Networks and Heterogeneous Media

http://www.aimspress.com/journal/nhm

NHM, 18(4): 1678–1691.
DOI: 10.3934/nhm.2023073
Received: 30 December 2022
Revised: 04 June 2023
Accepted: 21 September 2023
Published: 27 September 2023

*Research article*

# Algorithms for two-agent unbounded serial-batch scheduling with makespan and maximum lateness objectives

**Shuguang Li**[1,*], **Mingsong Li**[1] **and Muhammad Ijaz Khan**[2,3]

[1] School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China

[2] Department of Mechanical Engineering, Lebanese American University, Beirut 362060, Lebanon

[3] Department of Mechanics and Engineering Science, Peking University, Beijing 100871, China

* **Correspondence:** Email: sgliytu@hotmail.com; Tel: +86-18753509226.

**Abstract:** We study the problem of non-preemptively scheduling jobs from two agents on an unbounded serial-batch machine. Agents $A$ and $B$ have $n_A$ and $n_B$ jobs. The machine can process any number of jobs sequentially as a batch, and the processing time of the batch is equal to the total processing time of the jobs in it. Each batch requires a setup time before it is processed. Compatibility means that the jobs from different agents can be processed in a common batch; Otherwise, the jobs from different agents are incompatible. Both the compatible and incompatible models are considered, under both the batch availability and item availability assumptions. Batch availability means that any job in a batch is not available until all the jobs in this batch are completed. Item availability means that a job in a batch becomes available immediately after it is completed processing. The completion time of a job is defined to be the moment when it is available. The goal is to minimize the makespan of agent $A$ and the maximum lateness of agent $B$ simultaneously. For the compatible model with batch availability, an $O(n_A + n_B^2 \log n_B)$-time algorithm is presented which improves the existing $O(n_A + n_B^4 \log n_B)$-time algorithm. A slight modification of the algorithm solves the incompatible model with batch availability in $O(n_A + n_B^2 \log n_B)$ time, which has the same time complexity as the existing algorithm. For the compatible model with item availability, the analysis shows that it is easy and admits an $O(n_A + n_B \log n_B)$-time algorithm. For the incompatible model with item availability, an $O(n_A + n_B \log n_B)$-time algorithm is also obtained which improves the existing $O(n_A + n_B^2)$-time algorithm. The algorithms can generate all Pareto optimal points and find a corresponding Pareto optimal schedule for each Pareto optimal point.

**Keywords:** two-agent scheduling; unbounded serial-batch; compatibility/incompatibility; batch/item availability; maximum lateness

## 1. Introduction

We study a problem of two-agent scheduling on an unbounded serial-batch machine. There are two agents $A$ and $B$ having $n_A$ and $n_B$ jobs respectively, where $n_A + n_B = n$. Each agent has his own objective function to optimize that depends on the completion times of his own jobs only. However, all the jobs share a common processing resource (e.g., the serial-batch machine). In this paper we assume that agent $A$ expects the maximum completion time (makespan) of his jobs to be minimized, while agent $B$ expects the maximum lateness (to be defined later) of his jobs to be minimized. Using the standard scheduling notation, see, for example, [1], this circumstance can be modeled as an objective vector $(C^A_{\max}, L^B_{\max})$. For the purpose of meeting the needs of two agents to the maximum extent, the decision-maker has to design some strategies to stimulate the agents to cooperate.

The jobs are processed on the machine in batches, and a fixed setup time $s > 0$ is incurred before each batch is started. The *serial-batch* machine can process at most one job at a time and cannot process any job when a setup is being performed. The processing time of a batch is equal to the total processing time of the jobs in it. This is different from *parallel-batch*, where the jobs in the same batch are processed simultaneously and the processing time of a batch is equal to the largest processing time of the jobs in it [2, 3]. An *unbounded* machine means that the batch capacity $b$ (the maximum number of jobs in a batch) is unlimited, i.e., $b \geq n$; Otherwise, it is *bounded*, i.e., $b < n$. *Compatibility* means that the jobs from different agents can be processed in a common batch; Otherwise, the jobs from different agents are *incompatible*. We analyze both the compatible and incompatible models under both the batch availability and item availability assumptions. *Batch availability* means that any job in a batch is not available until all the jobs in this batch are completed. *Item availability* means that a job in a batch becomes available immediately after it is completed processing. The completion time of a job is defined to be the moment when it is available.

**Problem formulation**

There are two agents $A$ and $B$, and agent $X$ has a set $\mathcal{J}^X = \{J^X_1, J^X_2, \ldots, J^X_{n_X}\}$ of jobs to be non-preemptively processed on an unbounded serial-batch machine, $X \in \{A, B\}$. The jobs in $\mathcal{J}^X$ are called *X-jobs*. Assume that $\mathcal{J}^A \cap \mathcal{J}^B = \emptyset$. Let $\mathcal{J}^A \cup \mathcal{J}^B = \mathcal{J}$ and $n_A + n_B = n$. Each job $J^X_j$ has a processing time $p^X_j$. Each $B$-job $J^B_j$ has a due date $d^B_j$.

A schedule can be specified by a sequence of batches $B_1, B_2, \ldots, B_x$. A fixed setup time $s > 0$ is incurred before each batch. Let $p(B_i)$ and $C(B_i)$ denote the processing time and completion time of $B_i$, respectively. Clearly, $C(B_i) = i \cdot s + \sum_{q=1}^{i} p(B_q)$, $i = 1, 2, \ldots x$. For the case of batch availability, all jobs in $B_i$ are completed simultaneously at time $C(B_i)$. In contrast, for the case of item availability, a job is completed immediately after it is completed processing.

A batch containing only $X$-jobs is an *X-batch*, $X \in \{A, B\}$. A *mixed batch* refers to a batch that contains both $A$-jobs and $B$-jobs.

Given a feasible schedule $\sigma$ of the $n$ jobs, let $C^X_j(\sigma)$ denote the completion time of $J^X_j$ in $\sigma$. Let $L^B_j(\sigma) = C^B_j(\sigma) - d^B_j$ denote the lateness of job $J^B_j$. If there is no confusion, we simply use $C^X_j$ and $L^B_j$ to denote $C^X_j(\sigma)$ and $L^B_j(\sigma)$, respectively. Let $C^X_{\max} = \max_{j=1}^{n_X}\{C^X_j\}$ and $C_{\max} = \max\{C^A_{\max}, C^B_{\max}\}$ denote the makespan of $X$-jobs and the makespan of $\sigma$, respectively. Let $L^B_{\max} = \max_{j=1}^{n_B}\{L^B_j\}$ be the maximum lateness of $B$-jobs. Notice that the makespan and maximum lateness criteria are both regular, i.e., they are non-decreasing in the job completion times.

This paper studies the two-agent scheduling problem on an unbounded serial-batch machine to

minimize the makespan $C^A_{\max}$ of agent $A$ and the maximum lateness $L^B_{\max}$ of agent $B$ simultaneously. Using the three-field notation introduced by Graham et al. [4] and extended by T'kindt and Billaut [5], the four variants of the problem under consideration are denoted by $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$, $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$, $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$ and $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$, where "*co*" and "*inco*" mean "compatibility" and "incompatibility", and "*batch − avail*" and "*item − avail*" denote the "batch availability" and "item availability", respectively.

Since we are dealing with two criteria $C^A_{\max}$ and $L^B_{\max}$ which are not non-conflicting, we focus on the calculation of Pareto optima for $C^A_{\max}$ and $L^B_{\max}$. The approach is called *Pareto optimization* which is one of the most popular approaches in multi-criteria scheduling [5, 6].

Let $f^A$ and $g^B$ denote the two objectives of agents $A$ and $B$ respectively to be minimized. A feasible schedule $\sigma$ is *Pareto optimal* with respect to $f^A$ and $g^B$ if there is no feasible schedule $\sigma'$ such that $f^A(\sigma') \leq f^A(\sigma)$ and $g^B(\sigma') \leq g^B(\sigma)$ with at least one strict inequality. The objective vector $(f^A(\sigma), g^B(\sigma))$ is called a *Pareto optimal point* [5, 6].

**Literature review**

The problem under consideration falls into the categories of batch scheduling [2, 3, 7] and multi-criteria optimization [5,6], which are hotspots in scheduling research. Multi-agent scheduling [1,8–10] is a research topic in multi-criteria optimization. Here, we only review the results closely related to our study.

Webster and Baker [11] proposed an $O(n^2)$-time algorithm for $1|s-batch, b \geq n, batch-avail|L_{\max}$ (single-criterion problem). Wagelmans and Gerodimos [12] improved the time complexity to $O(n \log n)$. He et al. [13] presented an $O(n^2)$-time algorithm for $1|s-batch, b \geq n, batch-avail|(C_{\max}, L_{\max})$ (Pareto optimization problem). Later, the result has been extended to an $O(n^5)$-time algorithm for $1|s-batch, b \geq n, batch-avail|(C_{\max}, f_{\max})$ (maximum lateness $L_{\max}$ is replaced by maximum cost $f_{\max}$) [14]. He et al. [15] obtained an $O(n^6)$-time algorithm for $1|s-batch, b < n, batch-avail|(C_{\max}, L_{\max})$ (bounded serial-batch). Geng et al. [16] presented $O(n^4)$-time algorithms for $1|s-batch, b \geq n, batch-avail|(C_{\max}, f_{\max})$ and $1|s-batch, b < n, batch-avail|(C_{\max}, f_{\max})$. Hence, problems $1|s-batch, b \geq n, batch-avail|f_{\max}$ and $1|s-batch, b < n, batch-avail|f_{\max}$ can also be solved in $O(n^4)$ time.

There are many research results for two-agent scheduling on an incompatible serial-batch machine with batch availability. Kovalyov et al. [17] derived polynomial and pseudo-polynomial time algorithms for constrained optimization problems with various combinations of the objective functions. *Constrained optimization* means that minimizing the objective value of agent $A$, subject to the condition that the objective value of agent $B$ is bounded by a given threshold value. Feng et al. [18] provided an $O(n_A + {n_B}^4)$-time algorithm for $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$. He et al. [19] presented an $O(n_A + {n_B}^5)$-time algorithm for $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, f^B_{\max})$ and an $O(n_A + {n_B}^4 \log n_B)$-time algorithm for $1|s-batch, b < n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$. He and Lin [20] gave an improved algorithm for $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$ that runs in $O(n_A + {n_B}^2 \log n_B)$ time. He et al. [21] presented an $O(n_A + {n_B}^5)$-time algorithm for $1|s-batch, b < n, inco, batch-avail|(C^A_{\max}, f^B_{\max})$. He et al. [22] presented an $O({n_A}^3 {n_B}^3 n^2)$-time algorithm for $1|s-batch, b \geq n, inco, batch-avail|(L^A_{\max}, L^B_{\max})$.

To the best of our knowledge, very few results have been published for two-agent scheduling on

a serial-batch machine with compatibility or item availability. For a compatible model, Li et al. [23] provided either polynomial or pseudo-polynomial time algorithms for some constrained optimization problems arising from different combinations of several regular criteria, including the maximum cost, the total completion time, and the (weighted) number of tardy jobs. He et al. [22] presented an $O(n_A + {n_B}^4 \log n_B)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$, and an $O(n_A n_B n^3)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(L^A_{\max}, L^B_{\max})$. They also presented an $O(n_A + {n_B}^2)$-time algorithm for $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$, and an $O({n_A}^2 {n_B}^2 n)$-time algorithm for $1|s-batch, b \geq n, inco, item-avail|(L^A_{\max}, L^B_{\max})$.

**Contribution and organization**

The main result of this paper is an $O(n_A + {n_B}^2 \log n_B)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$, improving the $O(n_A + {n_B}^4 \log n_B)$-time algorithm presented in [22]. A slight modification of the algorithm can solve $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$ in $O(n_A + {n_B}^2 \log n_B)$ time, which has the same time complexity as the algorithm in [20]. We also consider the item availability. For $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$ and $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$, we present $O(n_A + n_B \log n_B)$-time algorithms. The algorithm for $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$ improves the $O(n_A + {n_B}^2)$-time algorithm presented in [22].

The paper is organized as follows. In Section 2, we present an $O(n_A + {n_B}^2 \log n_B)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$, and indicate the slight modification of the algorithm for the incompatible model. In Section 3, we present $O(n_A + n_B \log n_B)$-time algorithms for $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$ and $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$. Some concluding remarks are drawn in Section 4.

## 2. The batch availability

In this section we will present an $O(n_A + {n_B}^2 \log n_B)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$. At the end of this section, we will indicate the slight modification of the algorithm for the incompatible model.

We need the following lemma which describes the structure of the Pareto optimal schedules we are searching for.

**Lemma 2.1.** *( [22]) For any Pareto optimal point of* $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$, *there is a corresponding Pareto optimal schedule so that all A-jobs belong to a common batch, and all B-jobs are scheduled in EDD (earliest due date first) order.*

By this lemma, we re-index all $B$-jobs in EDD order so that $d^B_1 \leq d^B_2 \leq \cdots \leq d^B_{n_B}$. The single batch containing all $A$-jobs may also contain $B$-jobs. For brevity, we will represent a schedule by a batch sequence $B_1, B_2, \ldots, B_{n_B}, B_{n_B+1}, \ldots, B_{2n_B+1}$, where $B_{n_B+1}$ is the batch containing all $A$-jobs and possibly some $B$-jobs. Some batches in $B_1, B_2, \ldots, B_{n_B}, B_{n_B+1}, \ldots, B_{2n_B+1}$ may be empty or dummy. An *empty batch* has processing time zero and setup time zero, while a *dummy batch* has processing time zero and setup time $s$.

Let us roughly illustrate the basic idea of the algorithm, conveniently showing the difference between the empty and dummy batches. There are many empty batches in the initial schedule. We

iteratively adjust the schedule to decrease its $L^B_{\max}$-value. The adjustment is accomplished by moving some $B$-jobs (whose lateness values are not less than the currently fixed $L^B_{\max}$-value) to the left. Consequently, an empty batch may become nonempty and a setup time $s$ incurs because some jobs are moved into it, and a nonempty batch may become a dummy batch because all its jobs have been moved out. Although the nonempty batch now contains no jobs, we retain it in the batch sequence with processing time zero and setup time $s$. The motivation for introducing the concept of "dummy batch" is to ensure that any job cannot be moved to the right. It is worth pointing out that a dummy batch will never appear earlier than batch $B_{n_B+1}$.

Let $\Omega(\mathcal{J})$ denote the Pareto set which consists of all Pareto optimal points together with the corresponding schedules. The algorithm below for constructing $\Omega(\mathcal{J})$ will repeatedly apply the standard $\varepsilon$-constraint approach.

**Lemma 2.2.** *( [6]) Let $y$ be the optimal value of constraint optimization problem $\alpha|f \le \hat{x}|g$, and let $x$ be the optimal value of constraint optimization problem $\alpha|g \le y|f$. Then the standard $\varepsilon$-constraint approach tells us that $(x, y)$ is a Pareto optimal point for problem $\alpha||(f, g)$.*

Let $\Pi(\mathcal{J})$ denote the set of all feasible schedules for $\mathcal{J} = \mathcal{J}^A \cup \mathcal{J}^B$. Let $\Pi(\mathcal{J}, y) \subseteq \Pi(\mathcal{J})$ denote the set of the schedules whose $L^B_{\max}$-values are less than $y$, where $y$ is a given threshold value.

**Algorithm BATCHCO:**

Step 1. Initially, set $\Omega(\mathcal{J}) = \varnothing$, $z = 0$. Set $h = 0$, $y^{(h)} = +\infty$. Let $\sigma^{(h)} = (B^{(h)}_1, B^{(h)}_2, \ldots, B^{(h)}_{n_B}, B^{(h)}_{n_B+1}, \ldots, B^{(h)}_{2n_B+1})$ be the initial schedule, where $B^{(h)}_{n_B+1}$ consists of all $A$-jobs, $B^{(h)}_{2n_B+1}$ consists of all $B$-jobs, and all other batches are empty batches.

Step 2. The $(h + 1)$-th iteration:

Set $y^{(h+1)} = L^B_{\max}(\sigma^{(h)})$. Invoke Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$ to adjust $\sigma^{(h)}$ to construct $\sigma^{(h+1)}$ so that $\sigma^{(h+1)}$ has minimum $C^A_{\max}$ among the schedules in $\Pi\left(\mathcal{J}, y^{(h+1)}\right)$.

Step 3. If $\sigma^{(h+1)} = \emptyset$, then set $\pi^* = \sigma^{(h)}$. Let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(C^A_{\max}(\pi^*), L^B_{\max}(\pi^*), \pi^*)\}$ and return $\Omega(\mathcal{J})$. Otherwise, if $C^A_{\max}(\sigma^{(h+1)}) > C^A_{\max}(\sigma^{(h)})$, then set $z = z + 1$, $\pi_z = \sigma^{(h)}$, and let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(C^A_{\max}(\pi_z), L^B_{\max}(\pi_z), \pi_z)\}$.

Step 4. Set $h = h + 1$ and go to Step 2.

**Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$:**

Step 1. Check the batches in $\sigma^{(h)}$ backwardly: For $i = 2n_B + 1, 2n_B, \ldots, 1$, check the inequality $L^B_j(C(B^{(h)}_i)) < y^{(h+1)}$ for each job $J^B_j$ in $B^{(h)}_i$. Suppose that we find a job $J^B_k \in B^{(h)}_i$ which violates the inequality $L^B_k(C(B^{(h)}_i)) < y^{(h+1)}$. We distinguish two different cases:

**Case 1.** $i \le n_B + 1$.

If $i = 1$, or $i \le n_B$ and $J^B_k$ has the largest due date in $B^{(h)}_i$, then return $\sigma^{(h+1)} = \emptyset$. Otherwise, move all the $B$-jobs in $B^{(h)}_i$ with their due dates no more than $d^B_k$ into $B^{(h)}_{i-1}$. If $B^{(h)}_{i-1}$ is empty before these jobs are inserted, then introduce a new setup time $s$ for it.

**Case 2.** $i > n_B + 1$

**Subcase 2.1.** Batch $B^{(h)}_{i-1}$ is not an empty batch (but possibly it is a dummy batch).

Move all the $B$-jobs in $B_i^{(h)}$ with their due dates no more than $d_k^B$ into $B_{i-1}^{(h)}$. If $B_i^{(h)}$ contains no job after these jobs are moved, then it becomes a dummy batch.

**Subcase 2.2.** Batch $B_{i-1}^{(h)}$ is an empty batch, i.e., it has processing time zero and setup time zero.

**Subcase 2.2.1.** There is no dummy batch which succeeds $B_i^{(h)}$.

Introduce a new setup time $s$ for $B_{i-1}^{(h)}$. Move all the $B$-jobs in $B_i^{(h)}$ with their due dates no more than $d_k^B$ into $B_{i-1}^{(h)}$. If $B_i^{(h)}$ contains no job after these jobs are moved, then it becomes a dummy batch.

**Subcase 2.2.2.** There is a dummy batch which succeeds $B_i^{(h)}$.

Move all the $B$-jobs in $B_i^{(h)}$ with their due dates no more than $d_k^B$ into $B_{n_B+1}^{(h)}$. If $B_i^{(h)}$ contains no job after these jobs are moved, then it becomes a dummy batch. All the empty batches $B_{n_B+2}^{(h)}, B_{n_B+3}^{(h)}, \ldots, B_{i-1}^{(h)}$ keep unchanged.

Step 2. Update the completion times of the jobs and the batches. Update the lateness values of the jobs accordingly.

Step 3. Repeat Steps 1 and 2 until there is no inequality violation in adjusted $\sigma^{(h)}$. Let $\sigma^{(h+1)}$ be the final $\sigma^{(h)}$.

In the $h$-iteration, Algorithm BATCHCO constructs $\sigma^{(h)}$ by adjusting $\sigma^{(h-1)}$. The adjustment of $\sigma^{(h-1)}$ is accomplished by performing a series of inequality violation adjustments. During the iteration, we get a series of tentative schedules $\sigma_1^{(h)}, \sigma_2^{(h)}, \ldots, \sigma_{\lambda_h}^{(h)}$, where $\lambda_h$ is the total number of tentative schedules in the $h$-iteration. Note that $\sigma_1^{(h)}$ is just the last tentative schedule in the preceding iteration, i.e., $\sigma_1^{(h)} = \sigma_{\lambda_{h-1}}^{(h-1)}$. Each next tentative schedule is obtained from the current tentative schedule by performing an inequality violation adjustment. When we describe Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h)})$, all tentative schedules except for the last one are denoted by $\sigma^{(h-1)}$, and $\sigma_{\lambda_h}^{(h)}$ is denoted by $\sigma^{(h)}$. Here, for clarity of the discussion, the tentative schedules need to be distinguished more clearly. Among these tentative schedules, only $\sigma_{\lambda_h}^{(h)}$ belongs to $\Pi\left(\mathcal{J}, y^{(h)}\right)$.

If a schedule belongs to $\Pi\left(\mathcal{J}, y^{(h)}\right)$, then all jobs in it have lateness values less than $y^{(h)}$, and vice versa. To analyze Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h)})$, we need the concept of candidate schedule. A *candidate schedule* for $y^{(h)}$ is a schedule which has the potential to be in $\Pi\left(\mathcal{J}, y^{(h)}\right)$. Denote by $\Lambda\left(\mathcal{J}, y^{(h)}\right)$ the set of candidate schedules for $y^{(h)}$. If we find a job in a schedule with lateness greater than or equal to $y^{(h)}$, then we get evidence that this schedule cannot be in $\Lambda\left(\mathcal{J}, y^{(h)}\right)$. It is possible that at first a schedule is treated as in $\Lambda\left(\mathcal{J}, y^{(h)}\right)$ because we have not found any job in it with lateness greater than or equal to $y^{(h)}$, but later we find evidence and exclude this schedule from $\Lambda\left(\mathcal{J}, y^{(h)}\right)$. Clearly, $\Pi\left(\mathcal{J}, y^{(h)}\right) \subseteq \Lambda\left(\mathcal{J}, y^{(h)}\right)$.

For any given schedule $\sigma = (B_1, B_2, \ldots, B_{n_B}, B_{n_B+1}, \ldots, B_{2n_B+1})$, let $\sigma_L = (B_1, B_2, \ldots, B_{n_B})$ and $\sigma_R = (B_{n_B+2}, B_{n_B+3}, \ldots, B_{2n_B+1})$ denote the *left side* and *right side* of $\sigma$, respectively. Let $n(\sigma_L)$ and $n(\sigma_R)$ denote the numbers of nonempty $B$-batches on the left and right sides of $\sigma$, respectively. Let $\Gamma^B(\sigma)$ denote the set of the $B$-jobs in $\bigcup_{q=1}^{n_B+1} B_q$, i.e., the set of the $B$-jobs in $\sigma$ which are not on the right side.

**Lemma 2.3.** *Let $\sigma_w^{(h)} = (B_{w,1}^{(h)}, B_{w,2}^{(h)}, \ldots, B_{w,n_B}^{(h)}, B_{w,n_B+1}^{(h)}, \ldots, B_{w,2n_B+1}^{(h)})$ denote the $w$-th tentative schedule $(1 \le w \le \lambda_h)$ during the implementation of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h)})$ $(h = 0, 1\ldots)$. Let $\sigma =$*

$(B_1, B_2, \ldots, B_{n_B}, B_{n_B+1}, \ldots, B_{2n_B+1})$ *denote any schedule in* $\Lambda\left(\mathcal{J}, y^{(h)}\right)$. *Then the following properties hold:*

*(1)* $\Gamma^B(\sigma_1^{(h)}) \subseteq \Gamma^B(\sigma_2^{(h)}) \subseteq \cdots \subseteq \Gamma^B(\sigma_{\lambda_h}^{(h)}) \subseteq \Gamma^B(\sigma)$;

*(2) If* $\Gamma^B(\sigma_w^{(h)}) = \Gamma^B(\sigma)$, *then*

*(i)* $\bigcup_{q=i}^{n_B+1} B_{w,q}^{(h)} \supseteq \bigcup_{q=i}^{n_B+1} B_q$, $i = n_B+1, n_B, \ldots, 1$;

*(ii)* $n(\sigma_{w,L}^{(h)}) \leq n(\sigma_L)$;

*(iii) Batch* $B_{w,i}^{(h)}$ *starts and completes no later than batch* $B_i$, $i = 1, 2, \ldots, n_B+1$.

*Proof.* We prove the lemma by induction on $h$.

Consider $P_{BATCHCO}(\mathcal{J}, y^{(0)})$. Obviously, the initial schedule $\sigma^{(0)}$ (described in Step 1 of Algorithm BATCHCO) and any schedule in $\Lambda\left(\mathcal{J}, y^{(0)}\right) = \Pi\left(\mathcal{J}, y^{(0)}\right) = \Pi(\mathcal{J})$ satisfy the properties of the lemma, thus proving the base case.

Assume that the lemma holds for $P_{BATCHCO}(\mathcal{J}, y^{(0)}), P_{BATCHCO}(\mathcal{J}, y^{(1)}), \ldots, P_{BATCHCO}(\mathcal{J}, y^{(h)})$.

We now consider $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$.

Since $y^{(h+1)} < y^{(h)}$, we have: $\Lambda\left(\mathcal{J}, y^{(h+1)}\right) \subseteq \Lambda\left(\mathcal{J}, y^{(h)}\right)$. Let $\sigma$ be any schedule in $\Lambda\left(\mathcal{J}, y^{(h+1)}\right)$. Then $\sigma \in \Lambda\left(\mathcal{J}, y^{(h)}\right)$. Note that $\sigma_1^{(h+1)}$, the first tentative schedule for $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, is just $\sigma_{\lambda_h}^{(h)}$, the last one for $P_{BATCHCO}(\mathcal{J}, y^{(h)})$. By the inductive assumption, $\sigma_1^{(h+1)}$ and $\sigma$ satisfy the properties of the lemma. Assume that each of the first $w$ $(1 \leq w < \lambda_{h+1})$ tentative schedules and $\sigma$ satisfy the properties of the lemma. Below, we will prove that $\sigma_{w+1}^{(h+1)}$ and $\sigma$ also satisfy the properties of the lemma. Suppose that we find a job $J_k^B \in B_{w,i}^{(h+1)}$ which violates the inequality $L_k^B(C(B_{w,i}^{(h+1)})) < y^{(h+1)}$. There are two different cases to consider:

**Case 1.** $i \leq n_B + 1$.

As described in Case 1 of Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, we move all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ into $B_{w,i-1}^{(h+1)}$. Denote the obtained schedule as $\sigma_{w+1}^{(h+1)}$.

By the inductive assumption, we have: $\Gamma^B(\sigma_w^{(h+1)}) \subseteq \Gamma^B(\sigma)$. Since $\Gamma^B(\sigma_{w+1}^{(h+1)}) = \Gamma^B(\sigma_w^{(h+1)})$, we get: $\Gamma^B(\sigma_{w+1}^{(h+1)}) \subseteq \Gamma^B(\sigma)$. Hence, property (1) holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$.

To prove that property (2) holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$, we assume that $\Gamma^B(\sigma_{w+1}^{(h+1)}) = \Gamma^B(\sigma)$. Then we have: $\Gamma^B(\sigma_w^{(h+1)}) = \Gamma^B(\sigma)$. By the inductive assumption, we know that property (2) holds for $\sigma_w^{(h+1)}$ and $\sigma$. Hence, we have: $C(B_{w,i}^{(h+1)}) \leq C(B_i)$. Since $L_k^B(C(B_{w,i}^{(h+1)})) \geq y^{(h+1)}$, we get $L_k^B(C(B_i)) \geq y^{(h+1)}$. Hence, all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ have to be scheduled earlier than $B_i$ in $\sigma$.

The above argument ensures that property (i) of property (2) holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$. Thus, properties (ii) and (iii) of property (2) hold for $\sigma_{w+1}^{(h+1)}$ and $\sigma$.

**Case 2.** $i > n_B + 1$.

**Subcase 2.1.** Batch $B_{w,i-1}^{(h+1)}$ is not an empty batch (but possibly it is a dummy batch).

As described in Subcase 2.1 of Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, we move all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ into $B_{w,i-1}^{(h+1)}$.

If $i > n_B + 2$, then the operation deals with only the right side of $B_{w,n_B+1}^{(h+1)}$ and does not affect the two properties of the lemma. The lemma holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$.

If $i = n_B + 2$, then below we show an argument similar to the proof of Case 1, to prove that all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ have to be scheduled earlier than $B_i$ in $\sigma$.

Let $B_{w,z}^{(h+1)}$ denote the earliest dummy batch which succeeds $B_{w,i}^{(h+1)}$, where $z > i$. If there is no dummy batch which succeeds $B_{w,i}^{(h+1)}$, then set $z = 2n_B + 2$. It is easy to prove that $\bigcup_{q=i}^{z-1} B_{w,q}^{(h+1)} \subseteq \bigcup_{q=i}^{z-1} B_q$.

Remove the jobs in $(\bigcup_{q=i}^{z-1} B_q)\backslash(\bigcup_{q=i}^{z-1} B_{w,q}^{(h+1)})$ from batches $B_i, B_{i+1}, \ldots, B_{z-1}$ and let the obtained batches be $B'_i, B'_{i+1}, \ldots, B'_{z-1}$. Note that $B'_i, B'_{i+1}, \ldots, B'_{z-1}$ are all non-empty. We can get $\bigcup_{q=g}^{z-1} B_{w,q}^{(h+1)} \supseteq \bigcup_{q=g}^{z-1} B'_q$, $g = z-1, z-2, \ldots, i$. Hence, we get $C(B_{w,g}^{(h+1)}) \leq C(B'_g)$, $g = i, i+1, \ldots, z-1$. It follows that $C(B_{w,g}^{(h+1)}) \leq C(B_g)$, $g = i, i+1, \ldots, z-1$. Since $L_k^B(C(B_{w,i}^{(h+1)})) \geq y^{(h+1)}$, we get $L_k^B(C(B_i)) \geq y^{(h+1)}$. Hence, all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ have to be scheduled earlier than $B_i$ in $\sigma$.

Hence, for $i = n_B + 2$, property (2) holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$.

**Subcase 2.2.** Batch $B_{w,i-1}^{(h+1)}$ is an empty batch, i.e., it has processing time zero and setup time zero.

**Subcase 2.2.1.** There is no dummy batch which succeeds $B_{w,i}^{(h+1)}$.

As described in Subcase 2.2.1 of Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, we move all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ into $B_{w,i-1}^{(h+1)}$. The operation deals with only the right side of $B_{w,n_B+1}^{(h+1)}$ and does not affect the two properties of the lemma. The lemma holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$.

**Subcase 2.2.2.** There is a dummy batch which succeeds $B_{w,i}^{(h+1)}$.

Let $B_{w,z}^{(h+1)}$ denote the earliest dummy batch which succeeds $B_{w,i}^{(h+1)}$, where $z > i$. If we introduce a new setup time $s$ for $B_{w,i-1}^{(h+1)}$, then the completion times of batch $B_{w,i}^{(h+1)}$ and its succeeding batches will increase by $s$. Consequently, the jobs in $B_{w,z-1}^{(h+1)}$ will have costs not less than $y^{(h+1)}$. In order to decrease the undesirable costs of these jobs, we have to move all the jobs in $B_{w,g}^{(h+1)}$ into $B_{w,g-1}^{(h+1)}$, $g = z-1, z-2, \ldots, i$. Job $J_k^B$ has to violate its inequality once again. Nothing has changed except that the indices of $B_{w,i}^{(h+1)}, B_{w,i+1}^{(h+1)}, \ldots, B_{w,z-1}^{(h+1)}$ decrease by 1.

Therefore, we cannot introduce a new setup time $s$ for $B_{w,i-1}^{(h+1)}$. Instead, as described in Subcase 2.2.2 of Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, we move all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ into $B_{w,n_B+1}^{(h+1)}$. Denote the obtained schedule as $\sigma_{w+1}^{(h+1)}$.

Assume that $\Gamma^B(\sigma_w^{(h+1)}) = \Gamma^B(\sigma)$ before we do the operation described in Subcase 2.2.2 of Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$. Let $E$ denote the set of the jobs in $B_{w,i}^{(h+1)}, B_{w,i+1}^{(h+1)}, \ldots, B_{w,z-1}^{(h+1)}$. In $\sigma$, the number of the batches containing jobs in $E$ is not less than that number in $\sigma_w^{(h+1)}$, i.e., $z - i$. Hence, if $J_k^B$ is scheduled to the right side of $B_{n_B+1}$, then either $J_k^B$ or the jobs in the latest batch containing jobs in $E$ will have cost not less than $y^{(h+1)}$. Therefore, in $\sigma$, all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ cannot be scheduled to the right side of $B_{n_B+1}$.

The above argument shows that property (1) of the lemma holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$. Since we moved all the $B$-jobs in $B_{w,i}^{(h+1)}$ with their due dates no more than $d_k^B$ into $B_{w,n_B+1}^{(h+1)}$, combining the inductive assumption, we know that property (2) of the lemma also holds for $\sigma_{w+1}^{(h+1)}$ and $\sigma$. This completes the proof for $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$.

By the principle of induction, we complete the proof of the lemma.

□

We get:

**Lemma 2.4.** *Let $\sigma^{(h)}$ denote the last schedule upon the completion of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h)})$ $(h = 0, 1\ldots)$. If $\sigma^{(h)} = \varnothing$, then $\Pi\left(\mathcal{J}, y^{(h)}\right) = \varnothing$; Otherwise $\sigma^{(h)}$ has minimum $C_{\max}^A$ among all schedules in $\Pi\left(\mathcal{J}, y^{(h)}\right)$.*

*Proof.* During the entire implementation of Algorithm BATCHCO, any job can only be moved to the

left. Consequently, the completion time of any batch will not decrease, which in turn ensures that any job cannot be moved to the right.

Suppose that in implementing $P_{BATCHCO}(\mathcal{J}, y^{(h)})$ we find a job $J_k^B \in B_i^{(h-1)}$ such that $L_k^B(C(B_i^{(h-1)})) \geq y^{(h)}$. If $i = 1$, then $J_k^B$ cannot be feasibly scheduled in any schedule in $\Pi\left(\mathcal{J}, y^{(h)}\right)$, implying that $\Pi\left(\mathcal{J}, y^{(h)}\right) = \varnothing$. Therefore, we return $\sigma^{(h)} = \varnothing$. If $i \leq n_B$ and $B_i^{(h-1)}$ contains no jobs after we move all the $B$-jobs in $B_i^{(h-1)}$ with their due dates no more than $d_k^B$ into $B_{i-1}^{(h-1)}$, we also return $\sigma^{(h)} = \varnothing$. This can be verified by the minimality of the number of nonempty $B$-batches in $\sigma_L^{(h-1)}$ (by property (ii) of property (2) of Lemma 2.3).

On the other hand, if $\sigma^{(h)} \neq \varnothing$, then by property (iii) of property (2) of Lemma 2.3, $\sigma^{(h)}$ has minimum $C_{\max}^A$ among all schedules in $\Pi\left(\mathcal{J}, y^{(h)}\right)$.

□

We get:

**Theorem 2.5.** *Algorithm BATCHCO solves* $1|s-batch, b \geq n, co, batch-avail|(C_{\max}^A, L_{\max}^B)$ *in* $O(n_A + {n_B}^3)$ *time.*

*Proof.* The correctness of the algorithm follows from Lemma 2.2 and Lemma 2.4.

In $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, finding and adjusting an inequality violation requires $O(n_B)$ time. In each adjustment, at least one $B$-job has to be moved to the left. Since any $B$-job can only be moved to the left, in the entire implementation of Algorithm BATCHCO, each $B$-job can go through at most $2n_B$ batches. The total number of adjustments is $O({n_B}^2)$. The running time of Algorithm BATCHCO is $O(n_A + {n_B}^3)$ time.

□

Next, we show how to improve the time complexity of the algorithm to $O(n_A + {n_B}^2 \log n_B)$.

For storing the completion times of all $B$-jobs, we use an array *Abcompl*, where the $j$-position of *Abcompl* stores the completion time of job $J_j^B$ in the current schedule, $j = 1, 2, \ldots, n_B$. The completion times of all jobs in a common batch are equal. Recall that all $B$-jobs are scheduled in EDD order. We set an indicator for each batch that points to the first $B$-job (which has the smallest due date among the $B$-jobs in this batch) in this batch, whereby we immediately know the contents of all batches in the schedule.

It is useful to store the lateness values of all $B$-jobs in a max-heap [24], so that the $L_{\max}^B$-value of the current schedule can be extracted in $O(\log n_B)$ time. However, the difficulty comes from how to efficiently maintain the heap. Therefore, we give up the idea. Instead, we use a max-heap *Hblateness* to store the lateness values of all batches consisting of at least one $B$-job, where the lateness value of a batch is defined to be the maximum lateness value of the $B$-jobs in this batch, which is equal to the completion time of the batch minus the due date of the first $B$-job in it. Thus, the $L_{\max}^B$-value of the current schedule can be extracted from *Hblateness* in $O(\log n_B)$ time.

Let us illustrate how to efficiently maintain the array and the max-heap. Suppose that we are adjusting $\sigma^{(h)} = (B_1^{(h)}, B_2^{(h)}, \ldots, B_{n_B}^{(h)}, B_{n_B+1}^{(h)}, \ldots, B_{2n_B+1}^{(h)})$ in Step 1 of Procedure $P_{BATCHCO}(\mathcal{J}, y^{(h+1)})$, and we find a job $J_k^B \in B_i^{(h)}$ violating its inequality. As described in Step 1, there are two different cases to consider:

**Case 1.** $i \leq n_B + 1$.

If $i = 1$, or $i \le n_B$ and $J_k^B$ has the largest due date in $B_i^{(h)}$, then return $\sigma^{(h+1)} = \emptyset$. Otherwise, let $B_{i,1}^{(h)}$ denote the set of the jobs to be moved from $B_i^{(h)}$ to $B_{i-1}^{(h)}$, i.e., $B_{i,1}^{(h)}$ consists of the $B$-jobs with their due dates no more than $d_k^B$ in $B_i^{(h)}$. Let $p(B_{i,1}^{(h)}) = \sum_{J_j^B \in B_{i,1}^{(h)}} p_j^B$.

**Subcase 1.1.** $B_{i-1}^{(h)}$ is not an empty batch.

After the jobs in $B_{i,1}^{(h)}$ are moved into $B_{i-1}^{(h)}$, the completion time of batch $B_{i-1}^{(h)}$, $C(B_{i-1}^{(h)})$, will increase by $p(B_{i,1}^{(h)})$. The lateness value of $B_{i-1}^{(h)}$ will also increase by $p(B_{i,1}^{(h)})$. All the other batches keep their completion times and lateness values unchanged. We just update the lateness value of $B_{i-1}^{(h)}$ in *Hblateness*. Therefore, finding a $B$-job violating its inequality can be done in $O(\log n_B)$ time. Each job movement can be done in constant time. Since there are $O({n_B}^2)$ job movements, it takes $O({n_B}^2 \log n_B)$ time to deal with this case.

**Subcase 1.2.** $B_{i-1}^{(h)}$ is an empty batch.

After the jobs in $B_{i,1}^{(h)}$ are moved into $B_{i-1}^{(h)}$, $B_{i-1}^{(h)}$ becomes nonempty and a setup time $s$ incurs before it is started. The completion times of all the jobs in $B_{i-1}^{(h)}$ and succeeding batches will change. Therefore, we recalculate them and update *Abcompl* accordingly. Then, we recalculate the lateness values of $B_{i-1}^{(h)}$ and succeeding batches and update *Hblateness* accordingly in $O(n_B \log n_B)$ time. Since this case occurs at most $O(n_B)$ times, it takes $O({n_B}^2 \log n_B)$ time to deal with it.

**Case 2.** $i > n_B + 1$.

**Subcase 2.1.** Batch $B_{i-1}^{(h)}$ is not an empty batch (but possibly it is a dummy batch).

Similarly to Subcase 1.1, let $B_{i,1}^{(h)}$ denote the set of the jobs to be moved from $B_i^{(h)}$ to $B_{i-1}^{(h)}$. Let $p(B_{i,1}^{(h)}) = \sum_{J_j^B \in B_{i,1}^{(h)}} p_j^B$. After the jobs in $B_{i,1}^{(h)}$ are moved into $B_{i-1}^{(h)}$, the completion time of batch $B_{i-1}^{(h)}$, $C(B_{i-1}^{(h)})$, will increase by $p(B_{i,1}^{(h)})$. The lateness value of $B_{i-1}^{(h)}$ will also increase by $p(B_{i,1}^{(h)})$. We update the lateness value of $B_{i-1}^{(h)}$ in *Hblateness*. Moreover, if $B_i^{(h)}$ contains no job after the jobs are moved, then it becomes a dummy batch and thus we delete the lateness value of $B_i^{(h)}$ from *Hblateness*. Since there are $O({n_B}^2)$ job movements, it takes $O({n_B}^2 \log n_B)$ time to deal with this case.

**Subcase 2.2.** Batch $B_{i-1}^{(h)}$ is an empty batch, i.e., it has processing time zero and setup time zero.

**Subcase 2.2.1.** There is no dummy batch which succeeds $B_i^{(h)}$.

Similarly to Subcase 1.2, let $B_{i,1}^{(h)}$ denote the set of the jobs to be moved from $B_i^{(h)}$ to $B_{i-1}^{(h)}$. Let $p(B_{i,1}^{(h)}) = \sum_{J_j^B \in B_{i,1}^{(h)}} p_j^B$. After the jobs in $B_{i,1}^{(h)}$ are moved into $B_{i-1}^{(h)}$, $B_{i-1}^{(h)}$ becomes nonempty and a setup time $s$ incurs before it is started. The completion times of all the jobs in $B_{i-1}^{(h)}$ and succeeding batches will change. Therefore, we recalculate them and update *Abcompl* accordingly. Then, we recalculate the lateness values of $B_{i-1}^{(h)}$ and succeeding batches and update *Hblateness* accordingly in $O(n_B \log n_B)$ time. Particularly, if $B_i^{(h)}$ contains no job now, then it becomes a dummy batch and thus we delete the lateness value of $B_i^{(h)}$ from *Hblateness*.

Since this case occurs at most $O(n_B)$ times, it takes $O({n_B}^2 \log n_B)$ time to deal with it.

**Subcase 2.2.2.** There is a dummy batch which succeeds $B_i^{(h)}$. After we move all the $B$-jobs in $B_i^{(h)}$ with their due dates no more than $d_k^B$ into $B_{n_B+1}^{(h)}$, we need to update the lateness value of $B_{n_B+1}^{(h)}$ in *Hblateness*. Moreover, if $B_i^{(h)}$ contains no job after the jobs are moved, then it becomes a dummy batch and thus we delete the lateness value of $B_i^{(h)}$ from *Hblateness*. Since this case occurs at most $O(n_B)$ times, it takes $O({n_B}^2)$ time to deal with it.

The array *Abcompl* and max-heap *Hblateness* have sizes of $O(n_B)$. Hence, we get:

**Theorem 2.6.** *A careful implementation of Algorithm BATCHCO leads to an $O(n_A + {n_B}^2 \log n_B)$-time algorithm for $1|s-batch, b \geq n, co, batch-avail|(C^A_{\max}, L^B_{\max})$ with $O(n_B)$ memory requirements.*

With respect to the incompatible model, recall that we always represent a schedule by a batch sequence $B_1, B_2, \ldots, B_{n_B}, B_{n_B+1}, \ldots, B_{2n_B+1}$, where $B_{n_B+1}$ is the batch containing all $A$-jobs. Note that batch $B_{n_B+1}$ can no longer contain any $B$-job. Therefore, we simply treat batch $B_{n_B}$ as the left adjacent batch of $B_{n_B+2}$ so that $B_{n_B+1}$ can be bypassed. When we move the $B$-jobs from right to the left, those jobs moved out of batch $B_{n_B+2}$ should be directly put into batch $B_{n_B}$. With such a slight modification, the above algorithm and its analysis work properly for the incompatible model. Therefore, we get:

**Theorem 2.7.** *There is an $O(n_A + {n_B}^2 \log n_B)$-time algorithm for $1|s-batch, b \geq n, inco, batch-avail|(C^A_{\max}, L^B_{\max})$ with $O(n_B)$ memory requirements.*

## 3. The item availability

In this section we will present $O(n_A+n_B \log n_B)$-time algorithms for the compatible and incompatible models under item availability, i.e., $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$ and $1|s-batch, b \geq n, inco, item-avail|(C^A_{\max}, L^B_{\max})$.

For the compatible model, we have the following lemma:

**Lemma 3.1.** *For any Pareto optimal point of $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$, there is a corresponding Pareto optimal schedule with the following properties:*

*(1) All jobs are contained in a single batch;*

*(2) All A-jobs are processed consecutively, i.e., they form a block such that no B-job is scheduled between two A-jobs;*

*(3) All B-jobs are scheduled in EDD order.*

*Proof.* For any two consecutive batches, we can merge them into one batch with only a single setup retained. Repeat this process and we will obtain a schedule satisfying property (1), without increasing $C^A_{\max}$ or $L^B_{\max}$.

Fix the position of the last $A$-job in this single-batch schedule. Move all the earlier $A$-jobs to later such that all $A$-jobs are processed consecutively, without increasing $C^A_{\max}$ or $L^B_{\max}$. This schedule thus satisfies properties (1) and (2).

Fix a Pareto optimal schedule satisfying properties (1) and (2). If there are two $B$-jobs such that the earlier one has a larger due date than the later one, then we reschedule the $B$-job with a larger due date to let it complete immediately after the $B$-job with a smaller due date. Repeat this process and finally, we will obtain the desired Pareto optimal schedule.

□

Re-index all $B$-jobs in EDD order so that $d^B_1 \leq d^B_2 \leq \cdots \leq d^B_{n_B}$. By Lemma 3.1, there are $O(n)$ Pareto optimal schedules, each for an intermediate position of the block of $A$-jobs. Based on the idea, we present Algorithm ITEMCO below which solves $1|s-batch, b \geq n, co, item-avail|(C^A_{\max}, L^B_{\max})$ easily.

**Algorithm ITEMCO:**

Step 1. Let $\sigma^{(0)}$ be the initial schedule consisting of a single batch. The $A$-jobs in the batch are first processed in arbitrary order, and the $B$-jobs in the batch are then processed in EDD order. Let $\Omega(\mathcal{J}) = \{(C^A_{\max}(\sigma^{(0)}), L^B_{\max}(\sigma^{(0)}), \sigma^{(0)})\}$.

Step 2. For $j = 1, 2, \ldots, n_B$, perform the $j$-th iteration to get schedule $\sigma^{(j)}$: Move job $J_j^B$ earlier such that it starts immediately before the block of $A$-jobs. If $L_{\max}^B(\sigma^{(j)}) < L_{\max}^B(\sigma^{(j-1)})$, then let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(C_{\max}^A(\sigma^{(j)}), L_{\max}^B(\sigma^{(j)}), \sigma^{(j)})\}$.

We get:

**Theorem 3.2.** *Algorithm ITEMCO solves* $1|s-batch, b \geq n, co, item-avail|(C_{\max}^A, L_{\max}^B)$ *in* $O(n_A + n_B \log n_B)$ *time.*

For the incompatible model, we have the following lemma:

**Lemma 3.3.** *For any Pareto optimal point of* $1|s-batch, b \geq n, inco, item-avail|(C_{\max}^A, L_{\max}^B)$*, there is a corresponding Pareto optimal schedule with the following properties:*

*(1) All A-jobs belong to a common batch;*

*(2) All B-jobs are scheduled in EDD order;*

*(3) There are at most two B-batches and they are separated by the batch containing all A-jobs.*

Re-index all $B$-jobs in EDD order so that $d_1^B \leq d_2^B \leq \cdots \leq d_{n_B}^B$. By Lemma 3.3, there are $O(n)$ Pareto optimal schedules, each for an intermediate position of the batch containing all $A$-jobs. Based on the idea, we present Algorithm ITEMINCO below which solves $1|s-batch, b \geq n, inco, item-avail|(C_{\max}^A, L_{\max}^B)$ easily.

**Algorithm ITEMINCO:**

Step 1. Let $\sigma^{(0)} = (B_1, B_2, B_3)$ be the initial schedule, where $B_1$ is an empty batch, $B_2$ consists of all $A$-jobs, and $B_3$ consists of all $B$-jobs. The $B$-jobs in $B_3$ are processed in EDD order. Let $\Omega(\mathcal{J}) = \{(C_{\max}^A(\sigma^{(0)}), L_{\max}^B(\sigma^{(0)}), \sigma^{(0)})\}$.

Step 2. For $j = 1, 2, \ldots, n_B$, perform the $j$-th iteration to get schedule $\sigma^{(j)}$: Move job $J_j^B$ from $B_3$ into $B_1$. If $B_1$ is empty before $J_j^B$ is moved, then a setup incurs after $J_j^B$ is moved into $B_1$. If $L_{\max}^B(\sigma^{(j)}) < L_{\max}^B(\sigma^{(j-1)})$, then let $\Omega(\mathcal{J}) = \Omega(\mathcal{J}) \cup \{(C_{\max}^A(\sigma^{(j)}), L_{\max}^B(\sigma^{(j)}), \sigma^{(j)})\}$.

We get:

**Theorem 3.4.** *Algorithm ITEMINCO solves* $1|s-batch, b \geq n, inco, item-avail|(C_{\max}^A, L_{\max}^B)$ *in* $O(n_A + n_B \log n_B)$ *time.*

## 4. Conclusions

In this paper, we studied the two-agent scheduling problem on an unbounded serial-batch machine to minimize the makespan of agent $A$ and the maximum lateness of agent $B$ simultaneously. We presented improved algorithms for compatible and incompatible models under batch availability and item availability assumptions. For future research, it is interesting to consider the combinations of general min-max and min-sum criteria, for example, $(f_{\max}^A, f_{\max}^B)$, $(f_{\max}^A, \sum C_j^B)$ and $(\sum C_j^A, \sum C_j^B)$. The bounded batch scheduling model with multiple agents can also be considered.

## Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

## Acknowledgments

This work is supported by Natural Science Foundation of Shandong Province China (No. ZR2020MA030).

## Conflict of interest

The authors declare there is no conflict of interest.

## References

1. A. Agnetis, J. C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, *Multiagent scheduling: models and algorithms*, Berlin Heidelberg: Springer, 2014.
2. C. N. Potts, M. Y. Kovalyov, Scheduling with batching: a review, *Eur. J. Oper. Res.*, **120** (2000), 228–249. https://doi.org/10.1016/S0377-2217(99)00153-8
3. J. W. Fowler, L. Monch, A survey of scheduling with parallel batch (p-batch) processing, *Eur. J. Oper. Res.*, **298** (2022), 1–24.
4. R .L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math.*, **5** (1979), 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X
5. V. T'Kindt, J. C. Billaut, Multicriteria scheduling: theory, models and algorithms, second edition, Berlin: Springer Verlag, 2006.
6. H. Hoogeveen, Multicriteria scheduling, *Eur. J. Oper. Res.*, **167** (2005), 592–623. https://doi.org/10.1016/j.ejor.2004.07.011
7. A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, *Eur. J. Oper. Res.*, **246** (2015), 345–378. https://doi.org/10.1016/j.ejor.2015.04.004
8. K. R. Baker, J. Cole Smith, A multiple-criterion model for machine scheduling, *J. Scheduling*, **6** (2003), 7–16.
9. A. Agnetis, P. B. Mirchandani, D. Pacciarelli, A. Pacifici, Scheduling problems with two competing agents, *Oper. Res.*, **52** (2004), 229–242. https://doi.org/10.1016/j.ejor.2015.04.004
10. P. Perez-Gonzalez, J. M. Framinan, A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: multi-agent scheduling problems, *Eur. J. Oper. Res.*, **235** (2014), 1–16.
11. S. Webster, K. R. Baker, Scheduling groups of jobs on a single machine, *Oper. Res.*, **43** (1995), 692–703. https://doi.org/10.1287/opre.43.4.692
12. A. P. M. Wagelmans, A. E. Gerodimos, Improved dynamic programs for some batching problems involving the maximum lateness criterion, *Oper. Res. Lett.*, **27** (2000), 109–118. https://doi.org/10.1016/S0167-6377(00)00040-7
13. C. He, Y. Lin, J. Yuan, Bicriteria scheduling of minimizing maximum lateness and makespan on a serial-batching machine, *Found. Comput. Decis. S.*, **33** (2008), 369–376.

14. C. He, H. Lin, Y. Lin, J. Tian, Bicriteria scheduling on a series-batching machine to minimize maximum cost and makespan, *Cent. Eur. J. Oper. Res.*, **21** (2013), 177–186. https://doi.org/10.1007/s10100-011-0220-9

15. C. He, H. Lin, Y. Lin, Bounded serial-batching scheduling for minimizing maximum lateness and makespan, *Discrete Optim.*, **16** (2015), 70–75. https://doi.org/10.1016/j.disopt.2015.02.001

16. Z. Geng, J. Yuan, J. 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. https://doi.org/10.1016/j.cam.2017.10.002

17. M. Y. Kovalyov, A. Oulamara, A. Soukhal, Two-agent scheduling with agent specific batches on an unbounded serial batching machine, *J. Scheduling*, **18** (2015), 423–434. https://doi.org/10.1007/s10951-014-0410-0

18. Q. Feng, Z. Yu, W. Shang, Pareto optimization of serial-batching scheduling problems on two agents, *Proceedings of the 2011 International Conference on Advanced Mechatronic Systems*, (2011), 165–168.

19. C. He, C. Xu, H. Lin, Serial-batching scheduling with two agents to minimize makespan and maximum cost, *J. Scheduling*, **23** (2020), 609–617. https://doi.org/10.1007/s10951-020-00656-5

20. C. He, H. Lin, Improved algorithms for two-agent scheduling on an unbounded serial-batching machine, *Discrete Optim.*, **41** (2021), 100655. https://doi.org/10.1016/j.disopt.2021.100655

21. C. He, H. Lin, X. Han, Two-agent scheduling on a bounded series-batch machine to minimize makespan and maximum cost, *Discrete Appl. Math.*, **322** (2022), 94–101. https://doi.org/10.1016/j.dam.2022.08.001

22. C. He, S. S. Li, J. Wu, Simultaneous optimization scheduling with two agents on an unbounded serial-batching machine, *RAIRO-Oper. Res.*, **55** (2021), 3701–3714. https://doi.org/10.1051/ro/2021175

23. S. Li, T. Cheng, C. Ng, J. Yuan, Two-agent scheduling on a single sequential and compatible batching machine, *Nav. Res. Log.*, **64** (2017), 628–641. https://doi.org/10.1002/nav.21779

24. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, Cambridge: MIT press, 2022.


AIMS Press