1$ and $\Gamma^k := \{\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k\}$ with $\Gamma_i^k = B_i$ for $1 \leq i \leq n$ and go back to Step 2.

We illustrate the operation process of Algorithm $\mathcal{A}_{f^q}(S^q)$ by the following example.

Example 1: Let $\mathcal{J} = \{J_1, J_2, \cdots, J_6\}$ with $(p_1, p_2, \cdots, p_6) = (1, 2, 4, 7, 9, 14)$ and $(d_1, d_2, \cdots, d_6) = (1, 1, 4, 11, 17, 24)$, and let $S^q = \{\emptyset, \emptyset, \emptyset, \emptyset, \{J_1, J_2, J_3, J_4\}, \{J_5, J_6\}\}$ and $f_{\max} = L_{\max}$ (i.e., $f_j = L_j = C_j - d_j$) and $f^q = 5$.

The operation process of Algorithm $\mathcal{A}_{f^q}(S^q)$ is as follows. Let $k = 1$, $\Gamma^1 := S^q$, $\delta := (B_1, B_2, \cdots, B_n) = (\emptyset, \emptyset, \emptyset, \emptyset, \{J_1, J_2, J_3, J_4\}, \{J_5, J_6\})$, $\xi_1 = \cdots = \xi_4 = 0$, $\xi_5 = 4$ and $\xi_6 = 6$. By Step 2, we obtained $\sigma^1 = (\emptyset, \emptyset, \emptyset, \emptyset, \{J_1, J_2, J_3, J_4\}, \{J_5, J_6\})$ and $t_1 = \cdots = t_4 = 0$, $t_5 = 7$, $t_6 = 21$ and $l_1 = 4$. Since $L_6 = t_6 - d_6 = -3 \leq f^q$, $L_5 = t_6 - d_5 = 4 \leq f^q$, $L_4 = t_5 - d_4 = -4 \leq f^q$, $L_3 = t_5 - d_3 = -4 \leq f^q$ and $L_2 = t_5 - d_2 = 6 > f^q$, and clearly $h = 2 \neq 4 = \xi_5$, so let $B_5 := \{J_3, J_4\}$ and $l := l - 1 = 4$. By Step (5.1), let $B_4 := \{J_1, J_2\}$ and $\xi_4 = 2$ and go to Step 6. Update $\delta := (\emptyset, \emptyset, \emptyset, \{J_1, J_2\}, \{J_3, J_4\}, \{J_5, J_6\})$. From $\delta \neq \sigma^1$, let $k := k + 1 = 2$ and $\Gamma^2 := \{\emptyset, \emptyset, \emptyset, \{J_1, J_2\}, \{J_3, J_4\}, \{J_5, J_6\}\}$ and go to the next iteration.

Similarly, we obtain $\sigma^2 = (\emptyset, \emptyset, \emptyset, \{J_1, J_2\}, \{J_3, J_4\}, \{J_5, J_6\})$ and $t_1 = t_2 = t_3 = 0$,

$t_4 = 2$, $t_5 = 9$, $t_6 = 23$ and $l_2 = 3$. Since $L_6 = -1 \leq f^q$ and $L_5 = 6 > f^q$ and clearly $h = 5 \neq 6 = \xi_6$, let $B_6 := \{J_6\}$ and $l := l - 1 = 5$ and $B_5 := B_5 \bigcup \{J_5\} = \{J_3, J_4, J_5\}$ and $\xi_5 = 5$. From $L_i = t_5 - d_i \leq f^q$ hold for $i = 5, 4, 3$ and $L_i = t_4 - d_i \leq f^q$ hold for $i = 2, 1$, then go to Step 6. Update $\delta := (\emptyset, \emptyset, \emptyset, \{J_1, J_2\}, \{J_3, J_4, J_5\}, \{J_6\})$. From $\delta \neq \sigma^2$, let $k := k + 1 = 3$ and $\Gamma^3 := \delta$ and go to the next iteration.

Similar to the above, we have $\sigma^3 = (\emptyset, \emptyset, \emptyset, \{J_1, J_2\}, \{J_3, J_4, J_5\}, \{J_6\})$ and $\delta = (\emptyset, \emptyset, \emptyset, \{J_1, J_2, J_3\}, \{J_4, J_5\}, \{J_6\})$. Since $\delta \neq \sigma^3$, let $k := 4$ and $\Gamma^4 := \delta$ and continue the above process, we have $\sigma^4 = (\emptyset, \emptyset, \emptyset, \{J_1, J_2, J_3\}, \{J_4, J_5\}, \{J_6\})$ and $\delta = \sigma^4$. Let $\pi_q := \sigma^4$ and $k_q := 4$ and stop.

Note that we have the assumptions in the following. Let $\Gamma^k = \{\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k\}$ and $\sigma^k = (B_1^k, B_2^k, \cdots, B_n^k)$ be the set obtained at iteration k by Algorithm $\mathcal{A}_{f^q}(S^q)$ and the corresponding schedule, respectively, where $\Gamma_i^k = \{J_{\eta_{i-1}^k+1}, J_{\eta_{i-1}^k+2}, \cdots, J_{\eta_i^k}\}$ $(1 \leq i \leq n)$. Thus we have the following lemmas.

Lemma 3.5. Γ^k is a CSF of problem P_{f^q} for $1 \leq k \leq k_q$.

Proof. If $k = 1$, i.e., $\Gamma^k = S^q$, then the result holds by the known conditions. Assume that the result holds for $k = h < k_q$, i.e., Γ^h is a CSF of problem P_{f^q}. Let $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(f^q)$, then $f_{\eta_i^h}(C(B_l)) > f^q$ for $1 \leq i \leq n$ and $i < l \leq n$ by the definition of CSF of problem P_{f^q}. Let $\sigma^h = (B_1^h, B_2^h, \cdots, B_n^h)$ be the schedule obtained in Algorithm $\mathcal{A}_{f^q}(S^q)$. Then $B_i^h = \Gamma_i^h$ by the definitions of σ^h. So $p(B_i^h) = p_{\eta_i^h}$. From $\sigma \in \Psi(f^q)$ and Lemma 3.3 and Corollary 3.4, we have $p(B_i) \geq p_{\eta_i^h}$ for $1 \leq i \leq n$. Hence, $C(B_i^h) = \sum_{j=1}^{i} p(B_j^h) \leq \sum_{j=1}^{i} p(B_j) = C(B_i)$ for $1 \leq i \leq n$. Suppose that there is a job $J_{\eta_i^{h+1}} \in \Gamma_r^h$ and $J_{\eta_i^{h+1}} \in \Gamma_q^{h+1}$ and $q \neq r$. Then $q \leq r - 1$ and $f_{\eta_i^{h+1}}(C(B_{q+1}^h)) > f^q$ by the running rule of Algorithm $\mathcal{A}_{f^q}(S^q)$. Therefore, $f_{\eta_i^{h+1}}(C(B_{q+1})) \geq f_{\eta_i^{h+1}}(C(B_{q+1}^h)) > f^q$ by $C(B_{q+1}^h) \leq C(B_{q+1})$. Since $\sigma \in \Psi(f^q)$, we have $f_{\max}(\sigma) \leq f^q$. So $C_{\eta_i^{h+1}}(\sigma) < C(B_{q+1})$, i.e., job $J_{\eta_i^{h+1}}$ cannot be assigned to any batch whose index is larger than q in σ. By the arbitraries of σ and job $J_{\eta_i^{h+1}}$ and the definition of CSF of problem P_{f^q}, the result holds for $k = h + 1$. By the above induction assumption, Γ^k is a CSF of problem P_{f^q} for $1 \leq k \leq k_q$. $\square$

From Lemma 3.5, $\Psi(\Gamma^k, f^q)$ are meaningful for $1 \leq k \leq k_q$.

Lemma 3.6. Let $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(\Gamma^k, f^q)$. Then

(1) $C(B_i^k) \leq C(B_i)$ for $i = 1, 2, \cdots, n$;

(2) $C_{\max}(\sigma^k) \leq C_{\max}(\sigma)$, i.e., σ^k has the minimum makespan among all the schedules of $\Psi(\Gamma^k, f^q)$.

Proof. (1) By the definition of σ^k and Corollary 3.4, we have $p(B_i^k) = p_{\eta_i^k}$ and $p(B_i) \geq p_{\eta_i^k}$ for $1 \leq i \leq n$. Hence, $C(B_i^k) = \sum_{j=1}^{i} p(B_j^k) \leq \sum_{j=1}^{i} p(B_j) = C(B_i)$ for $1 \leq i \leq n$.

(2) From 3.6(1), we have $C_{\max}(\sigma^k) = C(B_n^k) \leq C(B_n) = C_{\max}(\sigma)$. $\square$

Theorem 3.7. Let π_q be defined as that in Algorithm $\mathcal{A}_{f^q}(S^q)$ (if exists). Then

(1) $\pi_q \in \Psi(f^q)$ and $C_{\max}(\pi_q) \leq C_{\max}(\sigma)$ for any $\sigma \in \Psi(\Gamma^{k_q}, f^q)$;

(2) π_q is an optimal schedule of problem P_{f^q} (i.e., $1|p-batch, b \geq n, f_{\max} \leq f^q|C_{\max}$).

Proof. (1) From the execution process of Algorithm $\mathcal{A}_{f^q}(S^q)$, we know that $\pi_q = \sigma^{k_q}$ and $f_{\max}(\pi_q) \leq f^q$. So $\pi_q \in \Psi(f^q)$. For any $\sigma \in \Psi(\Gamma^{k_q}, f^q)$, we have $C_{\max}(\pi_q) = C_{\max}(\sigma^{k_q}) \leq C_{\max}(\sigma)$ by Lemma 3.6(2).

(2) For any $\sigma' \in \Psi(f^q)$, we have $C_{\max}(\pi_q) \leq C_{\max}(\sigma')$ by $\Psi(f^q) \subseteq \Psi(\Gamma^{k_q}, f^q)$ and 3.7(1). Therefore, π_q is an optimal schedule of problem P_{f^q}. □

Lemma 3.8. Each iteration of Algorithm $\mathcal{A}_{f^q}(S^q)$ needs $O(n)$ time, i.e., obtaining Γ^k from Γ^{k-1} takes $O(n)$ time for $2 \leq k \leq k_q$. Moreover, the running time of Algorithm $\mathcal{A}_{f^q}(S^q)$ is $O(n^3)$ time.

Proof. Step 1 can be implemented in $O(n)$ time. Steps 2-6 are one iteration, which can be implemented in $O(n)$ time.

The *label* L_j^k of job J_j in $\Gamma^k = \{\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k\}$ is l if and only if $J_j \in \Gamma_l^k$. From the running process of Algorithm $\mathcal{A}_{f^q}(S^q)$, we have $n \geq L_j^1 \geq L_j^2 \geq \cdots \geq L_j^{k_q} \geq j$ for $1 \leq j \leq n$ (all empty batches are consecutively scheduled before all nonempty batches in Γ^{k_q}, so $L_j^{k_q} \geq j$). So job J_j goes through at most $n - j + 1$ sets and at least one job is moved forward in each iteration (except for the last iteration). Thus the total number of the iterations is at most $O(n^2)$. Hence, the overall running time of Algorithm $\mathcal{A}_{f^q}(S^q)$ is $O(n^2 \cdot n + n) = O(n^3)$ time. □

Next, we are ready to describe the main algorithm for constructing the Pareto optimal set $\pi(\mathcal{J})$ for problem $1|p-batch, b \geq n|(f_{\max}, C_{\max})$.

Algorithm PO:

Step 1 Initially, let $f^1 := +\infty$ and $S^1 := \{\emptyset, \cdots, \emptyset, \mathcal{J}\}$. Let $q := 1, f^1 := +\infty$ and $m := 1$ and $\pi(\mathcal{J}) := \emptyset$.

Step 2 Running Algorithm $\mathcal{A}_{f^q}(S^q)$. If Algorithm $\mathcal{A}_{f^q}(S^q)$ gets schedule π_q, then let $q := q + 1$ and $f^q := f_{\max}(\pi_{q-1}) - 1$ and $S^q := \Gamma^{k_{q-1}}$ and go back to Step 2. If Algorithm $\mathcal{A}_{f^q}(S^q)$ stops at Step 5, then let $Q := q$, obtain schedules $\pi_1, \pi_2, \cdots, \pi_{Q-1}$ and go to Step 3.

Step 3 If $m = Q-1$, then let $\pi(\mathcal{J}) := \pi(\mathcal{J}) \bigcup \{\pi_m\}$, obtain $\pi(\mathcal{J})$ and stop. Otherwise $m \leq Q - 2$, if $C_{\max}(\pi_m) < C_{\max}(\pi_{m+1})$, then let $\pi(\mathcal{J}) := \pi(\mathcal{J}) \bigcup \{\pi_m\}$. Let $m := m + 1$ and go back to Step 3.

Lemma 3.9. Suppose that f^q and S^q $(1 \leq q \leq Q)$ come from Algorithm PO. Then S^q is a CSF of problem P_{f^q} for $1 \leq q \leq Q$.

Proof. When $q = 1$, $f^1 := +\infty$ and $S^1 := \{\emptyset, \cdots, \emptyset, \mathcal{J}\}$, the result holds. Assume that the result holds for $q = q' < Q - 1$, i.e., $S^{q'}$ is a CSF of problem $P_{f^{q'}}$. Hence, $\Gamma^{k_{q'}}$ is a CSF of problem $P_{f^{q'}}$ by Lemma 3.5. Let $\pi_{q'}$ be defined as that in Algorithm PO.

Then $f^{q'+1} = f_{\max}(\pi_{q'}) - 1$. If $\Psi(f^{q'+1}) \neq \emptyset$, then let $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(f^{q'+1})$. Then $f_{\max}(\sigma) \leq f^{q'+1}$. So $\sigma \in \Psi(f^{q'})$ by $f^{q'} > f^{q'+1}$. Then $f_{\eta_i^{k_{q'}}}(C(B_l)) > f^{q'}$ for any $1 \leq i \leq n$ and $i < l \leq n$. Further, $f_{\eta_i^{k_{q'}}}(C(B_l)) > f^{q'+1}$ for any $1 \leq i \leq n$ and $i < l \leq n$ by $f^{q'} > f^{q'+1}$, i.e., $\Gamma^{k_{q'}}$ is a CSF of problem $P_{f^{q'+1}}$. Therefore, $S^{q'+1}$ is a CSF of problem $P_{f^{q'+1}}$ by $S^{q'+1} = \Gamma^{k_{q'}}$. By the induction hypothesis, S^q is a CSF of problem P_{f^q} for $1 \leq q \leq Q$. This completes the proof. □

Lemma 3.9 indicates each Algorithm $\mathcal{A}_{f^q}(S^q)$ in Algorithm PO and $\Psi(S^q, f^q)$ are meaningful for $1 \leq q \leq Q$. Note that $Q \geq 2$.

In the following, assume that $\Gamma^k = (\Gamma_1^k, \Gamma_2^k, \cdots, \Gamma_n^k)$ and $\sigma^k = (B_1^k, B_2^k, \cdots, B_n^k)$ $(1 \leq k \leq k_Q)$ come from Algorithm $\mathcal{A}_{f^Q}(S^Q)$. Let $\Gamma_i^k := \{J_{\eta_{i-1}^k+1}, J_{\eta_{i-1}^k+2}, \cdots, J_{\eta_i^k}\}$ and $|\sigma^k| = l_k$ for $1 \leq k \leq k_Q$ and $1 \leq i \leq n$. Note that $k_Q \geq 2$. Then we have the following lemmas.

Lemma 3.10. If $\Psi(f^Q) \neq \emptyset$, then for any $\sigma = (B_1, B_2, \cdots, B_n) \in \Psi(f^Q)$. Then

(1) $\eta_i^{k+1} \geq \eta_i^k$ for $1 \leq k \leq k_Q - 1$ and $1 \leq i \leq n-1$;

(2) For $1 \leq k \leq k_Q$ and $l_k + 1 \leq i \leq n-1$, jobs $J_{\eta_i^k}$ and $J_{\eta_{i+1}^k}$ are scheduled in the different batches in schedule σ.

Proof. (1) The results hold by the running rule of Algorithm $\mathcal{A}_{f^Q}(S^Q)$.

(2) For any given i $(l_k + 1 \leq i \leq n-1)$, we prove the inclusions by induction on k. When $k = 1$, we prove $J_{\eta_i^1}$ and $J_{\eta_{i+1}^1}$ for $l_1 + 1 \leq i \leq n-1$ are scheduled in the different batches in σ. If there is a $l_1 + 1 \leq i \leq n-1$ so that jobs $J_{\eta_i^1}$ and $J_{\eta_{i+1}^1}$ are scheduled in the same batch B_l in σ, then $f_{\eta_i^1}(C(B_l)) = f_{\eta_i^1}(\sigma) \leq f_{\max}(\sigma) \leq f^Q$. Since $\pi_{Q-1} \in \Psi(\Gamma^{k_{Q-1}}, f^{Q-1})$ and $\sigma^1 = \pi_{Q-1}$ and $\Gamma^1 = \Gamma^{k_{Q-1}}$, then $\sigma^1 \in \Psi(\Gamma^1, f^{Q-1})$. Hence, we have $f_{\eta_i^1}(C(B_{i+1}^1)) > f^{Q-1}$. Thus $C(B_l) < C(B_{i+1}^1)$ by $f^Q < f^{Q-1}$. Let $B_l := \{J_x, J_{x+1}, \cdots, J_{\eta_i^1}, \cdots, J_{\eta_{i+1}^1}, \cdots, J_y\}$, where $x \leq \eta_i^1$ and $y \geq \eta_{i+1}^1$. Construct a new schedule $\sigma' = (B_1, B_2, \cdots, B_{l-1}, \{J_x, J_{x+1}, \cdots, J_{\eta_{i+1}^1}\}, B_{i+2}^1, B_{i+3}^1, \cdots, B_n^1)$. Clearly, $C_j(\sigma') \leq C_j(\sigma)$ for $1 \leq j \leq \eta_{i+1}^1$ and $C_j(\sigma') \leq C_j(\sigma^1)$ for $\eta_{i+1}^1 + 1 < j \leq n$ by $p_y \geq p_{\eta_{i+1}^1}$ and $C(B_l) < C(B_{i+1}^1)$. Therefore, $C_{\max}(\sigma') \leq C_{\max}(\sigma^1)$ and $f_{\max}(\sigma') \leq f^{Q-1}$ by $f_{\max}(\sigma) \leq f^Q < f^{Q-1}$ and $f_{\max}(\sigma^1) \leq f^{Q-1}$, which contradicts to the optimality of σ^1 ($\pi_{Q-1} = \sigma^1$ is an optimal schedule of problem $P_{f^{Q-1}}$ by Theorem 3.7(2)). Hence, jobs $J_{\eta_i^1}$ and $J_{\eta_{i+1}^1}$ for $l_1 + 1 \leq i \leq n-1$ cannot be scheduled in the same batch in σ, i.e., the conclusion holds for $k = 1$.

Assume that the conclusions hold for $k \leq m-1 (< k_Q)$. Next, we prove the conclusion holds for $k = m$, i.e., for $l_m + 1 \leq i \leq n-1$, $J_{\eta_i^m}$ and $J_{\eta_{i+1}^m}$ are scheduled in the different batches in σ. If there is a $l_m + 1 \leq i \leq n-1$ so that jobs $J_{\eta_i^m}$ and $J_{\eta_{i+1}^m}$ are scheduled in the same batch in σ, then we prove the result in the following two cases by $\eta_i^m \geq \eta_i^{m-1}$ and $\eta_{i+1}^m \geq \eta_{i+1}^{m-1}$ in 3.10(1):

(i) If $\eta_i^m = \eta_i^{m-1}$ and $\eta_{i+1}^m \geq \eta_{i+1}^{m-1}$, then $J_{\eta_i^{m-1}}$ and $J_{\eta_{i+1}^{m-1}}$ are scheduled in the same batch in σ by $\eta_{i+1}^m \geq \eta_{i+1}^{m-1}$. On the other hand, we know $J_{\eta_i^{m-1}}$ and $J_{\eta_{i+1}^{m-1}}$ are scheduled

in different batches in σ by the above induction hypothesis, a contradiction. So $J_{\eta_i^m}$ and $J_{\eta_{i+1}^m}$ cannot be scheduled in the same batch in σ.

(ii) If $\eta_i^m > \eta_i^{m-1}$ and $\eta_{i+1}^m \geq \eta_{i+1}^{m-1}$, then we have $f_{\eta_i^m}(C(B_{i+1}^{m-1})) > f^Q$ by the rule of Algorithm $\mathcal{A}_{f^Q}(S^Q)$. From the above induction hypothesis, we note that jobs $J_{\eta_{l_{m-1}+1}^{m-1}}, J_{\eta_{l_{m-1}+2}^{m-1}}, \cdots, J_{\eta_{i+1}^{m-1}}$ are scheduled in different batches in σ. So $C_{\eta_{i+1}^{m-1}}(\sigma) \geq p_{\eta_{l_{m-1}+1}^{m-1}} + p_{\eta_{l_{m-1}+2}^{m-1}} + \cdots + p_{\eta_{i+1}^{m-1}} = C(B_{i+1}^{m-1})$. Therefore, $C_{\eta_{i+1}^m}(\sigma) \geq C_{\eta_{i+1}^{m-1}}(\sigma) \geq C(B_{i+1}^{m-1})$ by $\eta_{i+1}^m \geq \eta_{i+1}^{m-1}$ in 3.10(1) and the SPT property of σ. If $J_{\eta_i^m}$ and $J_{\eta_{i+1}^m}$ can be scheduled in the same batch in σ, then $C_{\eta_i^m}(\sigma) = C_{\eta_{i+1}^m}(\sigma) \geq C(B_{i+1}^{m-1})$. So $f_{\eta_i^m}(\sigma) = f_{\eta_i^m}(C_{\eta_i^m}(\sigma)) \geq f_{\eta_i^m}(C(B_{i+1}^{m-1})) > f^Q$. Further, $f_{\max}(\sigma) \geq f_{\eta_i^m}(\sigma) > f^Q$, which contradicts to $\sigma \in \Psi(f^Q)$. Hence, $J_{\eta_i^m}$ and $J_{\eta_{i+1}^m}$ cannot be scheduled in the same batch in σ. From (i) and (ii), the conclusions hold for $k = m$. Therefore, jobs $J_{\eta_i^k}$ and $J_{\eta_{i+1}^k}$ are scheduled in the different batches for $1 \leq k \leq k_Q$ and $l_k + 1 \leq i \leq n-1$ in σ by induction hypothesis. □

Lemma 3.11. When Algorithm $\mathcal{A}_{f^Q}(S^Q)$ stops at Step 5 in Algorithm PO, problem P_{f^Q} is infeasible.

Proof. When Algorithm $\mathcal{A}_{f^Q}(S^Q)$ stops at Step 5 in Algorithm PO, we have $k = k_Q$ and $h = \xi_l = \eta_l^{k_Q}$ and $f_h(C(B_l^{k_Q})) = f_{\eta_l^{k_Q}}(C(B_l^{k_Q})) > f^Q$ (since $C(B_i^{k_Q}) = C(B_i^{k_Q-1})$ for $1 \leq i \leq l$). If problem P_{f^Q} is feasible, then $\Psi(f^Q) \neq \emptyset$. Let $\sigma \in \Psi(f^Q)$, Then $f_{\max}(\sigma) \leq f^Q$. From Lemma 3.10(2), jobs $J_{\eta_1^{k_Q}}, J_{\eta_2^{k_Q}}, \cdots, J_{\eta_l^{k_Q}}$ are scheduled in the different batches in schedule σ. Further, we have $C_{\eta_l^{k_Q}}(\sigma) \geq p_{\eta_1^{k_Q}} + p_{\eta_2^{k_Q}} + \cdots + p_{\eta_l^{k_Q}} = C(B_l^{k_Q})$. So $f^Q \geq f_{\max}(\sigma) \geq f_{\eta_l^{k_Q}}(C_{\eta_l^{k_Q}}(\sigma)) \geq f_{\eta_l^{k_Q}}(C(B_l^{k_Q})) > f^Q$, a contradiction. Therefore, problem P_{f^Q} is infeasible. This completes this proof. □

Theorem 3.12. Algorithm PO can solve problem $1|p-batch, b > +\infty|(f_{\max}, C_{\max})$ in $O(n^3)$ time.

Proof. The set $\pi(\mathcal{J})$ obtained by Algorithm PO is the Pareto optimal set of problem $1|p-batch, b > +\infty|(f_{\max}, C_{\max})$, which is guaranteed by Theorem 2.1 and Theorem 3.7(2) and Lemma 3.9 and Lemma 3.11 and the selection rules of the schedules in $\pi(\mathcal{J})$.

Next, we analyze the complexity of Algorithm PO. During the running procedure of Algorithm PO, we get Γ^{k_q} by Algorithm $\mathcal{A}_{f^q}(S^q)$ for $1 \leq q \leq Q$. And to find the next candidate Pareto optimal point, we do not need to start running Algorithm $\mathcal{A}_{f^q}(S^q)$ from $S^q = S^1$ again, we just need to do it on the basis of the last CSF $\Gamma^{k_{q-1}} = S^q$ of problem $P_{f^{q-1}}$. From Algorithm PO, S^1 is the first input candidate set family, and S^Q is finally obtained through a series of iterations. Similar to the analysis of complexity of Algorithm $\mathcal{A}_{f^q}(S^q)$ in Lemma 3.8, each job J_j goes through at most $n-j+1$ sets and at least one job is moved forward in each iteration (except for the last iteration), the total number of the iterations is at most $O(n^2)$. And each iteration needs $O(n)$ time by Lemma 3.8. Moreover, at the beginning of the algorithm, sorting all jobs in SPT order takes $O(n \log n)$ time. Thus the overall running time of Algorithm PO is $O(n^2 \cdot n + n \log n) = O(n^3)$. □

As a consequence of Theorem 3.12, we have

Corollary 3.13. The single criterion problem $1|p-batch, b>+\infty|f_{\max}$ can also be solved in $O(n^3)$ time.

We illustrate the operation process of Algorithm PO by the following example.

Example 2: Let $\mathcal{J}=\{J_1,J_2,\cdots,J_6\}$ with $(p_1,p_2,\cdots,p_6)=(1,2,4,7,9,14)$ and $(d_1,d_2,\cdots,d_6)=(1,1,4,11,17,24)$, and let $f_{\max}=L_{\max}$.

Let $q=1, f^1=+\infty$ and $S^1=\{\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\mathcal{J}\}$ and $m:=1$ and $\pi(\mathcal{J}):=\emptyset$. Running Algorithm $\mathcal{A}_{f^1}(S^1)$, we get schedule $\pi_1=(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\mathcal{J})$. Let $q:=q+1=2$ and $f^2:=f_{\max}(\pi_1)-1=12$ and $S^2:=\Gamma^{k_1}=S^1$ and go back to Step 2.

Continue running Algorithm $\mathcal{A}_{f^2}(S^2)$, we get $\pi_2:=(\emptyset,\emptyset,\emptyset,\emptyset,\{J_1,J_2\},\{J_3,\cdots,J_6\})$. Let $q:=q+1=3$ and $f^3:=f_{\max}(\pi_2)-1=11$ and $S^3:=\Gamma^{k_2}=\pi_2$ and go back to Step 2. Repeating the above process, we get $\pi_3=(\emptyset,\emptyset,\emptyset,\emptyset,\{J_1,J_2,J_3\},\{J_4,J_5,J_6\})$, $\pi_4=(\emptyset,\emptyset,\emptyset,\emptyset,\{J_1,\cdots,J_4\},\{J_5,J_6\})$, and $\pi_5=(\emptyset,\emptyset,\emptyset,\{J_1,J_2,J_3\},\{J_4,J_5\},\{J_6\})$. Let $q:=6$ and $f^6=2$ and $S^6:=\{\emptyset,\emptyset,\emptyset,\{J_1,J_2,J_3\},\{J_4,J_5\},\{J_6\}\}$. Running Algorithm $\mathcal{A}_{f^6}(S^6)$. Let $k:=1$ and $\Gamma^1:=S^6$ and obtain schedule $\sigma^1=(\emptyset,\emptyset,\emptyset,\{J_1,J_2,J_3\},\{J_4,J_5\},\{J_6\})$, and let $\delta:=S^6=\sigma^1$. Because $C_6(\delta)-d_6=27-24=3>f^6$, i.e., $h=6=\xi_6$. Algorithm $\mathcal{A}_{f^6}(S^6)$ stops at Step 5, then let $Q:=6$, we obtain schedules $\pi_1,\pi_2,\cdots,\pi_5$ and go to Step 3.

From $C_{\max}(\pi_1)<C_{\max}(\pi_2)<C_{\max}(\pi_3)<C_{\max}(\pi_4)<C_{\max}(\pi_5)$, obtain the Pareto optimal set $\pi(\mathcal{J})=\{\pi_1,\pi_2,\pi_3,\pi_4,\pi_5\}$. Therefore, $\pi_1,\pi_2,\pi_3,\pi_4,\pi_5$ are all Pareto optimal schedules and the corresponding Pareto optimal points are $(13,14),(12,16),(7,18),(6,21),(3,27)$, respectively.

4 Computational complexity experiments and results comparison

In the section, we will compare the experimental results of our algorithm and the algorithm in [7] on computational complexity. We coded two algorithms in MATLAB and perform the experiments on a personal computer powered by an AMD Ryzen 5 4600H with Radeon Graphics, 3.00GHz with 16GB RAM operating under Windows 11.

The instances are generated randomly. We vary in the experiments the following three factors: the total number of jobs n, the maximum processing time of jobs $p_{\max}$, and the coefficient of the maximum due date of jobs ρ, i.e., $d_{\max}\leq\rho\Sigma_{j=1}^{n}p_j$. We generated the instances with $n=5,10,20,40,70,100$ jobs. For each job $J_j(j=1,2,...,n)$, we sampled its processing time p_j from the uniform integer distributions $U[1,p_{\max}]$ (i.e., $p_j\sim U[1,p_{\max}]$ and p_j is an integer) with $p_{\max}\in\{60,150\}$, and we sampled the due dates from the uniform integer distributions $U[1,\rho\Sigma_{j=1}^{n}p_j]$ with $\rho\in\{0.2,0.5,0.7\}$. Moreover, we sorted the processing times such that $p_1\leq p_2\leq\cdots\leq p_n$. For each testing case, 50 instances are

Table 1: Average and maximum running times compare

P_{max}	ρ	n	average-time Algorithm PO	average-time Reference [7]	max-time Algorithm PO	max-time Reference [7]
60	0.2	5	7.7860e-06	1.6106e-05	4.9200e-05	5.5300e-05
		10	7.9780e-06	2.8378e-05	5.4300e-05	6.9500e-05
		20	8.7460e-06	5.4866e-05	5.8100e-05	1.0240e-04
		40	9.5420e-06	1.1544e-04	7.2200e-05	1.7360e-04
		70	2.0826e-05	4.2723e-04	9.7400e-05	5.4420e-04
		100	3.0672e-05	6.1187e-04	1.3240e-04	0.0011
60	0.5	5	8.6920e-06	1.7918e-05	6.8500e-05	9.4000e-05
		10	1.1756e-05	3.0610e-05	7.3100e-05	1.0370e-04
		20	1.1852e-05	5.4442e-05	7.7400e-05	1.1090e-04
		40	2.1760e-05	1.8674e-04	9.9000e-05	2.7420e-04
		70	2.7414e-05	4.0517e-04	1.1320e-04	5.2960e-04
		100	3.9040e-05	0.0011	1.9160e-04	0.0020
60	0.7	5	1.5020e-05	2.9216e-05	1.1660e-04	1.4180e-04
		10	2.1870e-05	4.8360e-05	9.7300e-05	1.2250e-04
		20	3.0894e-05	1.6608e-04	1.5700e-04	2.9420e-04
		40	4.6560e-05	2.6638e-04	1.7370e-04	4.6160e-04
		70	6.8518e-05	4.5190e-04	1.6330e-04	7.7500e-04
		100	1.0180e-04	0.0019	2.2400e-04	0.0034
150	0.2	5	1.0886e-05	1.6276e-05	5.6400e-05	5.6600e-05
		10	1.1698e-05	3.1598e-05	7.0800e-05	8.8600e-05
		20	2.9760e-05	9.9906e-05	8.3600e-05	1.5770e-04
		40	3.1554e-05	2.2936e-04	8.4600e-05	4.0240e-04
		70	3.3806e-05	4.1979e-04	1.1190e-04	5.8700e-04
		100	4.6148e-05	0.0010	1.1230e-04	0.0016
150	0.5	5	8.3780e-06	1.6434e-05	5.6000e-05	8.4800e-05
		10	8.6880e-06	2.8022e-05	6.3500e-05	6.9900e-05
		20	8.9080e-06	6.6594e-05	6.9000e-05	1.3260e-04
		40	2.3310e-05	1.8541e-04	7.7300e-05	2.9140e-04
		70	3.7434e-05	5.7291e-04	1.1370e-04	8.5870e-04
		100	7.4082e-05	0.0013	1.7390e-04	0.0017
150	0.7	5	1.1690e-05	1.8250e-05	6.8200e-05	7.5500e-05
		10	2.6842e-05	7.084e-05	7.8900e-05	1.1710e-04
		20	3.0420e-05	1.2458e-04	7.9100e-05	1.9460e-04
		40	3.0512e-05	1.9440e-04	9.6600e-05	2.7420e-04
		70	3.6308e-05	5.1715e-04	1.1650e-04	6.6320e-04
		100	8.6482e-05	0.0013	2.0930e-04	0.0021

randomly generated. Therefore, we generate $6 \times 2 \times 3 \times 50 = 1800$ instances in total.

The average and maximum running times of Algorithm PO in the paper and the algorithm in [7] are showed respectively in Table 1.

The experimental results reveal that our algorithm runs faster than that in [7], which is consistent with the theoretical results.

5 Conclusions

The multicriteria scheduling on batching machines is a significant topic in scheduling theory. The paper studies the bicriteria unbounded parallel-batching problem of minimizing maximum cost and makespan simultaneously, i.e., $1|\text{p-batch}, b \geq n|(f_{\max}, C_{\max})$. For the problem, Geng and Yuan [7] improved our $O(n^3 \log P)$-time algorithm (see [14]) to $O(n^4)$-time algorithm, where P is the sum of the processing times of all jobs. The paper gives an $O(n^3)$-time algorithm, which improved the $O(n^4)$-time bound in [7] and remove this gap between our problem and the special problem $1|\text{p-batch}, b \geq n|(L_{\max}, C_{\max})$ in [13]. As a by-product, our algorithm can also solve the single-criterion problem $1|\text{p-batch}, b \geq n|f_{\max}$ in $O(n^3)$ time, which improved the best known $O(n^4)$-time algorithm in [7].

More models with other criteria remain to further investigate. For example, our work should be generalized to the case of minimizing two maximum costs $f_{\max}$ and $g_{\max}$ simultaneously, i.e., $1|\text{p-batch}, b \geq n|(f_{\max}, g_{\max})$. On the other hand, the serial-batching problem $1|\text{p-batch}, b \geq n|(f_{\max}, g_{\max})$ would be meaningful.

References

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

[2] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, S.L. Van De Velde, Scheduling a batching machine, Journal of Scheduling, **1**(1998), 31-54.

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

[4] P. Brucker, Scheduling Algorithms (third edition), Springer, Berlin 2001.

[5] Q. Feng, W.P. Shang, C.W. Jiao, W.J. Li, Two-Agent Scheduling on a Bounded Parallel-Batching Machine with Makespan and Maximum Lateness Objectives, Journal of the Operations Research Society of China, **8**(2020), 189C196.

[6] Q. Feng, J.J. Yuan, H.L. Liu, C. He, A note on two-agent scheduling on an unbounded parallel-batching machine with makespan and maximum lateness objectives, Applied Mathematical Modelling, **37**(2013), 7071C7076.

[7] Z.C. Geng, J.J. Yuan, A note on unbounded parallel-batch scheduling, Information Processing Letters, **115**(2015), 969-974.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics, **5**(1979), 287-326.

[9] C. He, L. Li, Hierarchical Optimization with two maximum costs on an Unbounded Parallel-Batching Machine, RAIRO-Operations Research, **52**(2018), 55-60.

[10] C. He, S.S. Li, J. Wu, Simultaneous optimization scheduling with two agents on an unbounded serial-batching machine, RAIRO-Operations Research, **55**(2021), 3701-3714.

[11] C. He, H. Lin, L. Li, Hierarchical Minimization of Two Maximum Costs on a Bounded Serial-Batching Machine, RAIRO-Operations Research, **55**(2021), 135C140.

[12] C. He, Y.X. Lin, J.J. Yuan, Bicriteria scheduling of minimizing maximum lateness and makespan on a serial-batching machine, Foundations of Computing and Decision Sciences, **33**(2008) , 369-376.

[13] C. He, Y.X. Lin, J.J. Yuan, Bicriteria Scheduling on a Batching Machine to Minimize Maximum Lateness and Makespan, Theoretical Computer Science, **381**(2007), 234-240.

[14] C. He, H. Lin, J.J. Yuan, Y.D. Mu, Batching machine scheduling with bicriteria: maximum cost and makespan, Asia-Pacific J. of Operational Research, **31:4**(2014), 1-10.

[15] C. He, X.M. Wang, Y.X. Lin, Y.D. Mu, An Improved Algorithm for a Bicriteria Batching Scheduling Problem, RAIRO Operations Research, **47**(2013), 1-8.

[16] C. He, J. Wu, H. Lin, Two-agent bounded parallel-batching scheduling for minimizing maximum cost and makespan, Discrete Optimization, **45**(2022), 100698.

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

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

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

[20] A.A. Lazarev, D.I. Arkhipov, F. Werner, Scheduling jobs with equal processing times on a single machine: minimizing maximum lateness and makespan, Optimization Letters, **11:1**(2017), 165-177.

[21] V. T'kindt, J.-C. Billaut, Multicriteria scheduling problems: a survey, RAIRO-Operations Research, **35**(2001), 143-163.