RAIRO-Oper. Res. 60 (2026) 1475–1485
https://doi.org/10.1051/ro/2026034

RAIRO Operations Research
www.rairo-ro.org

MINIMIZING BICRITERIA SCHEDULING OF TWO-COMPONENT JOBS ON A BOUNDED SERIES-BATCHING MACHINE

Cheng He*, Hao Lin, Yuan Zhang, Xin Chai and Cunchang Gu

Abstract. Minimizing maximum cost and makespan simultaneously on a bounded series-batching machine is considered in the paper, in which each job contains a specific component and a standard component. Specific components are scheduled separately, while standard components are scheduled in batches. Completing a job means that its two components have been completed. Additionally, the two components of a job can be scheduled in any order. We present an $O(n^4)$-time algorithm for the simultaneous optimization scheduling problem. When the maximum cost is the maximum lateness, time complexity of the algorithm is $O(n^3 \log n)$ time.

Mathematics Subject Classification. 90C27, 90B35.

Received July 19, 2025. Accepted April 2, 2026.

1. Introduction

Assume that starting from time zero, there are a bounded series-batching machine available and n jobs $1, 2, \ldots, n$. Each job j contains a specific component and a standard component, and p_j and q_j are their processing times, respectively. Specific components are scheduled separately, while standard components are scheduled in batches, a batch can process at most b standard components and $b < n$ (the unbounded batch capacity is denoted by $b \geq n$). In addition, the processing time of a batch is the sum of processing times of all standard components from the batch, and the standard components from the same batch have the same starting and completion times, and there is a setup time s before processing each batch. At any time, at most one component can be processed on the machine and preemption is prohibited. A job is available if and only if its two components have been completed. The two components of a job can be scheduled in any order. A series-batching machine (denoted by "s-batch") is usually considered a bottleneck due to its high energy consumption and equipment cost. Therefore, makespan $C_{\max}$ is often concerned. Moreover, assume that each job has a cost related 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 depends on its storage specification. Yet, the final storage cost depends on the highest storage scale. The paper studies the bicriteria scheduling to minimize the maximum cost $f_{\max}$ and makespan $C_{\max}$ simultaneously on a bounded series-batching machine. Consulting [1], our problem can be denoted by $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$. Seeking out all the Pareto optimal schedules and their corresponding Pareto optimal points in polynomial time is our goal.

Keywords. Two-component, bounded series-batching, maximum cost, makespan, simultaneous optimization.

School of Mathematics and statistics, Henan University of Technology, Zhengzhou, Henan 450001, P.R. China.
*Corresponding author: hech202@163.com

Baker [2] described that two-component job problems was inspired by the production of components on common equipment, for subsequent assembly into end items. For example, a production process has fabrication stage and assembly stage. Final products are made from the assembly of several components. Gerodimos *et al.* [6] presented a specific background of a two-component job problem, which originates from a particular industrial application in food manufacturing where the final products are made by blending base ingredients according to given recipes. Further, the vast majority of recipes contain only two ingredients. Since the blending operation does not form a bottleneck in the production process, under relatively mild assumptions, the manufacturing of food products can be modelled as a two-component job problem. To summarize, our model originates from manufacturing systems with two stages. In the previous stage, raw materials were processed into sub products, and in the latter stage, these sub products will be assembled into a complete product. For example, substrate production and subsequent filling in computer assembly, the manufacturing of automotive engines, etc. It can be seen that the application scope of our model is very wide.

Multicriteria scheduling has been widely discussed (see *e.g.* [3,4,14,19]) since the 1990s. Furthermore, series-batching scheduling problems are also an important branch in scheduling field (see *e.g.* [4]). For the simultaneous optimization of bicriteria scheduling, Hoogeveen [15] showed that the problem of minimizing two maximum cost criteria has an $O(n^4)$-time algorithm. He *et al.* [7] proved that the series-batching scheduling to minimize makespan and maximum lateness $1|s\text{-}batch, b \geq n|(L_{\max}, C_{\max})$ has an $O(n^2)$-time algorithm. For the corresponding bounded case of the above problem, He *et al.* [9] presented an $O(n^6)$-time algorithm. An $O(n^3)$-time algorithm for problem $1|s\text{-}batch, b \geq n|(f_{\max}, C_{\max})$ is proposed by He *et al.* [8]. Problem $1|s\text{-}batch, b \geq n$ (or $b < n$)$|(f_{\max}, C_{\max})$ is solved in $O(n^4)$ time (referring to [5]). He *et al.* [10,11] solved series-batching hierarchical optimization of two maximum costs, *i.e.*, $1|s\text{-}batch, b \geq n$ (or $b < n$)$|Lex(f_{\max}, g_{\max})$, in $O(n^4)$ time.

The scheduling problem with two-component jobs is proposed in [2]. An $O(n^2)$-time dynamic programming algorithm for problem $1|(p_j, q_j), s\text{-}batch, b \geq n|L_{\max}$ is given by Gerodimos *et al.* [6]. They also proved that $1|(p_j, q_j), s\text{-}batch, b \geq n|\Sigma U_j$ is NP-hard, and provided a pseudo-polynomial time algorithm. For parallel-batching scheduling $1|(p_j, q_j), p\text{-}batch, b \geq n|(f_{\max}, C_{\max})$, He *et al.* [12] offered an $O(n^4)$-time algorithm. For $1|(p_j, q_j), s\text{-}batch, b \geq n|(L_{\max}, C_{\max})$, Li [17] gave an $O(n^2 \log n)$-time algorithm. For $1|(p_j, q_j), s\text{-}batch, b \geq n|(f_{\max}, C_{\max})$, He *et al.* [13] gave an $O(n^4)$-time algorithm. For the bounded case, Li [18] supplied an $O(n^2 \log n)$-time algorithm for $1|(p_j = p, q_j), s\text{-}batch, b < n|(L_{\max}, C_{\max})$, where $p_j = p$ represents that the standard components of all jobs have equal processing time p. For our problem $1|(p_j, q_j), s\text{-}batch, b < n|(f_{\max}, C_{\max})$, an $O(n^4)$-time algorithm is presented. Moreover, this algorithm can solve problem $1|(p_j, q_j), s\text{-}batch, b < n|(L_{\max}, C_{\max})$ in $O(n^3 \log n)$ time.

The paper's structure is as follows. Section 2 states some preliminaries. Section 3 presents a polynomial-time algorithm for our problem.

2. Preliminaries

Assume that job set $\mathcal{J} := \{1, 2, \ldots, n\}$, where $1, 2, \ldots, n$ is n independent jobs. Each job j contains two components: specific component j' and standard component j'', and p_j and q_j are their processing times respectively. Set $\Gamma \subseteq \mathcal{J}$. Then we call $\mu_1(\Gamma) = \{j' : j \in \Gamma\}$ and $\mu_2(\Gamma) = \{j'' : j \in \Gamma\}$ *the sets of specific and standard components* of Γ, respectively. Specific components from $\mu_1(\mathcal{J})$ are scheduled separately, while standard components from $\mu_2(\mathcal{J})$ are scheduled in batches on a bounded series-batching machine. Let batch capacity be b. Then 'bounded' means that the machine can schedule no more than b standard components at the same time (with $b < n$); 'series-batching' means that the standard components from a batch have the same starting and completion times, the processing time of a batch is the sum of processing time of all standard components from the batch, and there is a setup time s before processing each batch. So, completing standard component j'' means that the batch that contains j'' is completed, completing a job means that its two components have been completed. Besides, job j has a due date d_j and a nondecreasing cost function $f_j(t)$, where t denotes the completion time of job j $(j = 1, 2, \ldots, n)$ in a schedule. For a given schedule σ, let $C_{j'}(\sigma)$ and $C_{j''}(\sigma)$ and

$C_j(\sigma)$ be the completion times of specific component j' and standard component j'' and job j in σ, respectively. Then the cost of job j in σ is $f_j(\sigma) = f_j(C_j(\sigma))$, and name $f_{\max}(\sigma) = \max_{j=1}^{n} f_j(\sigma)$ the maximum cost of σ. In addition, $L_j(\sigma) = L_j(C_j(\sigma)) - d_j$ and $L_{\max}(\sigma) = \max_{j=1}^{n} L_j(\sigma)$ are called lateness of job j and the maximum lateness of σ, respectively. For the sake of argument, all job parameters and cost functions are assumed to be integers.

If all jobs arrive at time 0, then the problems to minimize a nondecreasing objective function must have an optimal schedule without idle time. Therefore, only schedules with this attribute are considered. Consequently, a schedule σ is actually a sequence composed of batches and specific components. That is, $\sigma = (A_0, B_1, A_1, B_2, A_2, \ldots, B_r, A_r)$, where $(B_1, B_2, \ldots, B_r)$ is a batch sequence of $\mu_2(\mathcal{J})$ and $(A_0, A_1, \ldots, A_r)$ is a schedule of $\mu_1(\mathcal{J})$, and $B_i \neq \emptyset$ for $1 \leq i \leq r$, and each A_i is a subschedule of $\mu_1(\mathcal{J})$ and there may exist $0 \leq i \leq r$ such that $A_i = \emptyset$. Let $l_0 := \lceil \frac{n}{b} \rceil$. Note that n jobs are separated into at least l_0 batches and at most n batches in any feasible schedule, *i.e.*, $l_0 \leq r \leq n$. The processing times of subschedule A_i and batch B_i are $p(A_i) = \Sigma_{j' \in A_i} p_j$ and $q(B_i) = \Sigma_{j'' \in B_i} q_j$, respectively. Let $T := \Sigma_{j \in J}(p_j + q_j)$. Then the makespan of schedule σ is $C_{\max}(\sigma) = rs + \Sigma_{i=1}^{r} q(B_i) + \Sigma_{i=0}^{r} p(A_i) = rs + T$. In addition, let $C(A_i)$ and $C(B_i)$ be the completion times of A_i and B_i for all i's values, respectively. Then $C(B_i) = C(A_{i-1}) + s + q(B_i)$ and $C(A_i) = C(B_i) + p(A_i)$ for all i's values.

Definition 1. Let σ be a feasible schedule of problem $\alpha|\beta|(f_{\max}, C_{\max})$. If $f_{\max}(\pi) > f_{\max}(\sigma)$ or $C_{\max}(\pi) > C_{\max}(\sigma)$ for any feasible schedule π $(\neq \sigma)$, then σ is a Pareto optimal schedule of $\alpha|\beta|(f_{\max}, C_{\max})$ and $(f_{\max}(\sigma), C_{\max}(\sigma))$ is called Pareto optimal point corresponding to Pareto optimal schedule σ.

Suppose that $\alpha|f \leq x|g$ is solvable. Then all Pareto optimal schedules of $\alpha||(f, g)$ can be found by unilateral ε-constraint approach in the following.

Unilateral ε-constraint approach ([14]): Suppose that π^1 is an optimal schedule for $\alpha|f \leq x|g$ and $(x^1, y^1) := (f(\pi^1), g(\pi^1))$, where x is an upper bound of f. Let (x^i, y^i) be a previous obtained point. Then generate a schedule π^{i+1} by solving $\alpha|f < x^i|g$ (or $\alpha|f \leq x^i - 1|g$ if g is integral) and the next point $(x^{i+1}, y^{i+1}) = (f(\pi^{i+1}), g(\pi^{i+1}))$. Repeat the above process until $\alpha|f < x^i|g$ is infeasible.

Theorem 1 ([14]). *All Pareto optimal schedules of $\alpha||(f, g)$ can be derived by Unilateral ε-constraint approach.*

3. A polynomial-time algorithm

Theorem 1 indicates that solving problem $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$ is equivalent to solving a series of constraint problems $1|(p_j, q_j), \text{s-batch}, b < n, f_{\max} \leq f|C_{\max}$ (Abbreviation P_f) for $\underline{f} \leq f \leq \overline{f}$, where $\overline{f}$ and $\underline{f}$ are upper bound and lower bound of $f_{\max}$, respectively. Without loss of generality, for any feasible schedule σ of P_f, let $\sigma := (A_0, B_1, A_1, B_2, A_2, \ldots, B_r, A_r)$, where $(A_0, A_1, \ldots, A_r)$ is a schedule of $\mu_1(\mathcal{J})$ and $(B_1, B_2, \ldots, B_r)$ is a batch sequence of $\mu_2(\mathcal{J})$ and $l_0 \leq r \leq n$. Let $A_i^{-1} := \{j : j' \in A_i\}$ and $B_i^{-1} := \{j : j'' \in B_i\}$ for $1 \leq i \leq r$. Let $\Gamma \subseteq \mathcal{J}$ and $p(\Gamma) := \Sigma_{j \in \Gamma} p_j$ and $q(\Gamma) := \Sigma_{j \in \Gamma} q_j$.

Lemma 1. *If P_f is feasible, then there is an optimal schedule $\sigma = (A_0, B_1, A_1, \ldots, B_r, A_r)$ so that specific component j' is scheduled after the standard component j'', i.e., $C_j(\sigma) = C_{j'}(\sigma) > C_{j''}(\sigma)$, for any $1 \leq j \leq n$.*

Proof. If job j satisfies $C_j(\sigma) = C_{j''}(\sigma) > C_{j'}(\sigma)$, then let $j' \in A_i$ and $j'' \in B_k$. Thus we have $i < k$. Let σ' come from σ by shifting specific component j' to the beginning of A_k. Then $C_j(\sigma') = C_{j'}(\sigma') = C_{j''}(\sigma)$ and $C_l(\sigma') \leq C_l(\sigma)$ for $l \neq j$. Therefore, $C_{\max}(\sigma') = C_{\max}(\sigma)$ and $f_l(\sigma') \leq f_l(\sigma) \leq f$ for $1 \leq l \leq n$ by the nondecreasing property of all costs, *i.e.*, σ' is also optimal for P_f by the optimality of σ. Set $\sigma := \sigma'$. Finally, a required schedule σ can be derived by repeating the above process, completes the proof. □

From Lemma 1, we have $A_0 = \emptyset$. So we assume that $\sigma := (B_1, A_1, \ldots, B_r, A_r)$ for any feasible schedule. For convenience of explanation, suppose that all jobs are indexed in LPT-rule of standard components, *i.e.*, $q_1 \geq q_2 \geq \cdots \geq q_n$ (a tie is arbitrarily broken), and let $|\Gamma|$ and $|B_i|$ and $|A_i|$ be the numbers of jobs in Γ and standard components in B_i and the specific components in A_i for $1 \leq i \leq r$, respectively.

Lemma 2. *If P_f is feasible, then there exists an optimal schedule $\sigma = (B_1, A_1, \ldots, B_r, A_r)$ that satisfies:*

(1) *Let $1 \le i < k \le r$. Then $f_j(C(B_k)) > f$ for any $j' \in A_i$, i.e., while keeping the feasibility, every specific component is scheduled as far back as possible.*
(2) *For $1 \le i \le r$, let $\Gamma_i = \bigcup_{i \le k \le r} A_k^{-1} \setminus \bigcup_{i+1 \le k \le r} B_k^{-1}$ and $\widetilde{\Gamma}_i$ consist of the $\min\{b, |\Gamma_i|\}$ jobs with the smallest index from Γ_i (i.e., if $s, t \in \Gamma_i$ and $s < t$, then if $t \in \widetilde{\Gamma}_i$, then $s \in \widetilde{\Gamma}_i$. So $\widetilde{\Gamma}_i$ contains $\min\{b, |\Gamma_i|\}$ smallest-numbered jobs among those that are completed after B_i, excluding those for which both components take place after B_i). Then $B_i = \mu_2(\widetilde{\Gamma}_i)$.*

Proof. (1) Assume that σ satisfies Lemma 1. If there is a specific component $j' \in A_i$ and a $k > i$ satisfying $f_j(C(B_k)) \le f$ (where we choose k as large as possible), then let σ' come from σ by moving j' to the beginning of A_k. Then $C_{j'}(\sigma') = C(B_k) > C(A_i) \ge C_{j'}(\sigma) > C_{j''}(\sigma) = C_{j''}(\sigma')$ by Lemma 1 and $C_l(\sigma') \le C_l(\sigma)$ for $l \ne j$. Thus, $C_j(\sigma') = C_{j'}(\sigma') = C(B_k)$. Therefore, $f_j(\sigma') = f_j(C(B_k)) \le f$ and $f_l(\sigma') \le f_l(\sigma) \le f_{\max}(\sigma) \le f$ for $l \ne j$, *i.e.*, σ' is also a feasible schedule for P_f. Hence, σ' is also optimal for P_f by $C_{\max}(\sigma') = C_{\max}(\sigma)$ and the optimality of σ. Set $\sigma := \sigma'$. By repeating the above process, we can finally obtain the required schedule σ.

(2) If (2) is false, then let B_l be the last batch that does not satisfy the definition of batch in (2), *i.e.*, $B_i = \mu_2(\widetilde{\Gamma}_i)$ for $l+1 \le i \le r$ and $B_l \ne \mu_2(\widetilde{\Gamma}_l)$. Without loss of generality, assume that σ be the optimal schedule with the smallest l subscript among all optimal schedules of P_f, *i.e.*, for any optimal schedule $\sigma' = (B'_1, A'_1, \ldots, B'_r, A'_r)$ of P_f, assume that $B'_i = \mu_2(\widetilde{\Gamma}_i)$ for $l' + 1 \le i \le r$ and $B'_{l'} \ne \mu_2(\widetilde{\Gamma}_{l'})$. Then $l \le l'$. From Lemmas 1 and 2(1), we have $B_l \subseteq \mu_2(\Gamma_l)$ and $|B_l| \le \min\{b, |\Gamma_l|\}$. If $|B_l| < \min\{b, |\Gamma_l|\}$, then let $j := \min\{k | k \in \Gamma_l \setminus B_l^{-1}\}$ and $j'' \in B_h$. Then $h < l$ and $C_j(\sigma) = C_{j'}(\sigma) > C(B_l)$ by the definition of j and Lemma 1. Let σ' come from σ by shifting j'' from B_h to B_l. Then $|B_l \bigcup \{j''\}| = |B_l| + 1 < \min\{b, |\Gamma_l|\} + 1 \le b$ and $C_k(\sigma') \le C_k(\sigma)$ for $k \ne j$ and $C_j(\sigma') = C_j(\sigma)$. Therefore, σ' is also optimal for P_f from the optimality of σ. Set $\sigma := \sigma'$. Repeating the above process until $|B_l| = \min\{b, |\Gamma_l|\}$. Hence, $|B_l| = |\widetilde{\Gamma}_l|$.

If $b \ge |\Gamma_l|$, then $|B_l| = |\widetilde{\Gamma}_l| = |\Gamma_l|$ by the definition of $\widetilde{\Gamma}_l$, and we have $B_l = \mu_2(\Gamma_l) = \mu_2(\widetilde{\Gamma}_l)$ by $B_l \subseteq \mu_2(\Gamma_l)$ and the definition of $\widetilde{\Gamma}_l$. So if $B_l \ne \mu_2(\widetilde{\Gamma}_l)$, then $b < |\Gamma_l|$ and $|B_l| = |\widetilde{\Gamma}_l| = b$. Thus, $|B_l^{-1} \setminus \widetilde{\Gamma}_l| = |\widetilde{\Gamma}_l \setminus B_l^{-1}| > 0$. According to the selection rule of the jobs in $\widetilde{\Gamma}_l$, we have $q_u > q_v$ (if $q_u = q_v$, then u and v are considered equivalent) for any $u \in \widetilde{\Gamma}_l \setminus B_l^{-1}$ and $v \in B_l^{-1} \setminus \widetilde{\Gamma}_l$. Let $u'' \in B_h$. Then $h < l$ by the definition of $\widetilde{\Gamma}_l$. Let $\sigma' = (B'_1, A'_1, \ldots, B'_r, A'_r)$ be the schedule obtained by exchanging the positions of u'' and v'' in σ, *i.e.*, $A'_k = A_k$ for $1 \le k \le r$ and $B'_k = B_k$ for $k \ne h, l$ and $B'_h = (B_h \setminus \{u''\}) \bigcup \{v''\}$ and $B'_l = (B_l \setminus \{v''\}) \bigcup \{u''\}$. Then $C_k(\sigma') \le C_k(\sigma)$ for $1 \le k \le n$ by $q_u > q_v$. Hence, σ' is also optimal for P_f and $|(B'_l)^{-1} \setminus \widetilde{\Gamma}_l| = |B_l^{-1} \setminus \widetilde{\Gamma}_l| - 1 < |B_l^{-1} \setminus \widetilde{\Gamma}_l|$. Set $\sigma := \sigma'$. Repeating the above process until $|(B'_l)^{-1} \setminus \widetilde{\Gamma}_l| = 0$, *i.e.*, $B'_l = \mu_1(\widetilde{\Gamma}_l)$. Then σ' is also optimal for P_f and $B'_i = \mu_1(\widetilde{\Gamma}_i)$ for $l \le i \le r$, which contradicts to the definition of σ (with the smallest l). Therefore, (2) is true, *i.e.*, $B_i = \mu_2(\widetilde{\Gamma}_i)$ for $1 \le i \le r$. □

By adjusting Lawler's algorithm in [16], we obtain A_k for $1 \le k \le r$.

Lemma 3. *Let $\sigma = (B_1, A_1, \ldots, B_r, A_r)$ be an optimal schedule of P_f. Then for $1 \le k \le r$, any specific component j' from A_k can be processed at the end of A_k if $f_j(t_k) \le f$, where $t_k = C(A_k)$ and $t_r = rs + T$.*

Proof. By Lemma 1, we have $C_j(\sigma) = C_{j'}(\sigma)$ for any job $1 \le j \le n$ in σ. Assume that $A_k = (k'_1, k'_2, \ldots, k'_{h_k})$ ($1 \le k \le r$). Then $C_{k_{h_k}}(\sigma) = t_k$ and $f_{k_{h_k}}(t_k) \le f$. Let $k'_x \in A_k$ with $f_{k_x}(t_k) \le f$ and $x \ne h_k$. Then updating $A_k := (k'_1, \ldots, k'_{x-1}, k'_{x+1}, \ldots, k'_{h_k}, k'_x)$. After updating A_k, we still have $f_{\max}(\sigma) \le f$ and $C_{\max}(\sigma) = rs + T$, as required. □

Lemma 3 gives a rule for constructing subschedule A_k ($1 \le k \le r$): from back to front, at current time t, arbitrarily select a job j from the current available jobs and schedule j' in the interval $[t - p_j, t]$. In the following, when a feasible schedule $\sigma := (B_1, A_1, \ldots, B_r, A_r)$ of P_f is mentioned, it implies that A_k ($1 \le k \le r$) are determined according to the rule. Clearly, n standard components can form up to n batches and at least $\lceil \frac{n}{b} \rceil$ batches in any feasible schedule. To simplify the discussion, we insert $n - r$ empty batches and $n - r$

empty subschedules before the beginning of σ so that σ contains exactly n subschedules and n batches, write the new schedule as σ', *i.e.*, $\sigma' = (B'_1, A'_1, \ldots, B'_n, A'_n)$ with $A'_1 = \cdots = A'_{n-r} = \emptyset$ and $B'_1 = \cdots = B'_{n-r} = \emptyset$ and $A'_{n-r+k} = A_k$ and $B'_{n-r+k} = B_k$ for $1 \le k \le r$ and $r \ge \lceil \frac{n}{b} \rceil$. And in the following, we agree that setup time $s = 0$ if $B'_k = \emptyset$ for any $1 \le k < n$. Obviously, $C_{\max}(\sigma') = C_{\max}(\sigma)$ and $f_{\max}(\sigma') = f_{\max}(\sigma)$. Hence, our problem has not been affected after the introduction of empty subschedules and empty batches. So in the following, suppose that a feasible schedule of P_f is $\sigma := (B_1, A_1, \ldots, B_n, A_n)$, where $A_1 = \cdots = A_{n-r} = \emptyset$ and $B_1 = \cdots = B_{n-r} = \emptyset$ and $B_k \ne \emptyset$ (there may be $A_k = \emptyset$) for $n - r + 1 \le k \le n$ and $r \ge \lceil \frac{n}{b} \rceil$.

Definition 2. A feasible schedule $\sigma = (B_1, A_1, \ldots, B_n, A_n)$ of P_f is called *adjoint-schedule* if σ satisfies Lemmas 1–3 and σ has no idle time. An optimal schedule of problem P_f is called *the optimal adjoint-schedule* if it is an adjoint-schedule.

From Lemmas 1–3, we have Corollary 1.

Corollary 1. *For any given f, P_f has an optimal adjoint-schedule if P_f is feasible.*

Next, only adjoint-schedules with $\lceil \frac{n}{b} \rceil \le r \le n$ is considered. Let $||\sigma||$ be the number of the nonempty batches contained in σ.

Lemma 4. *Let $\sigma = (B_1, A_1, \ldots, B_n, A_n)$ and $\sigma' = (B'_1, A'_1, \ldots, B'_n, A'_n)$ be optimal adjoint-schedules corresponding to problems P_f and $P_{f'}$ with $f' < f$ respectively, and let $||\sigma|| = r$ and $||\sigma'|| = r'$. Then for $1 \le i \le n$, we have*

(1) $C_{\max}(\sigma') \ge C_{\max}(\sigma)$, *i.e.*, $r' \ge r$.
(2) $t'_i \ge t_i$, *where* $t_i = C(A_i)$ *and* $t'_i = C(A'_i)$.
(3) $\bigcup_{i \le l \le n} (A'_l)^{-1} \subseteq \bigcup_{i \le l \le n} A_l^{-1}$.
(4) $\sum_{i \le l \le n} |B'_l| \le \sum_{i \le l \le n} |B_l|$.
(5) $\sum_{i \le l \le n} q(B'_l) \le \sum_{i \le l \le n} q(B_l)$.

Proof. From $f' < f$, σ' is a feasible schedule for P_f. So $C_{\max}(\sigma') \ge C_{\max}(\sigma)$, *i.e.*, $r's + T \ge rs + T$. So (1) holds. And we have $\bigcup_{n-r+1 \le l \le n} A_l = \mu_1(\mathcal{J})$ and $\bigcup_{n-r+1 \le l \le n} B_l = \mu_2(\mathcal{J})$. Hence, (2)–(5) hold for $1 \le i \le n - r + 1$. Next, We prove that (2)–(5) hold for $n - r + 2 \le i \le n$ by inductive hypothesis about i. From the known conditions and (1), we have $t'_n = C_{\max}(\sigma') \ge C_{\max}(\sigma) = t_n$. It is feasible to schedule A'_n in the interval $[t'_n - p(A'_n), t'_n]$ with respect to $f_{\max} \le f'$, it is also feasible to schedule A'_n in the interval $[t_n - p(A'_n), t_n]$ with respect to $f_{\max} \le f$ by $f' < f$ and $t'_n \ge t_n$ and the definition of σ'. Therefore, we have $(A'_n)^{-1} \subseteq A_n^{-1}$ by Lemma 2(1). And $|B'_n| \le |B_n|$ and $q(B'_n) \le q(B_n)$ by Lemma 2(2). Hence, (2)–(5) hold for $i = n$. Assume that (2)–(5) hold for $i = k + 1 \ge n - r + 3$. Next, we prove that (2)–(5) hold for $i = k$.

From (1) and the above assumption, we have $t'_k - p((A'_k)^{-1} \bigcap \bigcup_{k+1 \le l \le n} A_l^{-1}) = C_{\max}(\sigma') - (n - k)s - \sum_{k+1 \le l \le n} (p(A'_l) + q(B'_l)) - p((A'_k)^{-1} \bigcap \bigcup_{k+1 \le l \le n} A_l^{-1}) = C_{\max}(\sigma') - (n - k)s - (p(\bigcup_{k+1 \le l \le n} A'_l) + p((A'_k)^{-1} \bigcap \bigcup_{k+1 \le l \le n} A_l^{-1}) - q(\bigcup_{k+1 \le l \le n} B'_l) \ge C_{\max}(\sigma) - (n - k)s - p(\bigcup_{k+1 \le l \le n} A_l) - q(\bigcup_{k+1 \le l \le n} B_l) = C_{\max}(\sigma) - (n - k)s - \sum_{k+1 \le l \le n} (p(A_l) + q(B_l)) = t_k$. So (2) holds for $i = k$. Since scheduling A'_k in the interval $[t'_k - p(A'_k), t'_k]$ is feasible with respect to $f_{\max}(\sigma') \le f'$, scheduling subschedule $A'_k|_{(A'_k)^{-1} \setminus \bigcup_{k+1 \le l \le n} A_l^{-1}}$ in the interval $[t_k - p((A'_k)^{-1} \setminus \bigcup_{k+1 \le l \le n} A_l^{-1}), t_k]$ is also feasible with respect to $f_{\max}(\sigma) \le f$ by $t'_k - p((A'_k)^{-1} \bigcap \bigcup_{k+1 \le l \le n} A_l^{-1}) \ge t_k$ and $f > f'$, where notation $A|_B$ represents the schedule obtained by confining A to set B. Thus, $(A'_k)^{-1} \setminus \bigcup_{k+1 \le l \le n} A_l^{-1} \subseteq A_k^{-1}$ by the definition of A_k. Hence, $\bigcup_{k \le l \le n} (A'_l)^{-1} \subseteq \bigcup_{k \le l \le n} A_l^{-1}$ by assumption $\bigcup_{k+1 \le l \le n} (A'_l)^{-1} \subseteq \bigcup_{k+1 \le l \le n} A_l^{-1}$, *i.e.*, (3) holds for $i = k$.

(i) If $|B_k| < b$, then $\bigcup_{k \le l \le n} B_l^{-1} = \bigcup_{k \le l \le n} A_l^{-1}$ by Lemma 2(2). So $\bigcup_{k \le l \le n} (B'_l)^{-1} \subseteq \bigcup_{k \le l \le n} (A'_l)^{-1} \subseteq \bigcup_{k \le l \le n} A_l^{-1} = \bigcup_{k \le l \le n} B_l^{-1}$ by the definition of σ' and (3), *i.e.*, $\bigcup_{k \le l \le n} B'_l \subseteq \bigcup_{k \le l \le n} B_l$. Thus, (4)–(5) hold for $i = k$.

(ii) If $|B_k| = b$, then $|B'_k| \le b = |B_k|$. So $\sum_{k\le l\le n}|B'_l| \le \sum_{k\le l\le n}|B_l|$ by assumption $\sum_{k+1\le l\le n}|B'_l| \le \sum_{k+1\le l\le n}|B_l|$, *i.e.*, (4) holds for $i = k$. Next, we prove that (5) holds for $i = k$.

Let $\widehat{Q}_k := \bigcup_{k\le l\le n} B_l \setminus \bigcup_{k\le l\le n} B'_l$ and $\widehat{Q}'_k := \bigcup_{k\le l\le n} B'_l \setminus \bigcup_{k\le l\le n} B_l$. Then $|\widehat{Q}_k| \ge |\widehat{Q}'_k|$ by (4), and (5) is equivalent to $q(\widehat{Q}_k) \ge q(\widehat{Q}'_k)$. If $\widehat{Q}'_k = \emptyset$, then (5) holds for $i = k$. If $\widehat{Q}'_k \ne \emptyset$, we can define an injective mapping $\varphi : \widehat{Q}'_k \to \widehat{Q}_k$ such that if $\varphi(j'') = u''$, then $q_u \ge q_j$. The method is as follows.

Step 0 Let $t := 1$ and $v := 1$ and $w := 1$ and $l_t := k$ and $Q_w := \widehat{Q}_k$ and $Q'_w := \widehat{Q}'_k$.

Step 1 Let $q_{j_v} = \max_{j''\in Q'_w} q_j$ and $j''_v \in B'_{h_t}$ (if there exists $j'' \in Q'_w$ with $q_j = q_{j_v}$ and $j'' \in B'_{h'}$, then $h_t \ge h'$, *i.e.*, a tie is broken by choosing the standard component with the largest completion time). Then $h_t \ge l_t$ and $j_v \in \bigcup_{h_t\le l\le n}(A'_l)^{-1} \subseteq \bigcup_{h_t\le l\le n} A_l^{-1}$ by Lemma 2(2) and (3), and $j''_v \notin \bigcup_{k\le l\le n} B_l$ by $j''_v \in Q'_w$. So $j_v \in \bigcup_{u\le l\le n} A_l^{-1} \setminus \bigcup_{u+1\le l\le n} B_l^{-1} = \Gamma_u$ for $k \le u \le h_t$. Thus, j_v is candidate job when deciding batch B_l $(k \le l \le h_t)$ by Lemma 2(2). On the other hand, by $j''_v \notin \bigcup_{k\le l\le h_t} B_l$ and Lemma 2(2) and the definition of j_v, we have $|B_l| = b$ for $k \le l \le h_t$ and

$$q_u \ge q_{j_v} \ge q_j \qquad (*)$$

for any $u'' \in \bigcup_{k\le l\le h_t} B_l$ and $j'' \in Q'_w$. Let $x_1 := |Q'_w \bigcap \bigcup_{k\le l\le h_t} B'_l|$ and $x_2 := |Q_w \bigcap \bigcup_{k\le l\le h_t} B_l|$.

Step 2 If $x_1 \le x_2$, then let the jobs in $Q'_w \bigcap \bigcup_{k\le l\le h_t} B'_l$ and x_1 jobs selected randomly from $Q_w \bigcap \bigcup_{k\le l\le h_t} B_l$ form a one-to-one mapping. Then the one-to-one mapping satisfies $q_u \ge q_j$ when $\varphi(j'') = u''$ by $(*)$. Let $w := w + 1$ and Q_w come from Q_{w-1} by deleting the x_1 jobs that have formed the above mapping and $Q'_w := Q'_{w-1} \setminus (Q'_{w-1} \bigcap \bigcup_{k\le l\le h_t} B'_l) = Q'_{w-1} \bigcap \bigcup_{h_t+1\le l\le n} B'_l = \bigcup_{h_t+1\le l\le n} B'_l \setminus \bigcup_{k\le l\le n} B_l$. Clearly, $|Q_w| = |Q_{w-1}| - x_1 \ge |Q'_{w-1}| - x_1 = |Q'_w|$ is still true by $|Q_{w-1}| \ge |Q'_{w-1}|$. If $Q'_w = \emptyset$, then (5) holds for $i = k$, stop. Otherwise, let $v := v + 1$ and $t := t + 1$ and $l_t := h_{t-1} + 1$ and return to Step 1.

Step 3 If $x_1 > x_2$, then let $H := (\bigcup_{l=k}^{h_t} B_l) \bigcap (\bigcup_{l=h_t+1}^{n} B'_l)$. From $B_h \bigcap B_l = \emptyset$ and $B'_h \bigcap B'_l = \emptyset$ for any $1 \le h < l \le n$, we have $x_1 = |Q'_k \bigcap \bigcup_{l=k}^{h_t} B'_l| = |(\bigcup_{l=k}^{h_t} B'_l) \setminus (\bigcup_{l=k}^{n} B_l)| = |\bigcup_{l=k}^{h_t} B'_l| - |(\bigcup_{l=k}^{h_t} B'_l) \bigcap (\bigcup_{l=k}^{h_t} B_l)| - |(\bigcup_{l=k}^{h_t} B'_l) \bigcap (\bigcup_{h_t+1}^{n} B_l)|$ and $x_2 := |Q_k \bigcap \bigcup_{l=k}^{h_t} B_l| = |(\bigcup_{l=k}^{h_t} B_l) \setminus (\bigcup_{k\le l\le n} B'_l)| = |\bigcup_{l=k}^{h_t} B_l| - |(\bigcup_{l=k}^{h_t} B_l) \bigcap (\bigcup_{l=k}^{h_t} B'_l)| - |(\bigcup_{l=k}^{h_t} B_l) \bigcap (\bigcup_{h_t+1}^{n} B'_l)|$, we have

$$\begin{aligned} x_1 - x_2 &= \left|\bigcup_{l=k}^{h_t} B'_l\right| - \left|\left(\bigcup_{l=k}^{h_t} B'_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B_l\right)\right| - \left|\bigcup_{l=k}^{h_t} B_l\right| + \left|\left(\bigcup_{l=k}^{h_t} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| \\ &\le \left|\left(\bigcup_{l=k}^{n} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| - \left|\left(\bigcup_{l=k}^{n} B'_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B_l\right)\right| \\ &\quad (\text{by } |B_l| = b \text{ and } |B'_l| \le b \text{ for } k \le l \le h_t). \end{aligned}$$

Hence,

$$\begin{aligned} |H| &= \left|\left(\bigcup_{l=k}^{h_t} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| \\ &= \left|\left[\left(\bigcup_{l=k}^{n} B_l\right) \setminus \left(\bigcup_{l=h_t+1}^{n} B_l\right)\right] \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| \quad \left(\text{by } \left(\bigcup_{l=k}^{h_t} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B_l\right) = \emptyset\right) \\ &= \left|\left(\bigcup_{l=k}^{n} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| - \left|\left(\bigcup_{l=h_t+1}^{n} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| \\ &\quad \left(\text{by } \left(\bigcup_{l=k}^{h_t} B_l\right) \bigcap \left(\bigcup_{l=h_t+1}^{n} B_l\right) = \emptyset\right) \end{aligned}$$

$$\geq \left|\left(\bigcup_{l=k}^{n} B_l\right)\bigcap\left(\bigcup_{l=h_t+1}^{n} B'_l\right)\right| - \left|\left(\bigcup_{l=h_t+1}^{n} B_l\right)\bigcap\left(\bigcup_{l=k}^{n} B'_l\right)\right|$$
$$\geq x_1 - x_2 \quad (\text{by } h_t \geq k).$$

It indicates that there exist at least $x_1 - x_2$ jobs in $\bigcup_{k\leq l\leq h_t} B_l$ that belong to $\bigcup_{h_t+1\leq l\leq n} B'_l$. Let $C_{x''}(\sigma') = \max\{C_{j''}(\sigma') : j'' \in H\}$ and $x'' \in B_m \bigcap B'_h$, and let $t := t+1$ and $h_t := h$. Then $k \leq m \leq h_{t-1} < h_{t-1}+1 \leq h_t \leq n$ by the definitions of H and x. Similar to the previous proof, we infer that x is candidate job when deciding batch B_l $(m \leq l \leq h_t)$ by Lemma 2(2). On the other hand, by $x'' \notin \bigcup_{m+1\leq l\leq h_t} B_l$ and Lemma 2(2) and the definition of x, we have $|B_l| = b$ for $m+1 \leq l \leq h_t$ and $q_u \geq q_x$ for any $u'' \in \bigcup_{m+1<l\leq h_t} B_l$ by the definition of x and (3) and Lemma 2(2). In addition, we have $q_x > q_{j_v}$ by $x'' \in B_m$ and $k \leq m \leq h_{t-1}$ and $(*)$. So

$$q_u \geq q_{j_v} \tag{**}$$

for any $u'' \in \bigcup_{m+1<l\leq h_t} B_l$. From $(*)$ and $(**)$ and the definition of j_v, we have

$$q_u \geq q_{j_v} \geq q_j \tag{***}$$

for any $u'' \in \bigcup_{k\leq l\leq h_t} B_l$ and $j'' \in Q'_w \bigcap \bigcup_{k\leq l\leq h_t} B'_l$. Therefore,

$$q_u \geq q_{j_v} \geq q_j \tag{****}$$

for any $u'' \in Q_w \bigcap \bigcup_{k\leq l\leq h_t} B_l$ and $j'' \in Q'_w \bigcap \bigcup_{k\leq l\leq h_t} B'_l$. Updating $x_1 := |Q'_w \bigcap \bigcup_{k\leq l\leq h_t} B'_l|$ and $x_2 := |Q_w \bigcap \bigcup_{k\leq l\leq h_t} B_l|$ and go back to Step 2.

Whether it's $x_1 \leq x_2$ or $x_1 > x_2$, we have $h_t < h_{t+1}$ (if exists, because when $t = 1$, the process might stop). From $|Q_k| \geq |Q'_k|$ and $h_t < h_{t+1}$ and $|Q'_k| > |Q'_{k+1}|$, we can infer that $Q'_k = \emptyset$ in Step 2 can be achieved after Steps 1–3 are repeated at most $n-k+1$ times. When $Q'_k = \emptyset$ occurs, for each $j'' \in \widehat{Q}'_k$, we have found $u'' \in \widehat{Q}_k$ with $q_u \geq q_j$ so that $\varphi(j'') = u''$. And for any $j'', h'' \in \widehat{Q}'_k$, if $j \neq h$, then $\varphi(j'') \neq \varphi(h'')$. Therefore, the required injective mapping $\varphi : \widehat{Q}'_k \to \widehat{Q}_k$ is derived. Further, we have $q(\widehat{Q}_k) \geq q(\widehat{Q}'_k)$. So $\sum_{k\leq l\leq n} q(B_l) - \sum_{k\leq l\leq n} q(B'_l) = q(\widehat{Q}_k) - q(\widehat{Q}'_k) \geq 0$ by the definitions of $\widehat{Q}'_k$ and $\widehat{Q}_k$, *i.e.*, (5) holds for $i = k$.

Above all, (2)–(5) hold for $i = k$. By inductive hypothesis about i, we have (2)–(5) hold. It completes the proof. □

Let $\pi(\mathcal{J})$ denote the Pareto optimal set of $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$. From Lemmas 2–4 and Corollary 1, we obtain the following Pareto optimal algorithm for $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$.

Algorithm PO

Step 1 Initially, let $f^0 := +\infty$ and $k := 0$ and $\sigma^k := (B_1^k, A_1^k, \ldots, B_n^k, A_n^k)$ with $||\sigma^k|| = l_0 = \lceil \frac{n}{b} \rceil$, where A_n^k is any schedule of $\mu_1(\mathcal{J})$ and $A_i^k := \emptyset$ for $1 \leq i \leq n-1$, and B_i^k is decided according to Lemma 2(2) for $1 \leq i \leq n$. Let $h := 1$ and $\pi_h := \sigma^k$ and $t := C_{\max}(\sigma^k) = l_k s + T$ and $k := k+1$ and $f^h := f_{\max}(\pi_h) - 1$.

Step 2 Let $A_i^k := \emptyset$ and $B_i^k := \emptyset$ for $1 \leq i \leq n$, and let Γ^k be a list with n empty positions and $l := n$.

Step 3 If $(A_l^{k-1})^{-1} \neq \emptyset$, then go to Step 4. Otherwise go to Step 5.

Step 4 If there is $j \in (A_l^{k-1})^{-1}$ such that $f_j(t) \leq f^h$, then set $(A_l^{k-1})^{-1} := (A_l^{k-1})^{-1} \backslash \{j\}$ and $A_l^k = (j', A_l^k)$ and $t := t - p_j$, and updating Γ^k by inserting job j into Γ^k according to LPT-rule of standard components and go back to Step 3. Otherwise, $f_j(t) > f^h$ for all $j \in (A_l^{k-1})^{-1}$. If $l = 1$, then there is no feasible schedule with $l_{k-1}(= n)$ nonempty batches for problem P_{f^h}, go to Step 7. If $l > 1$, then let $A_{l-1}^{k-1} := (A_{l-1}^{k-1}, A_l^{k-1})$.

Step 5 If $\Gamma^k = \emptyset$, then there is no feasible schedule with at least l_{k-1} nonempty batches for P_{f^h}, go to Step 7; if $\Gamma^k \neq \emptyset$, then let $\widetilde{\Gamma}_l$ be the job set consisting of the first $\min\{b, |\Gamma^k|\}$ jobs in Γ^k and $B_l^k := \mu_2(\widetilde{\Gamma}_l)$, and let $t := t - q(B_l^k) - s$ and $\Gamma^k := \Gamma^k \setminus \widetilde{\Gamma}_l$ and $l := l - 1$.

Step 6 If $l \geq n - l_{k-1} + 1$, then go back to Step 3. Otherwise $l = n - l_{k-1}$.

(6.1) If $t = 0$, then we get a feasible schedule $\sigma^k := (B_1^k, A_1^k, \ldots, B_n^k, A_n^k)$ (with $||\sigma^k|| = l_{k-1}$) for problem P_{f^h}. Let $h := h + 1$ and $\pi_h := \sigma^k$ and $l_k := l_{k-1}$ and $t := C_{\max}(\sigma^k) = l_k s + T$ and $k := k + 1$ and $f^h := f_{\max}(\pi_h) - 1$ and go back to Step 2.

(6.2) If $t > 0$ (which implies $l \geq 1$ due to the continuity of nonempty batch), then there is no feasible schedule with l_{k-1} nonempty batches for problem P_{f^h}. Let $A_l^k := A_l^{k-1}$ and $\sigma^k := (B_1^k, A_1^k, \ldots, B_n^k, A_n^k)$ (this serves as the foundation for the next iteration, though it may not be feasible for problem P_{f^h}), where $B_1^k, \ldots, B_l^k$ is determined by $(A_l^k)^{-1} \bigcup \Gamma^k$ from Lemma 2(2). Let $l_k := l_{k-1} + \lceil \frac{|A_l^k| + |\Gamma^k|}{b} \rceil$ (so $\|\sigma^k\| = l_k$) and $t := l_k s + T$ and $k := k + 1$ and go back to Step 2.

Step 7 Obtain all Pareto optimal candidate schedules $\pi_1, \pi_2, \ldots, \pi_h$ (namely, the set $\{\pi_1, \pi_2, \ldots, \pi_h\}$ contains all Pareto optimal schedules). Let $\pi(\mathcal{J}) := \emptyset$ and $v := 1$.

Step 8 If $v = h$, then set $\pi(\mathcal{J}) := \pi(\mathcal{J}) \bigcup \{\pi_v\}$, stop. Otherwise $v \leq h - 1$, if $C_{\max}(\pi_v) < C_{\max}(\pi_{v+1})$, then set $\pi(\mathcal{J}) := \pi(\mathcal{J}) \bigcup \{\pi_v\}$ and $v := v + 1$, go back to Step 8.

The running process of Algorithm PO is illustrated by the following example.

Example Suppose that the setup time $s = 1$ and $b = 2$ and $f_{\max} = L_{\max}$. There are 6 jobs with due dates and processing times as follows:

i	1	2	3	4	5	6
q_i	5	4	3	3	2	1
p_i	3	2	2	1	5	5
d_i	38	14	7	6	27	24

Let $f^0 := +\infty$ and $k := 0$ and

$$\sigma^0 := (\emptyset, \ldots, \emptyset, \{5'', 6''\}, \emptyset, \{3'', 4''\}, \emptyset, \{1'', 2''\}, (4', 3', 2', 6', 5', 1')),$$

where $A_i^0 = \emptyset$ for $1 \leq i \leq 5$ and $A_6^0 = (4', 3', 2', 6', 5', 1')$. So $||\sigma^0|| = l_0 = 3$ and $(f_{\max}(\sigma^0), C_{\max}(\sigma^0)) = (17, 39)$ by $T = 36$. Let $h := 1$ and $\pi_1 := \sigma^0$ and $t := C_{\max}(\sigma^0) = 39$ and $k := 1$ and $f^1 := f_{\max}(\sigma^0) - 1 = 16$.

Let $A_i^1 := \emptyset$ and $B_i^1 := \emptyset$ for $1 \leq i \leq 6$ and $l := 6$ and Γ^1 be an empty list. Since $A_6^0 \neq \emptyset$, go to Step 4. From $f_1(t) = t - d_1 = 1 < f^1$, let $A_6^0 = (4', 3', 2', 6', 5')$ and $A_6^1 := (1')$ and $t = 39 - p_1 = 36$ and $\Gamma^1 := (1)$. Since $f_5(t) = t - d_5 = 9 < f^1$, let $A_6^0 = (4', 3', 2', 6')$ and $A_6^1 := (5', 1')$ and $t = 36 - p_5 = 31$ and $\Gamma^1 := (1, 5)$. Since $f_6(t) = 7 < f^1$, let $A_6^0 = (4', 3', 2')$ and $A_6^1 := (6', 5', 1')$ and $t = 26$ and $\Gamma^1 := (1, 5, 6)$. Since $f_2(t) = 12 < f^1$, let $A_6^0 = (4', 3')$ and $A_6^1 := (2', 6', 5', 1')$ and $t = 24$ and $\Gamma^1 := (1, 2, 5, 6)$. From $f_3(t) > f^1$ and $f_4(t) > f^1$ and $l > 1$, let $A_5^0 = (4', 3')$ and go to Step 5.

Let $B_6^1 := \{1'', 2''\}$ and $t = 24 - q_1 - q_2 - s = 14$ and $\Gamma^1 := (5, 6)$ and $l := l - 1 = 5$. Since $l \geq 6 - 3 + 1 = 4$, go back to Step 3. Similar to the aforementioned process, $A_5^1 = (4', 3')$ and $\Gamma^1 := (3, 4, 5, 6)$. So $B_5^1 := \{3'', 4''\}$ and $B_4^1 := \{5'', 6''\}$ and $t = 0$ and obtain a feasible schedule

$$\sigma^1 := (\emptyset, \ldots, \emptyset, \{5'', 6''\}, \emptyset, \{3'', 4''\}, (4', 3'), \{1'', 2''\}, (2', 6', 5', 1'))$$

with $(f_{\max}(\sigma^1), C_{\max}(\sigma^1)) = (12, 39)$. Let $h := h + 1 = 2$ and $\pi_2 := \sigma^1$ and $l_1 := 3$ and $t := C_{\max}(\sigma^1) = 39$ and $k := 2$ and $f^2 := 11$ and go back to Step 2.

Continue to run the aforementioned process, we obtain $A_6^2 := (6', 5', 1')$ and $B_6^2 := \{1'', 5''\}$ and $t = 18$ and $\Gamma^2 := (6)$ and $A_5^1 = (4', 3', 2')$. Hence, $A_5^2 := (4', 3', 2')$ and $\Gamma^2 := (2, 3, 4, 6)$ and $B_5^2 := \{2'', 3''\}$ and $B_5^2 := \{4'', 6''\}$ and $t = 0$. Therefore, obtain a feasible schedule

$$\sigma^2 := (\emptyset, \ldots, \emptyset, \{4'', 6''\}, \emptyset, \{2'', 3''\}, (4', 3', 2'), \{1'', 5''\}, (6', 5', 1'))$$

with $(f_{\max}(\sigma^2), C_{\max}(\sigma^2)) = (9, 39)$. Let $h := 3$ and $\pi_3 := \sigma^2$ and $l_2 := 3$ and $t := C_{\max}(\sigma^2) = 39$ and $k := 3$ and $f^3 := 8$ and go back to Step 2.

Repeat the above process, obtain $A_6^3 := (1')$ and $B_6^3 := \{1''\}$ and $t = 30$ and Γ^2 is an empty list and $A_5^2 = (4', 3', 2', 6', 5',)$. So $A_5^3 = (2', 6', 5',)$ and $B_5^3 := \{2'', 5''\}$ and $t = 11$ and $\Gamma^3 = (6)$ and $A_4^2 = (4', 3')$. Thus, $A_4^3 = (4', 3')$ and $B_4^3 := \{3'', 4''\}$ and $l = 3 = n - l_2$ and $t = 2 > 0$. So there is no feasible schedule with 3 nonempty batches for problem P_{f^3}. Let $\sigma^3 := (\emptyset, \cdots, \emptyset, \{6''\}, \emptyset, \{3'', 4''\}, (4', 3'), \{2'', 5''\}, (2', 6', 5'), \{1''\}, (1'))$. Let $l_3 := 4$ and $t := 40$ and $k := 4$ and $f^4 := f^3 = 8$ and go back to Step 2.

Repeat the above process, we obtain $A_6^4 := (1')$ and $B_6^4 := \{1''\}$ and $t = 31$ and Γ^2 is an empty list and $A_5^3 = (2', 6', 5',)$. So $A_5^4 = (2', 6', 5',)$ and $B_5^4 := \{2'', 5''\}$ and $t = 12$ and $\Gamma^4 = (6)$ and $A_4^3 = (4', 3')$. Thus, $A_4^4 = (4', 3')$ and $B_4^4 := \{3'', 4''\}$ and $B_3^4 := \{6''\}$. So obtain a feasible schedule

$$\sigma^4 = (\emptyset, \emptyset, \emptyset, \emptyset, \{6''\}, \emptyset, \{3'', 4''\}, (4', 3'), \{2'', 5''\}, (2', 6', 5'), \{1''\}, (1'))$$

with $(f_{\max}(\sigma^4), C_{\max}(\sigma^4)) = (7, 40)$. Let $h := 4$ and $\pi_4 := \sigma^4$ and $l_4 := 4$ and $t := C_{\max}(\sigma^4) = 40$ and $k := 5$ and $f^5 := 6$ and go back to Step 2.

Similarly, we obtain a feasible schedule

$$\sigma^5 = (\emptyset, \emptyset, \emptyset, \emptyset, \{3'', 4''\}, (4', 3'), \{2''\}, (2',), \{5'', 6''\}, (6', 5'), \{1''\}, (1'))$$

with $(f_{\max}(\sigma^4), C_{\max}(\sigma^4)) = (4, 40)$. Let $h := 5$ and $\pi_5 := \sigma^5$ and $l_4 := 4$ and $t := C_{\max}(\sigma^5) = 40$ and $k := 6$ and $f^6 := 3$ and go back to Step 2.

Continue to run the aforementioned process, we receive that there is no feasible schedule with at least 4 nonempty batches for problem P_{f^6}, go to Step 7.

Since $C_{\max}(\pi_1) = C_{\max}(\pi_2) = C_{\max}(\pi_3) = 39 < 40 = C_{\max}(\pi_4) = C_{\max}(\pi_5)$, we have $\pi(\mathcal{J}) = \{\pi_3, \pi_5\}$. Hence, π_3 and π_5 are all Pareto optimal schedules, the corresponding Pareto optimal points are $(9, 39)$ and $(4, 40)$, respectively.

Lemma 5. *Let π_i $(1 \le i \le h)$ come from Algorithm PO. Then π_i is the optimal adjoint-schedule of $P_{f^{i-1}}$.*

Proof. π_i is a feasible adjoint-schedule of $P_{f^{i-1}}$ for $1 \le i \le h$, derived from the running process of Algorithm PO. Next, we prove the result by inductive hypothesis about i.

Clearly, π_1 (which contains $l_0 = \lceil \frac{n}{b} \rceil$ nonempty batches) is the optimal adjoint-schedule of problem P_{f^0}. Assume that $\pi_k := \sigma^l = (B_1^l, A_1^l, \ldots, B_n^l, A_n^l)$ is the optimal adjoint-schedule of problem $P_{f^{k-1}}$. We are going to prove that π_{k+1} is the optimal adjoint-schedule of problem P_{f^k}. If the result is false, let $\pi_{k+1} := \sigma^{l'} = (B_1^{l'}, A_1^{l'}, \ldots, B_n^{l'}, A_n^{l'})$ (with $l < l'$), and assume that $\sigma := (B_1, A_1, \ldots, B_n, A_n)$ is the optimal adjoint-schedule of P_{f^k}. By inductive hypothesis and Lemma 4 and $f^k < f^{k-1}$, for $1 \le i \le n$, we have

(1) $C(A_i^l) \le C(A_i)$.
(2) $\bigcup_{i \le g \le n} A_g \subseteq \bigcup_{i \le g \le n} A_g^l$.

From the running process of Algorithm PO, we obtain schedules $\sigma^{l+1}, \sigma^{l+2}, \ldots, \sigma^{l'}$ $(= \pi_{k+1})$ based on σ^l $(= \pi_k)$ with $f_{\max} \le f^k$ kept unchanged. From (1)–(2) and the running rule of Algorithm PO and the definition of σ, we have $C(A_i^{l+1}) \le C(A_i)$ and $\bigcup_{i \le g \le n} A_g \subseteq \bigcup_{i \le g \le n} A_g^{l+1}$ for $1 \le i \le n$. Similarly, we can prove that $C(A_i^r) \le C(A_i)$ and $\bigcup_{i \le g \le n} A_g \subseteq \bigcup_{i \le g \le n} A_g^r$ for $1 \le i \le n$, where $l + 2 \le r \le l'$. Thereby, $C_{\max}(\pi_{k+1}) = C(A_n^{l'}) \le C(A_n) = C_{\max}(\sigma)$. Since π_{k+1} is a feasible adjoint-schedule of P_{f^k}, we have $C_{\max}(\pi_{k+1}) = C_{\max}(\sigma)$ by the optimality of σ. Therefore, π_{k+1} is the optimal adjoint-schedule of P_{f^k}. □

Theorem 2. *Algorithm PO can solve problem $1|(p_j, q_j), s\text{-}batch, b < n|(f_{\max}, C_{\max})$ in $O(n^4)$ time.*

Proof. The correctness of Algorithm PO comes from Theorem 1 and Lemma 5 and Corollary 1. Moreover, from the picking rule of the schedules in Steps 7–8, $\pi(\mathcal{J})$ that is obtained by Algorithm PO is the Pareto optimal set of $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$.

Next, we analyze the complexity of Algorithm PO. $O(n \log n)$ time is needed to run Step 1. Steps 2–6 are one round that needs $O(n^2 + n \log n) = O(n^2)$ time, *i.e.*, it takes $O(n^2)$ time to yield σ^k from σ^{k-1}. In each

round, each specific component can only move forward (assume that a specific component moves from A_k to A_i ($k \neq i$). Then "move forward" means $k > i$), and at least one specific component is moved forward (except the round that appears after the number of the nonempty batches increases, *i.e.*, if (6.2) occurs, then it is possible that $\sigma^k = \sigma^{k+1}$. For example, $\sigma^3 = \sigma^4$ in Example. This case involves at most $O(n)$ rounds). Each specific component can traverse up to $n-1$ sets. Thus, there is at most $O(n^2+n) = O(n^2)$ rounds. So $O(n^4)$ time is needed to run all rounds. And the set $\{\pi_1, \pi_2, \ldots, \pi_h\}$ contains at most $O(n^2)$ schedules and satisfies $f_{\max}(\pi_1) > f_{\max}(\pi_2) > \cdots > f_{\max}(\pi_h)$. Hence, it takes $O(n^3)$ time to compute and pick all Pareto optimal points. So the total running time of Algorithm PO is $O(n^4)$ time. □

We can derive the following corollaries by Theorem 2.

Corollary 2. *Problem* $1|(p_j, q_j), s\text{-}batch, b < n|(f_{\max}, C_{\max})$ *has at most* $O(n)$ *Pareto optimal points.*

Proof. Let $\pi(\mathcal{J}) = \{\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_{h'}}\} (\subseteq \{\pi_1, \pi_2, \ldots, \pi_h\})$ be the Pareto optimal set (obtained by Algorithm PO) of $1|(p_j, q_j), \text{s-batch}, b < n|(f_{\max}, C_{\max})$, and $f_{\max}(\pi_{i_1}) > f_{\max}(\pi_{i_2}) > \cdots > f_{\max}(\pi_{i_{h'}})$. Then $C_{\max}(\pi_{i_1}) < C_{\max}(\pi_{i_2}) < \cdots < C_{\max}(\pi_{i_{h'}})$. From $C_{\max}(\pi_{i_1}) = \|\pi_{i_1}\|s + T$ and r and T are constants, we have $\|\pi_{i_1}\| < \|\pi_{i_2}\| < \cdots < \|\pi_{i_{h'}}\| \leq n$. Hence, the result holds. □

Corollary 3. $1|(p_j, q_j), s\text{-}batch, b < n|f_{\max}$ *can be solved in* $O(n^4)$ *time.*

Lemma 6. *If* $1|(p_j, q_j), s\text{-}batch, b < n, L_{\max} \leq f|C_{\max}$ *is feasible, then there exists an optimal adjoint-schedule* $\sigma = (B_1, A_1, \ldots, B_n, A_n)$ *in which the specific components are scheduled in EDD-rule, i.e., if* $C_i(\sigma) < C_j(\sigma)$, *then* $d_i \leq d_j$.

Proof. It can be proved by Lemma 2 and the definition of $L_{\max}$. □

Corollary 4. *Problem* $1|(p_j, q_j), s\text{-}batch, b < n|(L_{\max}, C_{\max})$ *can be solved in* $O(n^3 \log n)$ *time.*

Proof. From Lemma 6, $O(n + n\log n) = O(n\log n)$ time is needed to execute each round in Algorithm PO. Similar to the proof of Theorem 2, problem $1|(p_j, q_j), \text{s-batch}, b < n|(L_{\max}, C_{\max})$ can be solved in $O(n^3 \log n)$ time, as required. □

Funding

This work was supported by NSFHN (No. 262300421833, 262300422612, 252300421483) and STRPHN (No. 262102241064).

References

[1] A. Agnetis, J.-C. Billaut, S. Gawiejnowicz, D. Pacciarelli and A. Soukhal, Multiagent Scheduling: Models and Algorithms. Springer-Verlag, Berlin (2014).
[2] K.R. Baker, Scheduling the production of components at a common facility. *IIE Trans.* **20** (1988) 32–35.
[3] K.R. Baker and J.C. Smith, A multiple-criterion model for machine scheduling. *J. Sched.* **6** (2003) 7–16.
[4] P. Brucker, Scheduling Algorithms, 3rd edition. Springer, Berlin (2001).
[5] Z.C. Geng, J.J. Yuan and J.L. Yuan, Scheduling with or without precedence relations on a serial-batch machine to minimize makespan and maximum cost. *Appl. Math. Comput.* **332** (2018) 1–18.
[6] A.E. Gerodimos, C.A. Glass and C.N. Potts, Scheduling the production of two-component jobs on a single machine. *Eur. J. Oper. Res.* **120** (2000) 250–259.
[7] C. He, Y.X. Lin, and J.J. Yuan, Bicriteria scheduling of minimizing maximum lateness and makespan on a serial-batching machine. *FCDS* **33** (2008) 369–376.
[8] C. He, X.M. Wang, Y.X. Lin and Y.D. Mu, An improved algorithm for a bicriteria batching scheduling problem. *RAIRO Oper. Res.* **47** (2013) 1–8.

[9] C. He, H. Lin and Y.X. Lin, Bounded serial-batching scheduling for minimizing maximum lateness and makespan. *Discrete Optim.* **16** (2015) 70–75.
[10] C. He, H. Lin and L. Li, Unbounded serial-batching scheduling on hierarchical optimization. *JORSC* **9** (2021) 909–914.
[11] C. He, H. Lin and L. Li, Hierarchical minimization of two maximum costs on a bounded serial-batching machine. *RAIRO Oper. Res.* **55** (2021) 135–140.
[12] C. He, J. Wu, H. Lin, Y. Zhang and Y. Zhao, The unbounded parallel-batching bicriteria scheduling with two-component jobs. *JORSC* (2024) 1–18. DOI: https://doi.org/10.1007/s40305-024-00545-0.
[13] C. He, J. Wu, H. Guo, B.R. Sun, H. Lin and Y. Zhang, Bicriteria scheduling with two-component jobs on an unbounded series-batching machine. *JORSC* (2025), Accepted.
[14] H. Hoogeveen, Multicriteria scheduling. *Eur. J. Oper. Res.* **167** (2005) 592–623.
[15] J.A. Hoogeveen, Single-machine scheduling to minimize a function of two or three maximum cost criteria. *J. Algorithms* **21** (1996) 415–433.
[16] E.L. Lawler, Optimal sequencing of a single machine subject to precedence constraints. *Manag. Sci.* **19** (1973) 544–546.
[17] Y.J. Li, Bicriteria fabrication scheduling of two component jobs on a single machine. *Oper. Res.* **23** (2023) 1–13.
[18] Y.J. Li, Bounded scheduling two-component jobs simultaneously or hierarchically. *RAIRO Oper. Res.* **58** (2024) 1115–1130.
[19] V. T'kindt and J.C. Billaut, Multicriteria scheduling problems: a survey. *RAIRO Oper. Res.* **35** (2001) 143–163.