J. Oper. Res. Soc. China (2014) 2:499–510
DOI 10.1007/s40305-014-0062-y

Scheduling a Bounded Parallel-Batching Machine with Incompatible Job Families and Rejection

Shi-Sheng Li · Ren-Xia Chen

Received: 14 June 2014 / Revised: 6 November 2014 / Accepted: 12 November 2014 /
Published online: 30 November 2014

Abstract We study a scheduling problem with incompatible job families and rejection on a parallel-batching machine, where the objective is to minimize the makespan of all accepted jobs plus the total penalty of all rejected jobs. We provide a polynomial-time algorithm for the case where all jobs have identical release dates and a pseudo-polynomial-time algorithm for the case where the number of distinct release dates is fixed. We also present a 2-approximation algorithm and a polynomial-time approximation scheme for the general problem.

Keywords Parallel-batching scheduling · Incompatible job families · Rejection · Approximation algorithm

1 Introduction

We are given a set $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ of n jobs that are classified into m families $\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_m$. Each job $J_j\,(j = 1, 2, \cdots, n)$ is associated with an integral processing time p_j, an integral release date r_j, and an integral rejection penalty e_j. Job J_j is either accepted and then processed on the machine, or it is rejected by the machine and then a rejection penalty e_j is paid. The accepted jobs may be combined to form batches, where a batch is a set of jobs which are processed simultaneously. The jobs processed in the same batch have the same starting time and will have to

This research was supported in part by National Natural Science Foundation of China (NSFC, Nos. 11326191, 11401604, 11401605 and 11171313), NSF of Henan Province (No. 132300410392) and the Education Department of Henan Province Natural Science Research Program (No. 14A110027)

S.-S. Li (✉) · R.-X. Chen
College of Science, Zhongyuan University of Technology, Zhengzhou 450007, China
e-mail: shishengli96@163.com

wait until all jobs in the batch have been completed. Therefore, the processing time of a batch B, denoted by $p(B)$, is equal to the largest processing time of the jobs in it, i.e., $p(B) = \max\{p_j : J_j \in B\}$. This type of batching is referred to as *parallel-batching* or *p-batch*. This is different from *serial-batching* or *s-batch*, where jobs are processed sequentially with a setup time for each batch [2]. Each batch can contain at most b jobs and it can start processing only after each job in it is released. Due to the logistical and management constraints, we require that jobs from different families are not allowed to be processed together in any single batch. Let $\mathcal{A}$ be the set of jobs accepted, and $\mathcal{R} = \mathcal{J} \setminus \mathcal{A}$ be the set of jobs rejected. For a given schedule π, let C_j denote the completion time of job $J_j \in \mathcal{A}$. The problem is to determine $\mathcal{A}$ and a feasible schedule of jobs in $\mathcal{A}$ on the machine so as to minimize the makespan of the accepted jobs (i.e., $C_{\max}(\mathcal{A}) = \max\{C_j : J_j \in \mathcal{A}\}$) plus the total penalty of the rejected jobs. Using the traditional three-field notation for scheduling problems, we denote the problem under consideration by $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A}) + \sum_{J_j \in \mathcal{R}} e_j$. Here, *rej* implies that rejection is allowed, *p-batch* implies that the set of accepted jobs is to be processed on a parallel-batching machine, *family-jobs* imply that the job families are incompatible.

The above described scheduling problem falls into the category of parallel-batching scheduling with incompatible job families and the category of scheduling with rejection. Although these two categories of scheduling problems have been extensively studied in the literature; however, to the best of our knowledge, this subject has not been approached. Thus, our main aim is to consider the parallel-batching scheduling problem with incompatible job families and rejection allowed.

The parallel-batching scheduling model was initially motivated by the burn-in operations that are performed in ovens during the final testing stage of circuit board manufacturing (Lee et al. [13]). Other applications of such batching scheduling models have been found in many real-life industry applications such as the numerically controlled routers for cutting metal sheets or printed circuit boards, the diffusion area in semiconductor wafer fabrication facilities, heat treatment facilities in the ceramic industries, and certain chemical vapor deposition processes (e.g., Hochbaum and Landy [10], Dobson and Nambimadom [7], Mathirajan and Sivakumar [19]). Much work similar to the study of this paper without rejection has been reported in literature. For example, Lee and Uzosy [12] provided an $O(n^2)$ time algorithm for $1|p-batch, r_j, b=n|C_{\max}$. Brucker et al. [3] proved that $1|p-batch, r_j, b<n|C_{\max}$ is strongly NP-hard when the number of distinct release dates is arbitrary. Deng et al. [6] provided a polynomial-time approximation scheme (PTAS) for $1|p-batch, r_j, b<n|C_{\max}$. Yuan et al. [24] proved that $1|p-batch, family-jobs, r_j, b=n|C_{\max}$ is strongly NP-hard, and provided a PTAS. When the number of incompatible job families is fixed, Nong et al. [21] and Li and Yuan [14] presented PTASes for $1|p-batch, family-jobs, r_j, b<n|C_{\max}$ and $P|p-batch, family-jobs, r_j, b<n|C_{\max}$, respectively. However, whether there exists a PTAS for $1|p-batch, family-jobs, r_j, b<n|C_{\max}$ when the number of families is arbitrary is still unsolved.

The machine scheduling problem with rejection was first considered by Bartal et al. [1]. They introduced a multiprocessor scheduling model with rejection to

minimize the makespan of the accepted jobs plus the total penalty of the rejected jobs. Hoogeveen et al. [11] considered the off-line version in Bartal et al. [1] when job preemption is allowed. Engels et al. [9] considered single-machine scheduling problem with rejection to minimize the total weighted completion time of the accepted jobs plus the total penalty of the rejected jobs. Dosa and He [8] considered the scheduling problem with machine cost and rejection. Lu et al. [16, 17], Cao and Yang [4], Miao et al. [20] considered scheduling problems with rejection on unbounded or bounded parallel-batching machines. Cheng and Sun [5], Li and Yuan [15] considered the deteriorating job scheduling problem with rejection on single machine and identical parallel machines, respectively. Shabtay [22] considered the scheduling problem with rejection on a single serial-batching machine. The reader is referred to Shabtay et al. [23] for more relevant and detailed discussion of this topic.

The remainder of this paper is organized as follows. In Sect. 2, we introduce some notation and basic lemmas. In Sect. 3, we first present a polynomial-time algorithm when all jobs have identical release dates, then we design a pseudo-polynomial-time algorithm for the case where the number of distinct release dates is fixed. In Sect. 4, we provide a 2-approximation algorithm and a polynomial-time approximation scheme. In Sect. 5, we conclude the paper.

2 Preliminaries

An algorithm $\mathcal{A}$ is a $(1+\rho)$-approximation algorithm for a minimization problem if it produces a solution that is at most of $(1+\rho)$ times the optimal solution. A family of algorithms $\{\mathcal{A}_\varepsilon : \varepsilon > 0\}$ is called a *polynomial-time approximation scheme* (PTAS) if, for each $\varepsilon > 0$, the algorithm $\mathcal{A}_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm running in polynomial time in the input size. We use the following notation in the rest of this paper.

- $r(B) = \max\{r_j : J_j \in B\}$ the release date of batch B
- $p_{\max} = \max\{p_j : J_j \in \mathcal{J}\}$ the largest processing time of the jobs in $\mathcal{J}$
- $r_{\max} = \max\{r_j : J_j \in \mathcal{J}\}$ the largest release date of the jobs in $\mathcal{J}$
- $E(\mathcal{S}) = \sum_{j\in\mathcal{S}} e_j$ the total rejection penalty of the jobs in $\mathcal{S}$
- n_f the number of jobs in family $\mathcal{F}_f$ (where $\sum_{f=1}^{m} n_f = n$), $f = 1, 2, \cdots, m$
- J_{fj} the j-th job in family $\mathcal{F}_f$, $f = 1, 2, \cdots, m$, $j = 1, 2, \cdots, n_f$
- p_{fj} the processing time of job J_{fj}, $f = 1, 2, \cdots, m$, $j = 1, 2, \cdots, n_f$
- r_{fj} the release date of job J_{fj}, $f = 1, 2, \cdots, m$, $j = 1, 2, \cdots, n_f$
- e_{fj} the rejection penalty of job J_{fj}, $f = 1, 2, \cdots, m$, $j = 1, 2, \cdots, n_f$

A schedule for the accepted jobs in $\mathcal{A}$ is a sequence $(B_1, t_1), (B_2, t_2), \cdots, (B_k, t_k)$ such that (1) $\{B_1, B_2, \cdots, B_k\}$ is a partition of $\mathcal{A}$; (2) all the jobs in B_i belong to the same family and $|B_i| \leqslant b$ for $1 \leqslant i \leqslant k$; and (3) $t_i \geqslant r(B_i)$ and $t_i + p(B_i) \leqslant t_{i+1}$ for $1 \leqslant i \leqslant k$, where t_i is starting time of batch B_i. The makespan of the schedule is $t_k + p(B_k)$.

The following lemma is very useful for our subsequent analysis of the dynamic programming (DP) formulation.

Lemma 2.1 *For problem* $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$, *if* $\{B_1, B_2, \cdots, B_k\}$ *is the set of processing batches of* $\mathcal{A}$ *in a schedule* π (*assuming* $r(B_1) \leqslant r(B_2) \leqslant \cdots \leqslant r(B_k)$), *then the minimum makespan can be expressed as*

$$C^{\pi}_{\max}(\mathcal{A}) = \max_{1\leqslant i\leqslant k}\{r(B_i) + \sum_{j=i}^{k} p(B_j)\}. \tag{2.1}$$

Proof Note that if the processing batches of $\mathcal{A}$ are determined in advance, the problem reduces to the classical problem $1|r_j|C_{\max}$ ([2]). Here each processing batch B_i $(1 \leqslant i \leqslant k)$ can be regarded as an "artificial" job J'_i with processing time $p'_i = p(B_i)$ and release date $r'_i = r(B_i)$. Recall that problem $1|r_j|C_{\max}$ can be solved by the *Earliest Release Date* (ERD) rule [2] in $O(n\log n)$ time. Then expression (2.1) can be easily obtained by induction on the number of processing batches. □

Next, we describe the FBLPT-FAMILY (Full Batch Large Processing Time) rule of Nong et al. [21], which computes the optimal solution for problem $1|p-batch, family-jobs, b<n|C_{\max}$.

FBLPT-FAMILY rule

For $f = 1, 2, \cdots, m$, do as follows:

Step 1 Sort the jobs in $\mathcal{F}_f$ in nonincreasing order of their processing times and obtain a job list.

Step 2 If there are more than b jobs in the job list, then place the first b jobs in a batch and iterate. Otherwise, place the remaining jobs in a batch.

Step 3 Sequence the batches in an arbitrary order without any idle time on the machine.

Given a job subset $\mathcal{S} \subseteq \mathcal{J}$, let $T(\mathcal{S})$ be the total processing time of the batches that are obtained by applying the FBLPT-FAMILY rule to set $\mathcal{S}$. By the optimality of the FBLPT-FAMILY rule for $1|p-batch, family-jobs, b<n|C_{\max}$, the following lemma holds immediately.

Lemma 2.2 *For problem* $1|rej, p-batch, family-jobs, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$, *there exists an optimal schedule in which the accepted jobs are assigned to the machine according to the* FBLPT-FAMILY *rule*.

3 Exact Algorithms

In this section, we first present a polynomial-time algorithm for the case where all jobs have the identical release dates and then provide a pseudo-polynomial-time algorithm for the case with h distinct release dates.

3.1 The Case with Identical Release Dates

In this subsection, we restrict our attention to the case where all jobs have identical release dates. Without loss of generality, we assume that all jobs are released at time zero. Due to the fact that the job families are incompatible and the objective function is additive, hence in this case our problem can be decomposed into m separable subproblems, where each subproblem corresponds to a single family problem. Note that when there is only one family, Lu et al. [16] have presented an optimal algorithm for the problem $1|rej, p-batch, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$, whose time complexity is $O(n^2)$. By utilizing the result of [16], we propose the following algorithm for $1|rej, p-batch, family-jobs, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$.

Algorithm A_1

Step 1 For $1\leqslant f\leqslant m$, run an optimal algorithm for the job set $\mathcal{J}$ restricted to family $\mathcal{F}_f$, and let $\mathcal{A}_f$ and $\mathcal{R}_f$ be the accepted jobs and rejected jobs of $\mathcal{F}_f$.

Step 2 Reject all jobs in $\mathcal{R}_1\cup\mathcal{R}_2\cup\cdots\cup\mathcal{R}_m$, and accept all jobs in $\mathcal{A}_1\cup\mathcal{A}_2\cup\cdots\cup\mathcal{A}_m$ and schedule them according to the FBLPT-FAMILY rule.

Recall that $\sum_{f=1}^{m} n_f=n$, thus we have $\sum_{f=1}^{m} O(n_f)^2\leqslant O(n^2)$. Hence, we have the following result.

Theorem 3.1 Algorithm A_1 solves problem $1|rej, p-batch, family-jobs, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$ in $O(n^2)$ time.

3.2 The case with h distinct release dates

In this subsection, we consider the problem with h ($h\geqslant 2$) distinct release dates. This problem $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$ is clearly NP-hard, since the fundamental problem $1|p-batch, r_j, b<n|C_{\max}$ with two distinct release dates is already NP-hard [18]. By exploiting some useful properties, we design a pseudo-polynomial-time dynamic programming (DP) algorithm for it when the number of distinct release dates h is fixed.

Let $R_1, R_2, \cdots, R_h$ be the h distinct job release dates satisfying $R_1<R_2<\cdots<R_h$. For convenience, we assume that $R_1=0$ and $R_{h+1}=+\infty$. Note that the accepted job set $\mathcal{A}$ can be partitioned according to a schedule π into a sequence of h disjoint subsets, $\mathcal{A}_1(\pi), \mathcal{A}_2(\pi), \cdots, \mathcal{A}_h(\pi)$, such that $\mathcal{A}_i(\pi)$ contains exactly those jobs started at or after time R_i but strictly before time R_{i+1}. The following lemma implies that we can locally rearrange the schedule of each $\mathcal{A}_i(\pi)$ without increasing the makespan of the accepted jobs so that the schedule follows the FBLPT-FAMILY rule for each $\mathcal{A}_i(\pi)$.

Lemma 3.2 *For any schedule π with makespan $C^{\pi}_{\max}(\mathcal{A})$, there exists a schedule π' with makespan $C^{\pi'}_{\max}(\mathcal{A})\leqslant C^{\pi}_{\max}(\mathcal{A})$ such that $\mathcal{A}_i(\pi')=\mathcal{A}_i(\pi)$ and the schedule for $\mathcal{A}_i(\pi')$ follows the* FBLPT-FAMILY *rule for any $1\leqslant i\leqslant h$.*

Proof If for each subset $\mathcal{A}_i(\pi)$, its schedule according to π follows the FBLPT-FAMILY rule, then we can simply set $\pi'=\pi$. Otherwise, we can schedule the jobs

in $\mathcal{A}_i(\pi)$ as follows: Since all the jobs in $\mathcal{A}_i(\pi)$ are available at or before time R_i, we can treat the scheduling of these jobs as the $1|p-batch, family-jobs, b<n|C_{\max}$ problem. We can obtain a new schedule π' by applying the FBLPT-FAMILY rule to the job set $\mathcal{A}_i(\pi)$. By Lemma 2.2, the FBLPT-FAMILY rule is optimal for the problem on the job set $\mathcal{A}_i(\pi)$. Hence the makespan of the new schedule will not be larger than the original one. This argument holds for arbitrary $\mathcal{A}_i(\pi)$. □

By Lemma 3.2, once the accepted job set $\mathcal{A}$ is determined, the original scheduling problem boils down to that of partition $\mathcal{A}$ into a sequence of h disjoint subsets $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_h$. Once this partition is fixed, we can assume that each subset $\mathcal{A}_i$ is scheduled (started but not necessarily completed) in the interval $[R_i, R_{i+1})$ according to the FBLPT-FAMILY rule. Moreover, each batch's starting time can be determined according to Lemma 2.1. Therefore, a simple enumeration approach to find the best partition is to examine all the different possibilities of h^n. However, we can do better.

Recall that $T(\mathcal{S})$ denote the total processing time of the batches that are obtained by applying the FBLPT-FAMILY rule to the job set $\mathcal{S}$. In view of Lemmas 2.1 and 3.2, the following result holds.

Lemma 3.3 *For problem* $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$, *if* $\{\mathcal{A}_1(\pi), \mathcal{A}_2(\pi), \cdots, \mathcal{A}_h(\pi)\}$ *is the partition of* $\mathcal{A}$ *in a schedule* π, *then its minimum makespan can be expressed as*

$$C^{\pi}_{\max}(\mathcal{A}) = \max_{1\leqslant i\leqslant q}\{R_i + \sum_{j=i}^{q} T(\mathcal{A}_i(\pi))\}, \tag{3.1}$$

where $q = \arg\max\{j : \mathcal{A}_j(\pi) \neq \emptyset\}$.

Next, we present an exact DP formulation for $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A}) + \sum_{J_j\in\mathcal{R}} e_j$. The idea behind the DP algorithm mainly relies on the structural properties described in Lemmas 3.2 and 3.3. To simplify the notation, we assume that all the jobs $J_{f1}, J_{f2}, \cdots, J_{f,n_f}$ of family $\mathcal{F}_f$ $(1 \leqslant f \leqslant m)$ are indexed such that $p_{f1} \geqslant p_{f2} \geqslant \cdots \geqslant p_{f,n_f}$.

We describe the DP approach in an enumerative form. In our enumeration, we build a schedule starting with job J_{11} of family $\mathcal{F}_1$ (stage 1) and ending with job J_{m,n_m} of family $\mathcal{F}_m$ (stage n). At stage k, $1 \leqslant k \leqslant n$, we enumerate all the possibilities of the positions for job J_{fj} of family $\mathcal{F}_f$ (with $\sum_{l=1}^{f-1} n_l < k = \sum_{l=1}^{f-1} n_l + j \leqslant \sum_{l=1}^{f} n_l$), i.e., job J_{fj} is either rejected or accepted and assigned to the last batch among the batches that start in the interval $[R_i, R_{i+1})$ for each $R_i \geqslant r_{fj}$. Specifically, a set $\mathcal{V}_k$ composed of states is generated at stage k. Each state $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ in $\mathcal{V}_k$ is associated with a feasible schedule for the first k jobs. Variable l_i denotes the total processing time of the batches that start in the interval $[R_i, R_{i+1})$ (not necessarily finished), variable c_i $(1 \leqslant c_i \leqslant b)$ denotes the number of jobs in the last batch that start in the interval $[R_i, R_{i+1})$, and variable e denotes the total penalty of the rejected jobs. At stage n, we can compute the minimum makespan of the accepted jobs plus the total penalty of the rejected jobs for each

complete schedule (which corresponds to a specific state $\langle n; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ in $\mathcal{V}_n$) as $\max_{1\leqslant i\leqslant q}\{R_i + \sum_{j=i}^{q} l_j\} + e$, where $q = \arg\max\{i : l_i > 0\}$. Then, we can compute the minimum solution value over all these complete schedules (by Lemma 3.3). Note that each state $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ in $\mathcal{V}_k$ carries all the information on the first k jobs needed to facilitate the forward movement of the DP.

We next state the two observations that will help us identify and retain a set of partial assignments at each stage such that one of them will provably lead to an optimal full assignment for the n jobs at the end of stage n. The first observation can be easily proved through a straightforward identical completion argument and the second observation holds since jobs from different families cannot be placed in the same batch.

Observation 3.4 At stage k of the DP, if $l_i \leqslant l'_i$ for $1 \leqslant i \leqslant h$ and $e \leqslant e'$, then state $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ dominates state $\langle k; l'_1, l'_2, \cdots, l'_h; c_1, c_2, \cdots, c_h; e'\rangle$, i.e., it suffices to retain only the state $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ for further enumeration.

Observation 3.5 Assume that k belongs to one of the set $\Omega = \{0, n_1, n_1 + n_2, \cdots, \sum_{f=1}^{m} n_f\}$. Given $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ and $\langle k; l_1, l_2, \cdots, l_h; c'_1, c'_2, \cdots, c'_h; e\rangle$ at stage k of the DP, it suffices to retain only either $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ or $\langle k; l_1, l_2, \cdots, l_h; c'_1, c'_2, \cdots, c'_h; e\rangle$ for further enumeration. Furthermore, for those states $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$ with the same values l_i for $1 \leqslant i \leqslant h$ and e, we can simply replace them by $\langle k; l_1, l_2, \cdots, l_h; b, b, \cdots, b; e\rangle$ for further enumeration.

From the definition of $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e\rangle$, its domain is

$$0 \leqslant l_i \leqslant T(\mathcal{J}),\ 1 \leqslant i \leqslant h, \tag{3.2}$$

$$1 \leqslant c_i \leqslant b,\ 1 \leqslant i \leqslant h, \tag{3.3}$$

$$0 \leqslant e \leqslant \sum_{j=1}^{n} e_j. \tag{3.4}$$

Combining the above analysis, a formal description of the DP is given as follows.

Algorithm A_2

Step 0	For $f = 1, 2, \cdots, m$, sort the jobs in $\mathcal{F}_f$ such that $p_{f1} \geqslant p_{f2} \geqslant \cdots \geqslant p_{f,n_f}$.
Step 1	Set $\mathcal{V}_0 := \{\langle 0; 0, \cdots, 0; b, \cdots, b; 0\rangle\}$ and $k := 0$.
Step 2	For $f = 1, 2, \cdots, m$ and $j = 1, 2, \cdots, n_f$, do the following:
Step 2.1	Set $k := k + 1$ (in this case $k = \sum_{l=1}^{f-1} n_l + j \leqslant \sum_{l=1}^{f} n_l$).
Step 2.2	Reset $\mathcal{V}_{k-1} := \{\langle k-1; l_1, \cdots, l_h; b, \cdots, b; e\rangle : \langle k-1; l_1, \cdots, l_h; c_1, \cdots, c_h; e\rangle \in \mathcal{V}_{k-1}\}$ if $j = 1$ (i.e., $k - 1 \in \Omega$).
Step 2.3	For each vector $\langle k-1; l_1, \cdots, l_h; c_1, \cdots, c_h, e\rangle \in \mathcal{V}_{k-1}$, do the following:

Step 2.3.1 Add one new vector $\langle k; l_1, \cdots, l_h; c_1, \cdots, c_h; e + e_{fj} \rangle$ temporarily to $\mathcal{V}_k$;

Step 2.3.2 Compute $g := \text{argmin}\{i : R_i \geqslant r_{fj}\}$. For each $i = g, g+1, \cdots, h$, add one new vector $\langle k; l_1, \cdots, l_h; c_1, \cdots, c_{i-1}, c_i + 1, c_{i+1}, \cdots, c_h; e \rangle$ temporarily to $\mathcal{V}_k$ if $c_i < b$, and $\langle k; l_1, \cdots, l_{i-1}, l_i + p_{fj}, l_{i+1}, \cdots, l_h; c_1, \cdots, c_{i-1}, 1, c_{i+1}, \cdots, c_h; e \rangle$ if $c_i = b$.

Step 2.4 Extract a minimum set of nondominated vectors from among the survivors in the current $\mathcal{V}_k$ and let this set be the final $\mathcal{V}_k$.

Step 3 From all the vectors in $\mathcal{V}_n$, select a member with the minimum solution value $\text{opt} = \min\{\max_{1 \leqslant i \leqslant q} \{R_i + \sum_{j=i}^{q} l_i\} + e : \langle n; l_1, \cdots, l_h; c_1, \cdots, c_h; e \rangle \in \mathcal{V}_n \text{ and } q = \arg\max\{i : l_i > 0\}\}$.

The above algorithm is correct as it never discards a partial schedule that upon completion may lead to an optimal schedule (unless there is another equivalent or better partial schedule) and it computes the minimum solution value correctly. The optimal schedule (including the partition of $\mathcal{A}$ and $\mathcal{R}$ of $\mathcal{J}$, and the partition $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_h$ of $\mathcal{A}$) can be found by tracing the solution back from the optimal vector in $\mathcal{V}_n$.

Theorem 3.6 *Problem* $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A}) + \sum_{J_j \in \mathcal{R}} e_j$ *can be solved by algorithm* A_2 *in* $O(nh(bT(\mathcal{J}))^h)$ *time, where* h *is the number of distinct release dates.*

Proof We only need to determine the complexity of algorithm A_2. Step 0 requires the implementation of the sorting procedure and takes $\sum_{f=1}^{m} O(n_f \log n_f) \leqslant O(n \log n)$ time. By Observations 3.4, we only need to keep the state $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e \rangle$ with the smallest penalty e for all states $\langle k; l_1, l_2, \cdots, l_h; c_1, c_2, \cdots, c_h; e' \rangle$ when the first $2h+1$ components are identical, so we have at most $O((bT(\mathcal{J}))^h)$ distinct vectors in $\mathcal{V}_k$ (recalling that the domain of the vectors in $\mathcal{V}_k$ is defined by (3.2)–(3.4)). Furthermore, since we construct at most $h+1$ vectors out of every vector in $\mathcal{V}_{k-1}$, the construction of set $\mathcal{V}_k$ in Steps 2.1-2.3 takes $O(h(bT(\mathcal{J}))^h)$ time, which is also the time required in Step 2.4. Step 3 requires at most $O(n(bT(\mathcal{J}))^h)$ time. Since Step 2 is repeated $O(n)$ times, the overall time complexity of algorithm A_2 is $O(nh(bT(\mathcal{J}))^h)$. □

Remark 3.7 When $m = 1$ (i.e., all the jobs belong to one family), Lu et al. [16] presented a DP algorithm for problem $1|rej, p-batch, r_j, b<n|C_{\max}(\mathcal{A}) + \sum_{J_j \in \mathcal{R}} e_j$, which ran in $O(nhb^h p_{\max}^{h-1}(\sum_{j=1}^{n} p_j)^h \sum_{j=1}^{n} e_j)$ time. Hence our algorithm is more efficient than their algorithm.

Remark 3.8 By removing the variable e and Observation 3.4, algorithm A_2 can be adapted to solve problem $1|p-batch, family-jobs, r_j, b<n|C_{\max}$ in $O(nh(bT(\mathcal{J}))^{h-1})$ time, where h is the number of distinct release dates. This is first pseudo-polynomial-time algorithm when h is fixed for the bounded version.

4 Approximation Algorithms

In this section, we provide a 2-approximation algorithm and a polynomial-time approximation scheme for problem $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$.

4.1 A 2-approximation Algorithm

Algorithm A_3

Step 1 For each $r\in\{r_j : j=1,2,\cdots,n\}$, partition the job set $\mathcal{J}$ into two sets of jobs such that $\mathcal{J}_1(r)=\{J_j : r_j\leqslant r\}$ and $\mathcal{J}_2(r)=\{J_j : r_j>r\}$.

Step 2 Set $r_j=0$ for each $J_j\in\mathcal{J}_1(r)$, and obtain a new instance $\mathcal{I}(r)$ just containing the jobs in $\mathcal{J}_1(r)$. Let $\mathcal{A}(r)$ and $\mathcal{R}(r)$ be the accepted jobs and rejected jobs of $\mathcal{J}_1(r)$ obtained from algorithm A_1. Schedule the jobs in $\mathcal{A}(r)$ from time r according to the FBLPT-FAMILY rule on the machine and reject all the other jobs. The resulting schedule for the original instance is denoted by $\sigma(r)$.

Step 3 Let $F(r)$ be the value of the objective value for each $\sigma(r)$. Among all the schedules obtained above, select the one with the minimum $F(r)$ value.

Theorem 4.1 *Algorithm A_3 is 2-approximation for $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A})+\sum_{J_j\in\mathcal{R}} e_j$, which runs in $O(n^3)$ time.*

Proof The schedule obtained by algorithm A_3 is denote by σ. Let F and F^* be the objective values of the schedule σ and an optimal schedule σ^*, respectively.

Let $\mathcal{A}^*$ and $\mathcal{R}^*$ be the sets of accepted and rejected jobs in σ^*, respectively. Define $r^*=\max\{r_j : J_j\in\mathcal{A}^*\}$. By the definition of r^*, we have $C_{\max}(\mathcal{A}^*)\geqslant r^*$ and $\mathcal{J}_2(r^*)=\{J_j : r_j>r^*\}\subseteq\mathcal{R}^*$. Hence, we have

$$F^*=C_{\max}(\mathcal{A}^*)+E(\mathcal{R}^*)\geqslant r^*+E(\mathcal{J}_2(r^*). \tag{4.1}$$

Again, by the optimality of algorithm A_1 and the definition of $\mathcal{I}(r^*)$, we also have

$$F^*\geqslant T(\mathcal{A}(r^*))+E(\mathcal{R}(r^*)). \tag{4.2}$$

From inequalities (4.1) and (4.2), we have

$$F\leqslant F(r^*)=r^*+T(\mathcal{A}(r^*))+E(\mathcal{R}(r^*))+E(\mathcal{J}_2(r^*)\leqslant 2F^*. \tag{4.3}$$

Since we have n choices of r, and each one can be done in $O(n^2)$ time (Theorem 3.1), the overall time complexity of algorithm A_3 is $O(n^3)$. □

4.2 A Polynomial-Time Approximation Scheme

Let F and F^* be the objective values of the schedule σ obtained by algorithm A_3 and an optimal schedule σ^*, respectively. By Theorem 4.1, we have $F^*\leqslant F\leqslant 2F^*$. For any job J_j with $r_j>F$, it must be rejected in σ^*. Otherwise, we have

$F^* \geqslant r_j > F \geqslant F^*$, a contradiction. Using the similar argument, for any job J_j with $p_j > F$, it must also be rejected in σ^*. Hence we can modify the release date r_j as $r_j = \min\{r_j, F\}$, and the processing time p_j as $p_j = \min\{p_j, F\}$, and the optimal objective value will not change. Hence, we can assume that $\max\{r_j, p_j\} \leqslant F$ for $1 \leqslant j \leqslant n$.

Algorithm A_ε

Step 1 For any $\varepsilon > 0$, set $\beta_1 = \varepsilon F/4$ and $\beta_2 = \varepsilon F/4n$. Given an instance I, define a new instance I' by rounding r_j and p_j in I such that $r'_j = \lfloor r_j/\beta_1 \rfloor \beta_1$, and $p'_j = \lfloor p_j/\beta_2 \rfloor \beta_2$ for each $1 \leqslant j \leqslant n$.

Step 2 Apply algorithm A_2 to the instance I' to obtain an optimal solution $\sigma^*(I')$ for the instance I'.

Step 3 Increase the starting time of each job in $\sigma^*(I')$ by β_1 and replace p'_j by the original p_j in $\sigma^*(I')$, for each $1 \leqslant j \leqslant n$, to obtain a feasible solution σ' for the instance I.

Theorem 4.2 *Algorithm A_ε is a polynomial-time approximation scheme for problem $1|rej, p-batch, family-jobs, r_j, b<n|C_{\max}(\mathcal{A}) + \sum_{J_j \in \mathcal{R}} e_j$.*

Proof By the execution of algorithm A_ε, σ' is a feasible schedule for the original instance I. We first address the accuracy issue. Let F_ε be the objective value of the schedule σ' obtained from algorithm A_ε. Also, Let $F^*(I')$ be the optimal objective value of the schedule $\sigma^*(I')$. Clearly, we have $F^* \geqslant F^*(I')$. From the fact that $x - 1 \leqslant \lfloor x \rfloor \leqslant x$ holds for any real x, we have

$$F_\varepsilon \leqslant F^*(I') + \beta_1 + \sum_{j=1}^{n}(p_j - p'_j) \leqslant F^* + \varepsilon F^*/2 + n\beta_2 \leqslant (1+\varepsilon)F^*.$$

Next, we analyze the time complexity of the algorithm. Since $r'_j \leqslant r_j \leqslant F$ and $\beta_1 = \varepsilon F/4$, we have $r_j/\beta_1 \leqslant 4/\varepsilon$ and hence $\lfloor \frac{r_j}{\beta_1} \rfloor \in \{0, 1, \cdots, \lfloor \frac{4}{\varepsilon} \rfloor\}$ for $1 \leqslant j \leqslant n$. Since $p'_j \leqslant p_j \leqslant F$ and $\beta_2 = \varepsilon F/4n$, we have $p_j/\beta_2 \leqslant 4n/\varepsilon$ and hence $\lfloor \frac{p_j}{\beta_2} \rfloor \in \{0, 1, \cdots, \lfloor \frac{4n}{\varepsilon} \rfloor\}$ for $1 \leqslant j \leqslant n$. From the definition of $T(\mathcal{J})$, we have $T(\mathcal{J})/\beta_2 \leqslant \sum_{j=1}^{n} p_j/\beta_2 \leqslant 4n^2/\varepsilon$ and hence $\lfloor T(\mathcal{J})/\beta_2 \rfloor \in \{0, 1, \cdots, \lfloor \frac{4n^2}{\varepsilon} \rfloor\}$. Therefore, using the algorithm A_2 in Sect. 3.2, we can obtain an optimal schedule for the rounded instance I' in time

$$O(nh(\frac{bT(\mathcal{J})}{\beta_2})^h) \leqslant O(n(1+\frac{4}{\varepsilon})(b\frac{4n^2}{\varepsilon})^{(1+\frac{4}{\varepsilon})}) = O(g(\varepsilon)b^{(1+\frac{4}{\varepsilon})}n^{(3+\frac{8}{\varepsilon})}),$$

where $g(\varepsilon) = (1+\frac{4}{\varepsilon})(\frac{4}{\varepsilon})^{(1+\frac{4}{\varepsilon})}$. Hence algorithm A_ε is a polynomial-time approximation scheme. □

Remark 4.3 For problem $1|p-batch, family-jobs, r_j, b<n|C_{\max}$, Nong et al. [21] provided a PTAS when the number of families m was fixed, while leave the open question whether there was PTAS when the number of families was arbitrary. Algorithm A_ε can be easily modified to obtain a PTAS for this problem.

5 Conclusions

We have studied the parallel-batching scheduling problem with incompatible job families and rejection on a single machine. In this problem, the jobs are processed in batches and jobs from different families cannot be placed into the same batch. The objective is to minimize the makespan of the accepted jobs plus the total penalty of the rejected jobs. We first present a polynomial-time algorithm when all jobs have identical release dates, then we design a pseudo-polynomial-time algorithm for the case where the number of distinct release dates is fixed. Finally, we provide a 2-approximation algorithm and a polynomial-time approximation scheme for the general problem. Future research may focus on extending this work to other type of scheduling criteria such as the total weighted completion time and maximum lateness.

Acknowledgments The authors would like to thank the associate editor and three anonymous reviewers for their helpful comments and suggestions on an earlier version of our paper.

References

[1] Bartal, Y., Leonardi, S., Spaccamela, A.M., Sgall, J., Stougie, L.: Multi-processor scheduling with rejection. SIAM J. Discret. Math. **13**, 64–78 (2000)
[2] Brucker, P.: Scheduling Algorithms, 5th edn. Springer, Berlin (2007)
[3] Brucker, P., Gladky, A., Hoogeveen, J.A., Kovalyov, M.Y., Potts, C.N., Tautenhahn, T., Van de Velde, S.L.: Scheduling a batch processing machine. J. Sched. **1**, 31–54 (1998)
[4] Cao, Z., Yang, X.: A PTAS for parallel batch scheduling with rejection and dynamic job arrivals. Theor. Comput. Sci. **410**, 2732–2745 (2009)
[5] Cheng, Y.S., Sun, S.J.: Scheduling linear deteriorating jobs with rejection on a single machine. Eur. J. Oper. Res. **194**, 18–27 (2009)
[6] Deng, X.T., Poon, C.K., Zhang, Y.Z.: Approximation algorithms in batch processing. J. Comb. Optim. **7**, 247–257 (2003)
[7] Dobson, G., Nambimadom, R.S.: The batch loading and scheduling problem. Oper. Res. **49**, 52–65 (2001)
[8] Dosa, G., He, Y.: Scheduling with machine cost and rejection. J. Comb. Optim. **12**, 337–350 (2006)
[9] Engels, D.W., Karger, D.R., Kolliopoulos, S.G., Sengupta, S., Uma, R.N., Wein, J.: Techniques for scheduling with rejection. J. Algorithms **49**, 175–191 (2003)
[10] Hochbaum, D.S., Landy, D.: Scheduling semiconductor burn-in operations to minimize total flowtime. Oper. Res. **45**, 874–885 (1997)
[11] Hoogeveen, H., Skutella, M., Woeginger, G.J.: Preemptive scheduling with rejection. Math. Program. **94**, 361–374 (2003)
[12] Lee, C.Y., Uzsoy, R.: Minimizing makespan on a single batch processing machine with dynamic job arrivals. Int. J. Prod. Res. **37**, 219–236 (1999)
[13] Lee, C.Y., Uzsoy, R., Martin-Vega, L.A.: Efficient algorithms for scheduling semi-conductor burn-in operations. Oper. Res. **40**, 764–775 (1992)
[14] Li, S.S., Yuan, J.J.: Parallel-machine parallel-batching scheduling with family jobs and release dates to minimize makespan. J. Comb. Optim. **19**, 84–93 (2010)
[15] Li, S.S., Yuan, J.J.: Parallel-machine scheduling with deteriorating jobs and rejection. Theor. Comput. Sci. **411**, 3642–3650 (2010)
[16] Lu, L.F., Cheng, T.C.E., Yuan, J.J., Zhang, L.Q.: Bounded single-machine parallel-batch scheduling with release dates and rejection. Comput. Oper. Res. **36**, 2748–2751 (2009)
[17] Lu, L.F., Zhang, L.Q., Yuan, J.J.: The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. Theor. Comput. Sci. **396**, 283–289 (2008)
[18] Liu, Z.H., Yu, W.: Scheduling one batch processor subject to job release dates. Discret. Appl. Math. **105**, 129–136 (2000)

[19] Mathirajan, M., Sivakumar, A.I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int. J. Adv. Manuf. Technol. **29**, 990–1001 (2006)
[20] Miao, C., Zhang, Y.Z., Wang, C.: Bounded parallel batch scheduling on unrelated parallel machines. Lect. Notes Comput. Sci. **6124**, 220–228 (2010)
[21] Nong, Q.Q., Ng, C.T., Cheng, T.C.E.: The bounded single-machine parallel-batching scheduling problem with family jobs and release dates to minimize makespan. Oper. Res. Lett. **111**, 435–440 (2008)
[22] Shabtay, D.: The single machine serial batch scheduling problem with rejection to minimize total completion time and total rejection cost. Eur. J. Oper. Res. **233**, 64–74 (2014)
[23] Shabtay, D., Gasper, N., Kaspi, M.: A survey on scheduling problems with rejection. J. Sched. **16**, 3–28 (2013)
[24] Yuan, J.J., Liu, Z.H., Ng, C.T., Cheng, T.C.E.: The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. Theor. Comput. Sci. **320**, 199–212 (2004)