Electronic Research Archive

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

ERA, 30(11): 4209–4219
DOI: 10.3934/era.2022213
Received: 20 May 2022
Revised: 15 September 2022
Accepted: 15 September 2022
Published: 21 September 2022

Research article

A linear time approximation scheme for scheduling unbounded batch machines with delivery times and inclusive processing set restrictions

Xiaofang Zhao and Shuguang Li*

College of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China

* **Correspondence:** Email: sgliytu@hotmail.com; Tel: +8618753509226.

Abstract: We consider the problem of scheduling jobs with delivery times and inclusive processing set restrictions on unbounded batch machines to minimize the maximum delivery completion time, which is equivalent to minimizing the maximum lateness from the optimization viewpoint. We develop a polynomial time approximation scheme for this strongly NP-hard problem that runs in linear time for any fixed accuracy requirement.

Keywords: scheduling; unbounded batch machines; inclusive processing set restrictions; maximum delivery completion time; polynomial time approximation scheme

1. Introduction

Machine scheduling problems with processing set restrictions have been extensively studied in the past few decades. In this class of problems, we are given a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ and a set of m parallel machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Each job j is associated with a subset of machines $\mathcal{M}_j \subseteq \mathcal{M}$, called its *processing set*, whose members are capable of processing job j. Each machine can process at most one job at a time. The goal is to find an optimal schedule where optimality is defined by some problem-dependent objective. The current research on scheduling problems with processing set restrictions is dominated by models with the makespan objective. Leung and Li [1,2] surveyed the state of the art of these problems.

It is well known that parallel machine scheduling problems are usually strongly NP-hard for standard objective functions (such as minimizing makespan, minimizing total weighted completion time), even with equal release times and without processing set restrictions. Most of the previous researches thus have concentrated the efforts on polynomial time approximation algorithms. The performance of an approximation algorithm can be measured by its approximation ratio. For a given instance $\mathcal{I}$ of a minimization problem and an approximation algorithm A, let $A(\mathcal{I})$ and $OPT(\mathcal{I})$ denote the objective value of the solution obtained by algorithm A and the optimal solution value, respectively, when applied to $\mathcal{I}$. If $A(\mathcal{I})/OPT(\mathcal{I}) \leq \rho$ for all $\mathcal{I}$, then we say that algorithm A has an *approximation ratio* ρ, and A is called a ρ-approximation algorithm for this problem. A *polynomial time approximation scheme* (PTAS) is an approximation algorithm which for any instance $\mathcal{I}$ and any fixed accuracy requirement $\varepsilon > 0$ produces a solution with a running time polynomial in the input size of $\mathcal{I}$ such that $A(\mathcal{I})/OPT(\mathcal{I}) \leq 1 + \varepsilon$ [3].

A particularly important special case of processing set restrictions that has received considerable attention is the so-called *inclusive* processing set restriction. In this case, for each pair $\mathcal{M}_{j_1}$ and $\mathcal{M}_{j_2}$, either $\mathcal{M}_{j_1} \subseteq \mathcal{M}_{j_2}$ or $\mathcal{M}_{j_2} \subseteq \mathcal{M}_{j_1}$. The problem of minimizing makespan with release times and inclusive processing set restrictions is denoted as $P|r_j, M_j(inclusive)|C_{\max}$, in the three-field notation of Graham et al. [4]. The special case of it where all jobs are released at the same time is denoted as $P|M_j(inclusive)|C_{\max}$. There are a number of algorithms for $P|M_j(inclusive)|C_{\max}$ with increasingly better approximation ratios: a $(2 - 1/(m - 1))$-approximation algorithm [5, 6], a 3/2-approximation algorithm [7], a 4/3-approximation algorithm and a PTAS [8]. Li and Wang [9] presented a PTAS for $P|r_j, M_j(inclusive)|C_{\max}$ which is based on dynamic programming.

Inclusive processing set restrictions have been extended to *bounded* batch machines. Each bounded batch machine M_i has a capacity $K_i < n$. Each job j has a size s_j, where $0 < s_j \leq \max_i K_i$. Several jobs can be processed simultaneously as a batch on machine M_i, provided that the total size of the jobs in the batch does not exceed K_i. The processing time of a batch is the largest processing time of all the jobs in the batch. This model is called *p-batch scheduling model* [10]. (There is a rich literature if for each job j we assume $0 < s_j \leq \min_i K_i$. See the survey papers [10–12].) Leung and Li [2] used $P|p - batch, M_j(inclusive)|C_{\max}$ to denote problem $P|M_j(inclusive)|C_{\max}$ with bounded batch machines. When all jobs have equal processing times, Wang and Leung [13] presented a 2-approximation algorithm for $P|p - batch, M_j(inclusive)|C_{\max}$ that runs in $O(n \log n)$ time. They also showed that, unless P=NP, for this special case there does not exist any polynomial time algorithm with approximation ratio better than 2. For $P|p - batch, M_j(inclusive)|C_{\max}$, Damodaran et al. [14] proposed a particle swarm optimization algorithm, and Jia et al. [15] presented a heuristic based on the First-Fit-Decreasing rule and a meta-heuristic based on Max-Min Ant System.

To the best of our knowledge, processing set restrictions have not been extended to *unbounded* batch machines. An unbounded batch machine can process up to B ($B \geq n$) jobs simultaneously as a batch, and the processing time of a batch is the largest processing time of all the jobs in the batch [16]. Although the machines have unbounded capacities, a job may not be assigned to some machines due to particular machine eligibility constraints which are determined by factors other than the machine capacities.

In this paper we consider the problem of scheduling jobs with release times and inclusive processing set restrictions on unbounded batch machines, with a more general objective than makespan minimization, i.e., minimizing the maximum delivery completion time. The problem we study can be formulated as follows. Given a set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ and a set of m unbounded batch machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$. Each unbounded batch machine can process up to B ($B \geq n$) jobs simultaneously as a batch, and the processing time of a batch is the largest processing time of all the jobs in the batch. Each job j is characterized by a quadruple of non-negative real numbers (r_j, a_j, p_j, q_j), where r_j is the *release time* before which job j cannot be processed, a_j is the *machine index* associated with job j which specifies the smallest index among the machines that can process job j, p_j and q_j are the *processing time* and *delivery time* of job j respectively. Job j can be processed by machine M_i if and only if $i \geq a_j$. The machines in $\{M_{a_j}, M_{a_j+1}, \ldots, M_m\}$ are called *eligible machines* for job j. If S_j denotes the time job j starts processing, it has been delivered at time $L_j = S_j + p_j + q_j$, which is called its *delivery completion time*. The goal is to find a schedule so as to minimize the *maximum delivery completion time*, $L_{\max} = \max_j L_j$. Using the notation of Graham et al. [4], we denote this problem as $P|r_j, M_j(inclusive), B \geq n|L_{\max}$. Makespan minimization is a special case of this problem where all $q_j = 0$.

From the optimization viewpoint, the scheduling problem with delivery times and the maximum delivery completion time objective is equivalent to one with *due dates* and the *maximum lateness* objective [17]. However, since the lateness of a job can be zero or even negative, the delivery-time model is preferable from the perspective of approximation algorithms [18].

Much research has been done on scheduling unbounded batch machines without processing set restrictions under various objective functions. We only mention the results with the maximum lateness (or the maximum delivery completion time) objective here. For the single machine case with equal release times, Brucker et al. [16] provided a dynamic programming algorithm that requires $O(n^2)$ time, and Wagelmans and Gerodimos [19] presented an improved algorithm that runs in $O(n \log n)$ time. For the single machine case with unequal release times, Cheng et al. [20] established its NP-hardness, and Bai et al. [21] developed a linear time approximation scheme. Liu et al. [22] proved that the problem of scheduling jobs with release times and deadlines on unbounded batch machines is strongly NP-hard. Since this problem is the decision version of $P|r_j, B \geq n|L_{\max}$ (scheduling jobs with release times on unbounded batch machines to minimize the maximum delivery completion time), $P|r_j, B \geq n|L_{\max}$ is strongly NP-hard. Liu et al. [22] also presented a PTAS for $P|r_j, B \geq n|L_{\max}$. For online scheduling jobs with delivery times on m unbounded batch machines, Fang et al. [23] presented an algorithm with competitive ratio $1.5 + o(1)$, and Liu and Lu [24] presented an algorithm with competitive ratio $1 + 2/\lfloor\sqrt{m}\rfloor$.

Since $P|r_j, B \geq n|L_{\max}$ is a special case of $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ studied in this paper (scheduling jobs with release times and inclusive processing set restrictions on unbounded batch machines to minimize the maximum delivery completion time), $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ is also strongly NP-hard. In this paper we present a linear time approximation scheme for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$. This result is the best possible in two regards: it achieves the best possible approximation ratio for this strongly NP-hard problem; and it has optimum asymptotic time complexity. This result is inspired by [9,22,25,26]. For the classical scheduling problem (each machine can process at most one job at a time) of minimizing the maximum delivery completion time with release times on parallel machines (without processing set restrictions), Hall and Shmoys [25] presented a PTAS with running time $O(n^{\text{poly}(1/\varepsilon)})$, and Mastrolilli [26] developed another one with running time $O(n + f(1/\varepsilon))$, where $f(1/\varepsilon)$ is a constant that depends exponentially on $1/\varepsilon$.

This paper is organized as follows. In Section 2 we describe several input transformations so that the input has certain nice structure. In Section 3, we develop a linear time approximation scheme for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$. We conclude this paper in Section 4.

2. Simplifying the input

In this section we aim to transform any instance of $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ into one with a simpler structure. This will make the subsequent dynamic program more efficient.

A released job j is *available* for machine M_i if $i \geq a_j$ and j has not been assigned to any machine. Let $q(B_g) = max\{q_j|j \in B_g\}$ denote the delivery time of batch B_g. Let $a(B_g) = max\{a_j|j \in B_g\}$ denote the machine index associated with batch B_g. Let $\mathcal{J}_i = \{j \in \mathcal{J}|a_j = i\}$, $i = 1, 2, \ldots, m$. We have $\mathcal{J} = \cup_{i=1}^{m} \mathcal{J}_i$. Let OPT be the objective value of an optimal schedule. Let $r_{\max} = \max_j r_j$, $p_{\max} = \max_j p_j$, $q_{\max} = \max_j q_j$. Clearly we have $OPT \geq r_{\max}$, $OPT \geq p_{\max}$, $OPT \geq q_{\max}$.

Let ε be an arbitrary small rational number. For simplicity, we assume that $0 < \varepsilon < 1$ and $1/\varepsilon$ is integral. We will perform several transformations each of which may potentially increase the objective value by a factor of $1 + O(\varepsilon)$. When we describe this type of transformation, we shall say it produces $1 + O(\varepsilon)$ *loss*.

There is a trivial 3-approximation algorithm for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$: wait until time $r_{\max}$ and schedule all the jobs in one batch on machine M_m. Let UB be the objective value of this schedule. We have $UB \leq r_{\max} + p_{\max} + q_{\max} \leq 3OPT$. Let $LB = \max\{r_{\max}, p_{\max}, q_{\max}, UB/3\}$. We have

$$LB \leq OPT \leq 3LB. \tag{2.1}$$

Let $\delta = \varepsilon \cdot LB$. Following [25], we can assume there are a constant number of different release times and delivery times, as the following lemma states.

Lemma 2.1. *With* $1 + 2\varepsilon$ *loss, we assume that there are at most* $1/\varepsilon + 1$ *different release times and* $1/\varepsilon + 1$ *different delivery times.*

Proof. For each job j we set $r'_j = \lfloor r_j/\delta \rfloor \cdot \delta$, $q'_j = \lfloor q_j/\delta \rfloor \cdot \delta$. Since $r_{\max} \leq (1/\varepsilon)\delta$ and $q_{\max} \leq (1/\varepsilon)\delta$, there are at most $1/\varepsilon + 1$ different release times and at most $1/\varepsilon + 1$ different delivery times in the rounded instance. Since the values are rounded down, the optimal objective value of the rounded instance is not greater than OPT. Given a schedule for the rounded instance, we add δ to each job's start time, and re-introduce the original delivery times. Therefore, we get a feasible schedule for the original instance. We may increase the objective value by $2\delta \leq 2\varepsilon \cdot OPT$.

□

By the inequalities (2.1) we know $OPT \leq (3/\varepsilon)\delta$. Divide the interval $[0, (3/\varepsilon)\delta]$ into intervals $\Delta_1, \Delta_2, \ldots, \Delta_{1/\varepsilon+1}$, where $\Delta_k = [(k-1)\delta, k\delta)$ denotes the k-th interval for $k = 1, 2, \ldots, 1/\varepsilon$, and $\Delta_{1/\varepsilon+1} = [(1/\varepsilon)\delta, (3/\varepsilon)\delta]$ denotes the last interval. Let $\rho_k = (k-1)\delta$ denote the k-th release time, $k = 1, 2, \ldots, 1/\varepsilon + 1$. Similarly, let $\xi_l = (l-1)\delta$ denote the l-th delivery time, $l = 1, 2, \ldots, 1/\varepsilon + 1$.

A *regular* scheduling criterion is one that is non-decreasing in the job completion times. Brucker et al. [16] proved that when all the jobs have equal release times, for minimizing any regular objective on an unbounded batch machine, there is an optimal schedule in which the jobs are processed in the *batch-SPT order*, i.e., for any two batches B_1 and B_2 in this schedule, if B_1 is processed before B_2, then there does not exist two jobs j and j' such that $j \in B_1$, $j' \in B_2$ and $p_j > p_{j'}$. On the other hand, recall that when all the jobs have equal release times, Jackson's rule [27] solves the classical scheduling problem of minimizing the maximum lateness on a single machine optimally: Schedule the jobs without idle time in order of non-increasing delivery times. Based on these two results, we get the following lemma.

Lemma 2.2. *There is an optimal schedule for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ with the following properties:*

for $i = 1, 2, \ldots, m$ (This ordering is used crucially, which means that we handle the machines in increasing order of their indices),

(i) on machine M_i, the batches started in Δ_k ($k = 1, 2, \ldots, 1/\varepsilon + 1$) are processed in strictly increasing order of batch processing times, and this order is also the strictly decreasing order of batch delivery times;

(ii) on machine M_i, the batches started in Δ_k ($k = 1, 2, \ldots, 1/\varepsilon + 1$) are filled in strictly increasing order of processing times such that each batch contains all currently available jobs with processing times no greater than the batch processing time and delivery times no greater than the batch delivery time.

Proof. Consider an optimal schedule for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$. We will transform it into another schedule (without increasing the objective value) that satisfies the properties described in the lemma. We perform the transformations by handling the machines in increasing order of their indices. Suppose that we have handled the machines $M_1, M_2, \ldots, M_{i-1}$. We now explain how to handle machine M_i.

We handle the intervals on M_i in increasing order of their indices. Suppose that we have handled the intervals $\Delta_1, \Delta_2, \ldots, \Delta_{k-1}$. We now explain how to handle Δ_k. Recall that the delivery time of batch B_g is defined to be the largest delivery time of all the jobs in B_g. Let $B_1, B_2, \ldots, B_x$ denote the batches which are processed on M_i and started in Δ_k. Among these batches, if there are two batches having the same delivery time, then we move all the jobs in the batch with smaller processing time into the batch with the larger processing time. Therefore, we can assume that the batches $B_1, B_2, \ldots, B_x$ have different delivery times.

Since all these batches are scheduled in the same interval, scheduling them in Δ_k is essentially an instance of the problem where all jobs have equal release times. Therefore, we can apply Jackson's rule to rearrange these batches and thereafter assume without loss generality that the batches $B_1, B_2, \ldots, B_x$ are processed successively on machine M_i one after another and $q(B_1) > q(B_2) > \cdots > q(B_x)$, where $q(B_h)$ denotes the delivery time of batch B_h, $h = 1, 2, \ldots, x$.

Among the batches $B_1, B_2, \ldots, B_x$, if there are two batches B_{h_1} and B_{h_2} ($1 \leq h_1 < h_2 \leq x$) such that $p(B_{h_1}) \geq p(B_{h_2})$, then we move all the jobs in B_{h_2} into B_{h_1}. Accordingly, the completion times of the jobs in B_{h_2} decrease, while the completion times of the jobs in other batches do not increase. A finite number of repetitions of this procedure yield an optimal schedule which satisfies property (i) described in the lemma (with respect to machine M_i and interval Δ_k).

We continue to transform the obtained schedule into the one satisfying property (ii) (with respect to machine M_i and interval Δ_k). To do so, we first delete all the jobs from the batches $B_1, B_2, \ldots, B_x$, but retain all the empty batches which are specified by the processing times and delivery times of $B_1, B_2, \ldots, B_x$. We fill these empty batches in strictly increasing order of processing times such that each batch contains all currently available jobs with processing times no greater than the batch processing time and delivery times no greater than the batch delivery time. Thereby we get an optimal schedule of the required form.

We repeat the procedure to handle the machines in increasing order of their indices, and within that to handle the intervals on each machine in increasing order of their indices. At the end, we will get an optimal schedule which satisfies the properties described in the lemma. □

Following [25], we classify both jobs and batches as large and small. Job j is *large* if $p_j \geq \delta/(1/\varepsilon+1)^2$, otherwise it is *small*. A batch is *large* if it contains at least a large job, otherwise it is *small*. We have:

Lemma 2.3. *With* $1+\varepsilon^2$ *loss, we assume that*

(i) $p_j = 0$ *for any small job* j;
(ii) no small job is included in large batches;
(iii) on each machine there is at most one small batch started in each interval.

Proof. Set $p_j = 0$ for any small job j. This will not increase the objective value. Given an optimal schedule for the transformed instance, for the batches started in the same interval on the same machine, we stretch an extra space of length $\delta/(1/\varepsilon+1)^2$ right before the first batch of these batches to reschedule all the small jobs started in this interval on this machine with the original processing times. Therefore, no small job is included in large batches, and on each machine there is at most one small batch started in each interval. Since there are $1/\varepsilon+1$ intervals, the objective value may increase by $\delta/(1/\varepsilon+1) \leq \varepsilon^2 \cdot OPT$. □

Lemma 2.4. *With* $1+\varepsilon$ *loss, the number of different processing times of large jobs,* λ, *can be bounded from above by* $(1+\varepsilon)/\varepsilon^4 - 1/\varepsilon + 1$.

Proof. Set $p'_j = \left\lfloor p_j/(\varepsilon\cdot\delta/(1/\varepsilon+1)^2)\right\rfloor \cdot (\varepsilon\cdot\delta/(1/\varepsilon+1)^2)$ for any large job j. This will not increase the objective value. Given an optimal schedule for the transformed instance, re-introduce the original processing times of large jobs. Since for large job j we have $p_j \geq \delta/(1/\varepsilon+1)^2$, the objective value may increase by $\varepsilon\cdot OPT$. Since $\delta/(1/\varepsilon+1)^2 \leq p_j \leq (1/\varepsilon)\delta$, we get $\lambda \leq (1+\varepsilon)/\varepsilon^4 - 1/\varepsilon + 1$. □

Thus, all large jobs have processing times of the form $h\cdot(\varepsilon\cdot\delta/(1/\varepsilon+1)^2)$, where $h \in \{1/\varepsilon, 1/\varepsilon+1, \ldots, (1+\varepsilon)/\varepsilon^4\}$. Without loss of generality, we assume $\gamma = (1+\varepsilon)/\varepsilon^4 - 1/\varepsilon + 1$. Let $P_t = (t+1/\varepsilon-1)\cdot(\varepsilon\cdot\delta/(1/\varepsilon+1)^2)$ denote the t-th processing time of the large jobs, $t \in \{1,2,\ldots,\gamma\}$. In addition, let $P_0 = 0$ denote the processing time zero of all the small jobs.

Let n_{klti} be the number of jobs with release time ρ_k, delivery time ξ_l, processing time P_t and machine index i, $k = 1,2,\ldots,1/\varepsilon+1$, $l = 1,2,\ldots,1/\varepsilon+1$, $t = 0,1,\ldots,\lambda$, $i = 1,2,\ldots,m$. Let $n_{klt}(i) = \sum_{h=1}^{i} n_{klth}$. These values can be computed in $O(n+m/\varepsilon^6)$ time. (Note that $((1+\varepsilon)/\varepsilon^4 - 1/\varepsilon+2)\cdot(1/\varepsilon+1)^2 \leq 12/\varepsilon^6$.)

3. A linear time approximation scheme

In this section we will present a linear time approximation scheme for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$.

Consider an optimal schedule Σ that satisfies Lemmas 2.1, 2.2, 2.3 and 2.4. (After the rounding and before the re-introducing, the objective value of the rounded instance is not greater than OPT.) By Lemma 2.2, we deal with the machines in increasing order of their indices. Let $n'_{klt}(i)$ be the number of available jobs for machine M_i with release time ρ_k, delivery time ξ_l and processing time P_t, $k = 1,2,\ldots,1/\varepsilon+1$, $l = 1,2,\ldots,1/\varepsilon+1$, $t = 0,1,\ldots,\lambda$, $i = 1,2,\ldots,m$. Let x_{klti} denote the number of jobs with release time ρ_k, delivery time ξ_l and processing time P_t that are assigned to machine M_i in Σ. It must be true that $x_{klti} \leq n'_{klt}(i)$, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $l = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 0, 1, \ldots, \lambda$, $i = 1, 2, \ldots, m$. We call the set $\{x_{klti} | \forall k, l, t\}$ an *assignment* for machine M_i, $i = 1, 2, \ldots, m$.

We then delete from Σ all the jobs, but retain all the empty batches. Let Y_{klti} be the set of the empty batches in Σ that are started in interval Δ_k with delivery time ξ_l and processing time P_t on machine M_i, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $l = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 0, 1, \ldots, \lambda$, $i = 1, 2, \ldots, m$. Let $y_{klti} = |Y_{klti}|$. By Lemmas 2.2 and 2.3, we have $\sum_{l=1}^{1/\varepsilon+1} y_{klti} \leq 1$, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 0, 1, \ldots, \lambda$, $i = 1, 2, \ldots, m$. We call the set $\{y_{klti} | \forall k, l, t\}$ a *configuration* for machine M_i, $i = 1, 2, \ldots, m$.

Example:

We now demonstrate an example to illustrate the techniques and definitions we discussed so far. We start with a schedule that satisfies Lemma 2.1. Let $LB = 18$ and $\varepsilon = 1/2$. We have: $\delta = \varepsilon \cdot LB = 9$. By Lemma 2.1, there are three different release times: $\rho_1 = 0$, $\rho_2 = 9$, $\rho_3 = 18$, and three different delivery times: $\xi_1 = 0$, $\xi_2 = 9$, $\xi_3 = 18$. The three time intervals on a machine are: $\Delta_1 = [0, 9)$, $= [9, 18)$, and $\Delta_3 = [18, 54)$. Consider an optimal schedule Σ for $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ which satisfies Lemma 2.2. Let us fix a particular machine M_i. In Σ, there are three batches started in Δ_1 and processed on M_i: B_1, B_2 and B_3; there are only one batch started in Δ_2 and processed on M_i: B_4; there are two batches started in Δ_3 and processed on M_i: B_5 and B_6. Batch B_1 starts at time 0 and completes at time 1/4, which consists of two jobs j_1 and j_2 with $r_{j_1} = r_{j_2} = \rho_1$, $p_{j_1} = 1/4$, $p_{j_2} = 1/6$, $q_{j_1} = \xi_3$, $q_{j_2} = \xi_2$. Batch B_2 starts at time 1/4 and completes at time 3/4, which consists of two jobs j_3 and j_4 with $r_{j_3} = r_{j_4} = \rho_1$, $p_{j_3} = p_{j_4} = 1/2$, $q_{j_3} = \xi_2$, $q_{j_4} = \xi_1$. Batch B_3 starts at time 3/4 and completes at time $10\frac{3}{4}$, which consists of two jobs j_5 and j_6 with $r_{j_5} = r_{j_6} = \rho_1$, $p_{j_5} = p_{j_6} = 10$, $q_{j_5} = q_{j_6} = \xi_1$. Batch B_4 starts at time $10\frac{3}{4}$ and completes at time $19\frac{3}{4}$, which consists of two jobs j_7 and j_8 with $r_{j_7} = r_{j_8} = \rho_2$, $p_{j_7} = 9$, $p_{j_8} = 5$, $q_{j_7} = \xi_3$, $q_{j_8} = \xi_2$. Batch B_5 starts at time $19\frac{3}{4}$ and completes at time 20 , which consists of two jobs j_9 and j_{10} with $r_{j_9} = \rho_2$, $r_{j_{10}} = \rho_3$, $p_{j_9} = p_{j_{10}} = 1/4$, $q_{j_9} = \xi_3$, $q_{j_{10}} = \xi_2$. Batch B_6 starts at time 20 and completes at time $20\frac{1}{2}$, which consists of two jobs j_{11} and j_{12} with $r_{j_{11}} = \rho_2$, $r_{j_{12}} = \rho_3$, $p_{j_{11}} = p_{j_{12}} = 1/2$, $q_{j_{11}} = q_{j_{12}} = \xi_2$.

Recall that $\delta/(1/\varepsilon + 1)^2 = 9/(1/(1/2) + 1)^2 = 1$ represents the threshold for classifying large and small jobs. Thus, jobs $j_1, j_2, j_3, j_4, j_9, j_{10}, j_{11}, j_{12}$ are small jobs, and jobs j_5, j_6, j_7, j_8 are large jobs. By Lemma 2.3, we round the processing times of all the small jobs down to zero. By Lemma 2.4, we round the processing times of large jobs down to the nearest integral multiple of $\varepsilon \cdot \delta/(1/\varepsilon + 1)^2 = 1/2$. (Certainly, for M_i we need not to round down the processing times of the large jobs, because all the large jobs processed on M_i have integral processing times.) Recall that $P_0 = 0$, and $P_t = (t + 1/\varepsilon - 1) \cdot (\varepsilon \cdot \delta/(1/\varepsilon + 1)^2) = (t + 1)/2$ denotes the t-th processing time of the large jobs, $t \in \{1, 2, \ldots, \gamma\}$. Therefore, the processing times of the jobs processed on M_i can be labeled as $p_{j_1} = p_{j_2} = P_0$, $p_{j_3} = p_{j_4} = P_0$, $p_{j_5} = p_{j_6} = P_{19}$, $p_{j_7} = P_{17}$, $p_{j_8} = P_9$, $p_{j_9} = p_{j_{10}} = P_0$, $p_{j_{11}} = p_{j_{12}} = P_0$.

We are now ready to determine the assignment and the configuration for M_i. We have: $x_{1,1,0,i} = 1$ (job j_4), $x_{1,2,0,i} = 2$ (jobs j_2 and j_3), $x_{1,3,0,i} = 1$ (job j_1), $x_{2,1,0,i} = 0$, $x_{2,2,0,i} = 1$ (job j_{11}), $x_{2,3,0,i} = 1$ (job j_9), $x_{3,1,0,i} = 0$, $x_{3,2,0,i} = 2$ (jobs j_{10} and j_{12}), $x_{3,3,0,i} = 0$. We also have: $x_{1,1,19,i} = 2$ (jobs j_5 and j_6), $x_{2,3,17,i} = 1$ (job j_7), $x_{2,2,9,i} = 1$ (job j_8). All other values of x_{klti} are equal to zero, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $l = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 1, 2, \ldots, \lambda$. Thereby, we get the set $\{x_{klti} | \forall k, l, t\}$, which is the assignment for machine M_i with respect to Σ.

Furthermore, we have: $Y_{1,3,0,i} = \{\bar{B}_1\}$, $Y_{1,2,0,i} = \{\bar{B}_2\}$, $Y_{1,1,19,i} = \{\bar{B}_3\}$, $Y_{2,3,17,i} = \{\bar{B}_4\}$, $Y_{3,3,0,i} = \{\bar{B}_5\}$, $Y_{3,2,0,i} = \{\bar{B}_6\}$, where $\bar{B}_h$ denotes the empty batch represented by the processing time of B_h,

$h = 1, 2, \ldots, 6$. We get: $y_{1,3,0,i} = 1$, $y_{1,2,0,i} = 1$, $y_{1,1,19,i} = 1$, $y_{2,3,17,i} = 1$, $y_{3,3,0,i} = 1$, $y_{3,2,0,i} = 1$. All other values of y_{klti} are equal to zero, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $l = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 0, 1, \ldots, \gamma$. Thereby, we get the set $\{y_{klti}|\forall k, l, t\}$, which is the configuration of machine M_i with respect to Σ.

Let Σ_i denote the restriction of Σ to machine M_i, $i = 1, 2, \ldots, m$. Given the assignment and the configuration for M_i, we can recover Σ_i by Lemma 2.2: for $k = 1, 2, \ldots, 1/\varepsilon + 1$, fill the empty batches in $\cup_{l=1}^{1/\varepsilon+1} \cup_{t=0}^{\lambda} Y_{klti}$ in strictly increasing order of processing times (this order is also the strictly decreasing order of batch delivery times except for the possible empty small batch) such that each batch contains all currently available jobs with processing times no greater than the batch processing time and delivery times no greater than the batch delivery time. Since $OPT \leq (3/\varepsilon)\delta$, Σ_i contains at most $(3/\varepsilon)\delta/(\delta/(1/\varepsilon + 1)^2) = 3(1/\varepsilon + 1)^2/\varepsilon$ empty large batches and at most $1/\varepsilon + 1$ empty small batches. Thus, the recovery of Σ_i can be done in $O((3(1/\varepsilon + 1)^2/\varepsilon + 1/\varepsilon + 1) \cdot (1/\varepsilon + 1)^2) = O((1/\varepsilon)^7)$ time.

We enumerate the possible configurations for M_i as follows. We have shown that Σ_i contains at most $(3/\varepsilon)\delta/(\delta/(1/\varepsilon + 1)^2) = 3(1/\varepsilon + 1)^2/\varepsilon$ empty large batches and at most $1/\varepsilon + 1$ empty small batches. The processing times of large batches are chosen from $\lambda \leq (1 + \varepsilon)/\varepsilon^4 - 1/\varepsilon + 1$ different values. The delivery times of all the batches are chosen from $1/\varepsilon + 1$ values. Hence, the number of different configurations for machine M_i ($i = 1, 2, \ldots, m$) can be bounded by $(\lambda(1/\varepsilon + 1) + 1)^{3(1/\varepsilon+1)^2/\varepsilon} \cdot (1/\varepsilon + 2)^{(1/\varepsilon+1)} \leq (5/\varepsilon^5)^{14/\varepsilon^3}$.

Let $D(\{x_{klti}|\forall k, l, t\})$ be the maximum delivery completion time of the jobs processed on M_i if assignment $\{x_{klti}|\forall k, l, t\}$ is adopted. From the above analysis, $D(\{x_{klti}|\forall k, l, t\})$ can be computed in $O((1/\varepsilon)^7 \cdot (5/\varepsilon^5)^{14/\varepsilon^3}) = O((5/\varepsilon^5)^{14/\varepsilon^3+2})$ time.

Let v_{klti} denote the number of jobs with release time ρ_k, delivery time ξ_l and processing time P_t that are assigned to machines $M_1, M_2, \ldots, M_i$ in Σ. It must be true that $v_{klti} \leq n_{klt}(i)$, $k = 1, 2, \ldots, 1/\varepsilon + 1$, $l = 1, 2, \ldots, 1/\varepsilon + 1$, $t = 0, 1, \ldots, \lambda$, $i = 1, 2, \ldots, m$. For $i = 1, 2, \ldots, m$, we call the set $\{v_{klti}|\forall k, l, t\}$ an *outline* for machines $M_1, M_2, \ldots, M_i$.

Let V_i be the set of all possible outlines for machines $M_1, M_2, \ldots, M_i$, $i = 1, 2, \ldots, m$. Unbounded batch machines have the following property: for any pair of values of l and t, if $v_{ulti} > 0$ for some $u \in \{1, 2, \ldots, 1/\varepsilon + 1\}$, then $v_{klti} = n_{klt}(i)$ for all $k \in \{1, 2, \ldots, u\}$. Hence, we get $|V_i| \leq (1/\varepsilon + 2)^{(1/\varepsilon+1)(\lambda+1)} \leq (1/\varepsilon + 2)^{6/\varepsilon^5}$, $i = 1, 2, \ldots, m$.

To solve $P|r_j, M_j(inclusive), B \geq n|L_{\max}$, we generalize the dynamic programming approach presented in [9] for $P|r_j, M_j(inclusive)|C_{\max}$.

For each $\{v_{klti}|\forall k, l, t\} \in V_i$ ($i = 1, 2, \ldots, m$), Let $F_i(\{v_{klti}|\forall k, l, t\})$ denote the minimum possible objective value if we use the outline $\{v_{klti}|\forall k, l, t\}$ for machines $M_1, M_2, \ldots, M_i$. We are interested in the schedules with objective value at most $(3/\varepsilon)\delta$. Therefore, for machines $M_1, M_2, \ldots, M_i$, we only need to consider the outlines with objective value no greater than $(3/\varepsilon)\delta$. Let $W_i \subseteq V_i$ be the set of these outlines for machines $M_1, M_2, \ldots, M_i$, $i = 1, 2, \ldots, m$.

We have the following recurrence relation:

$$F_{i+1}(\{v_{klt(i+1)}|\forall k, l, t\}) = \min_{w_{klti} \leq v_{klt(i+1)}} \{max\{D(\{v_{klt(i+1)} - w_{klti}|\forall k, l, t\}), F_i(\{w_{klti}|\forall k, l, t\})\}\}.$$

The boundary conditions are:

(i) $F_1(\{v_{klt1}|\forall k, l, t\}) = D(\{v_{klt1}|\forall k, l, t\})$ if $\{v_{klt1}|\forall k, l, t\} \in W_1$, and $F_1(\{v_{klt1}|\forall k, l, t\}) = +\infty$ otherwise.

(ii) $F_i(\{v_{klti}|\forall k, l, t\}) = +\infty$ if $\{v_{klti}|\forall k, l, t\} \notin W_i$.

Finally, the optimal objective value is given by $F_m(\{n_{klt}(m)|\forall k, l, t\})$.

The values of n_{klti} and $n_{klt}(i)$ for all k, l, t, i can be computed in $O(n + m/\varepsilon^6)$ time. Computing $D(\{v_{klt(i+1)} - w_{klti} | \forall k, l, t\})$ takes $O((1/\varepsilon)^7 \cdot (5/\varepsilon^5)^{14/\varepsilon^3}) = O((5/\varepsilon^5)^{14/\varepsilon^3+2})$ time. Executing the dynamic program takes time $O(m \cdot (5/\varepsilon^5)^{14/\varepsilon^3+2} \cdot |V_m|^2) = O(m \cdot (5/\varepsilon^5)^{14/\varepsilon^3+2} \cdot ((1/\varepsilon + 2)^{6/\varepsilon^5})^2) = O(m \cdot (5/\varepsilon^5)^{14/\varepsilon^3+2} \cdot (1/\varepsilon + 2)^{12/\varepsilon^5})$. Hence, the overall running time of the algorithm is $O(n + m \cdot (5/\varepsilon^5)^{14/\varepsilon^3+2} \cdot (1/\varepsilon + 2)^{12/\varepsilon^5})$.

Therefore we get:

Theorem 3.1. *Problem $P|r_j, M_j(inclusive), B \geq n|L_{\max}$ admits a PTAS that runs in linear time for any fixed accuracy requirement.*

4. Conclusions

In this paper we initiated the study of scheduling jobs with release times, delivery times and inclusive processing set restrictions on unbounded batch machines. The objective is to minimize the maximum delivery completion time. For this strongly NP-hard problem, we presented a linear time approximation scheme. For future research, we can study this problem for other objective functions, such as minimizing total weighted completion time. Moreover, we admit that the proposed PTAS still has a bad dependence on ε. As pointed out by Mnich and Wiese [28], in practice exact algorithms are often desired and it would be interesting to investigate fixed-parameter scheduling. If we take the inverse of ε as a parameter, then what we developed can be seen as a fixed-parameter algorithm (for this parameter). The idea in fixed-parameter algorithms is to accept exponential running times, which are seemingly inevitable in solving NP-hard problems, but to restrict them to certain aspects of the problem, which are captured by parameters [29]. In future research, we can use the techniques of fixed-parameter scheduling to study problems of interest.

Acknowledgments

We thank the editor and reviewers for their helpful suggestions. 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. J. Y. T. Leunga, C. Li, Scheduling with processing set restrictions: A survey, *Int. J. Prod. Econ.*,**116** (2008), 251–262. https://doi.org/10.1016/j.ijpe.2008.09.003
2. J. Y. T. Leunga, C. Li, Scheduling with processing set restrictions: A literature update, *Int. J. Prod. Econ.*, **175** (2016), 1–11. https://doi.org/10.1016/j.ijpe.2014.09.038
3. C. H. Papadimitriou, K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Courier Dover Publications, 1998.
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. D. G. Kafura, V. Y. Shen, Task scheduling on a multiprocessor system with independent memories, *SIAM J. Comput.*, **6** (1977), 167–187. https://doi.org/10.1137/0206014

6. H. C. Hwang, S. Y. Chang , Y. Hong, A posterior competitiveness for list scheduling algorithm on machines with eligibility constraints, *Asia-Pacific J. Operat. Res.*, **21** (2004), 117–125. https://doi.org/10.1142/S0217595904000084

7. C. A. Glass, H. Kellerer, Parallel machine scheduling with job assignment restrictions, *Nav. Res. Log.*, **54** (2007), 250–257. https://doi.org/10.1002/nav.20202

8. J. Ou , J. Y. T. Leung, C. L. Li, Scheduling parallel machines with inclusive processing set restrictions, *Nav. Res. Log.*, **55** (2008), 328–338. https://doi.org/10.1002/nav.20286

9. C. L. Li, X. Wang, Scheduling parallel machines with inclusive processing set restrictions and job release times, *Eur. J. Oper. Res.*, **200** (2010), 702–710. https://doi.org/10.1016/j.ejor.2009.02.011

10. 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

11. M. Mathirajan, A. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, *Int. J. Adv. Manuf. Tech.*, **29** (2006), 990–1001. https://doi.org/10.1007/s00170-005-2585-1

12. J. W. Fowler, L. Mönch, A survey of scheduling with parallel batch (p-batch) processing, *Eur. J. Oper. Res.*, **299** (2022), 1–24. https://doi.org/10.1016/j.ejor.2021.06.012

13. J. Q. Wang, J. Y. T. Leung, Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan, *Int. J. Prod. Econ.*, **156** (2014), 325–331. https://doi.org/10.1016/j.ijpe.2014.06.019

14. P. Damodaran, D. A. Diyadawagamage, O. Ghrayeb, M. C. Velez-Gallego, A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines, *Int. J. Adv. Manuf. Tech.*, **58** (2012),1131–1140. https://doi.org/10.1007/s00170-011-3442-z

15. Z. Jia, K. Li, J. Y. T. Leung, Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities, *Int. J. Prod. Econ.*, **169** (2015), 1–10. https://doi.org/10.1016/j.ijpe.2015.07.021

16. P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, et al., Scheduling a batching machine, *J. Scheduling*, **1** (1998), 31–54. https://doi.org/10.1002/(SICI)1099-1425(199806)1:1¡31::AID-JOS4¿3.0.CO;2-R

17. J. Y. T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, 2004.

18. L. A. Hall, D. B. Shmoys, Jackson's rule for single-machine scheduling: Making a good heuristic better, *Math. Oper. Res.*, **17** (1992), 22–35. https://doi.org/10.1287/moor.17.1.22

19. 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

20. T. E. Cheng, Z. Liu, W. Yu, Scheduling jobs with release dates and deadlines on a batch processing machine, *IIE Trans.*, **33** (2001), 685–690. https://doi.org/10.1080/07408170108936864

21. S. Bai, F. Zhang, S. Li, Q. Liu, Scheduling an unbounded batch machine to minimize maximum lateness, *Front. Algorithmics*, (2007), 172–177. https://doi.org/10.1007/978-3-540-73814-5_16

22. L. L. Liu, C. T. Ng, T. C. E. Cheng, Scheduling jobs with release dates on parallel batch processing machines, *Discrete Appl. Math.*, **157** (2009), 1825–1830. https://doi.org/10.1016/j.dam.2008.12.012

23. Y. Fang, X. Lu, P. Liu, Online batch scheduling on parallel machines with delivery times, *Theor. Comput. Sci.*, **412** (2011), 5333–5339. https://doi.org/10.1016/j.tcs.2011.06.011

24. P. Liu, X. Lu, Online unbounded batch scheduling on parallel machines with delivery times, *J. Comb. Optim.*,**29** (2015), 228–236. https://doi.org/10.1007/s10878-014-9706-4

25. L. A. Hall, D. B. Shmoys, Approximation schemes for constrained scheduling problems, in *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, (1989), 134–139. https://doi.org/10.1109/SFCS.1989.63468

26. M. Mastrolilli, Efficient approximation schemes for scheduling problems with release dates and delivery times, *J. Scheduling*, **6** (2003), 521–531. https://doi.org/10.1023/A:1026272526225

27. J. R. Jackson, Scheduling a production line to minimize maximum tardiness, DTIC Document, 1955.

28. M. Mnich, A. Wiese, Scheduling and fixed-parameter tractability, *Math. Program.*, **154** (2015), 533–562. https://doi.org/10.1007/s10107-014-0830-9

29. M. Mnich, R. van Bevern, Parameterized complexity of machine scheduling: 15 open problems, *Comput. & Oper. Res.*, **100** (2018), 254–261. https://doi.org/10.1016/j.cor.2018.07.020