The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan

J.J. YUAN[1,3], **Z.H. Liu**[2,3], **C.T. NG**[3] **and T.C.E. CHENG**[3*]

[1]Department of Mathematics, Zhengzhou University,

Zhengzhou, Henan 450052, People's Republic of China

[2]Department of Mathematics, East China University of Science and Technology,

Shanghai 200237, People's Republic of China

[3]Department of Logistics, The Hong Kong Polytechnic University,

Hung Hom, Kowloon, Hong Kong, People's Republic of China

ABSTRACT

In this paper we consider the unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. We show that this problem is strongly NP-hard, and give an $O\left(n\left(\frac{n}{m}+1\right)^m\right)$ time dynamic programming algorithm and an $O(mk^{k+1}P^{2k-1})$ time dynamic programming algorithm, where n is the number of jobs, m is the number of families, k is the number of distinct release dates and P is the sum of the processing times of all families. We further give a heuristic with a performance ratio 2. We also give a polynomial-time approximation scheme (PTAS) for the problem.

Keywords: Scheduling; Family; Batching; Makespan

1 Introduction and Problem Formulation

Let n jobs $J_1, J_2, ..., J_n$ and a single machine that can process the jobs concurrently in batches be given. Each job J_j has a processing time p_j and a release date r_j. The jobs are processed in batches, where a batch is a subset of the jobs and we require that the batches form a partition of the set of all jobs. The processing time of a batch is equal to

*Corresponding author

the maximum processing time among the jobs in the batch, i.e., the processing time of a batch B is defined as $p_B = \max\{p_j : J_j \in B\}$.

Suppose that a batch sequence (which indicates the processing order of a certain batch partition of the jobs) is given and we will process the batched jobs according to this batch sequence. We require that the starting time of a batch is at least the maximum release date of the jobs in it. So, we define the release date of a batch B as $r_B = \max\{r_j : J_j \in B\}$. When the objective function considered is regular, we suppose that all the jobs in the same batch start simultaneously at the earliest possible starting time. Consequently, the starting time of each batch is determined by the batch sequence. For a batch B, if the starting time of B is s_B, then the completion time of B and all the jobs in B is simply $s_B + p_B$. Following [3] and [9], we call this model the parallel batch scheduling problem and denote it by

$$1|p\text{-}batch;\ r_j|f,$$

where f is the objective function to be minimized, and "p-batch" means that the jobs contained in the same batch are processed in parallel, i.e., concurrently, so that the processing time of a batch is equal to the maximum processing time among the jobs in the batch.

The parallel batch scheduling problem is one of the important modern scheduling models that has received much attention in the literature. The fundamental model of the parallel batch scheduling problem was first introduced by Lee *et al.* [10] with the restriction that the number of jobs in each batch is bounded by a number b, which is denoted by $1|p\text{-}batch; b < n|f$. This bounded model is motivated by the *burn-in* operations in semiconductor manufacturing [10]. For example, a batch of integrated circuits (jobs) may be put inside an oven of limited size to test for their thermal standing ability. The circuits are heated inside the oven until all circuits are burned. The burn-in time of the circuits (job processing times) may be different. When a circuit is burned, it has to wait inside the oven until all circuits are burned. Therefore, the processing time of a batch of circuits is the processing time of the longest job in the batch.

An extensive discussion of the unbounded version of the parallel batch scheduling problem was provided in [1]. This unbounded model can be applied, for example, to situations where the batch contents are heated in a sufficiently large kiln and so the batch size is not restricted [1].

Recent developments of this topic can be found in [2] and from the web site [3]. In addition, [4], [6], [7], [11] and [12] presented new complexity results and approximation algorithms for the parallel batch scheduling problem subject to release dates.

In this paper we consider a generalization of the unbounded single machine parallel batch scheduling problem. Suppose that we have m families of jobs $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_m$, which partition the job set $\{J_1, J_2, ..., J_n\}$. Jobs from the same family are processed in batches while jobs from different families cannot be contained in the same batch. We call this problem the "unbounded single machine parallel batch scheduling with family jobs". The objective we consider in this paper is to minimize the makespan with release dates. In the sequel, this problem will be denoted by

$$1|\textit{family-jobs; p-batch; } r_j|C_{\max}.$$

We show in this paper that the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ is strongly NP-hard. We give an $O\left(n\left(\frac{n}{m}+1\right)^m\right)$ time dynamic programming algorithm and an $O(mk^{k+1}P^{2k-1})$ time dynamic programming algorithm for the problem, where n is the number of jobs, m is the number of families, k is the number of distinct release dates and P is the sum of the processing times of all families. We further give a heuristic with a performance ratio 2. We also give a polynomial-time approximation scheme (PTAS) for the problem.

2 A Useful Lemma

We first give an easy lemma, which will be used in the following sections.

Lemma 2.1 For the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$, there is an optimal batch sequence $BS = (B_1, B_2, ..., B_b)$ such that if two jobs J_i and J_j belong to the same family, where $J_i \in B_x$ and $J_j \in B_y$ with $x < y$, then $p_i > p_j$.

Proof Let $BS = (B_1, B_2, ..., B_b)$ be an optimal batch sequence for which the property of Lemma 2.1 does not hold. Then there are two jobs J_i and J_j that belong to the same family $\mathcal{F}_f$ such that $J_i \in B_x$ and $J_j \in B_y$ with $x < y$, but $p_i \le p_j$. We obtain a new batch sequence BS' by shifting the job J_i from B_x to B_y. It is easy to see that $C_{\max}(BS') \le C_{\max}(BS)$, and so BS' is optimal, too.

Continuing this procedure, we eventually obtain an optimal batch sequence with the required property.

$\square$

Corollary 2.2 There is an optimal batch sequence $BS = (B_1, B_2, ..., B_b)$ for the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ such that each batch B_x of family $\mathcal{F}_f$ is in the form $B_x = \{J_j \in \mathcal{F}_f : l \le p_j \le u\}$ for some numbers l and u.

3 NP-hardness Proof

We need the following strongly NP-complete 3-Partition problem.

3-Partition Problem: Given a set of $3t$ integers $a_1, a_2, ..., a_{3t}$, each of size between $B/4$ and $B/2$, such that $\sum_{i=1}^{3t} a_i = tB$, is there a partition of the a_i's into t groups of 3, each summing exactly to B?

By Garey and Johnson [8], we have

Lemma 3.1 The 3-Partition problem is strongly NP-complete.

Theorem 3.2 The problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ is strongly NP-hard.

Proof: The decision version of the problem is clearly in NP. To prove its NP-completeness, we use the strongly NP-complete 3-Partition problem for our reduction.

For a given instance of the 3-Partition problem with $a_1, a_2, ..., a_{3t}$, where $\frac{1}{t}\sum_{i=1}^{3t} a_i = B$, we construct an instance of the decision version of the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ as follows.

• Three key numbers are used in the construction:

$$\Delta = t^2 B + 1,$$

$$Z = t\Delta B + \frac{1}{2}(t^2 - t)B + 1,$$

$$X = t\Delta B + \frac{1}{2}(t^2 - t)(3Z + B) + 1;$$

• $3t^2$ jobs: $J_{(i,j)}$, $1 \le i \le 3t$, $1 \le j \le t$;

• $3t$ families $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_{3t}$, where

$$\mathcal{F}_i = \{J_{(i,j)} : 1 \le j \le t\}, \ \ 1 \le i \le 3t;$$

• Processing times of the jobs are defined as

$$p_{(i,j)} = (t + 1 - j)(Z + a_i), \ \ 1 \le i \le 3t, \ 2 \le j \le t,$$

and

$$p_{(i,1)} = X + \Delta a_i, \ \ 1 \le i \le 3t;$$

• Release dates of the jobs are defined as

$$r_{(i,j)} = 3(j-1)X + (j-1)\Delta B, \ 1 \le i \le 3t, 1 \le j \le t;$$

• Threshold value of the makespan is defined as

$$Y = 3tX + t\Delta B + \frac{1}{2}(t^2 - t)(3Z + B).$$

The decision version of the problem $1|\textit{family-jobs; P-batch; } r_j|C_{\max}$ asks whether there is a batch sequence BS such that the makespan $C_{\max}(BS) \le Y$.

Clearly, the construction can be done in polynomial time. We show in the sequel that the instance of the 3-Partition problem has a solution if and only if there is a batch sequence BS for the constructed instance of the scheduling problem such that the makespan $C_{\max}(BS) \le Y$.

Set $r^{(j)} = 3(j-1)X + (j-1)\Delta B$, $1 \le j \le t$. Then $r_{(i,j)} = r^{(j)}$, i.e., $r_{(i,j)}$ is independent of i. We will call $J_{(i,j)}$ the j-th job of family $\mathcal{F}_i$.

If the 3-Partition problem has a solution, we can re-lable the indices of $a_1, a_2, ..., a_{3t}$ such that

$$a_{3i-2} + a_{3i-1} + a_{3i} = B, \text{ for } 1 \le i \le t.$$

We construct a batch sequence BS of the scheduling problem as follows.

Each family $\mathcal{F}_f$, $1 \le f \le 3(t-1)$ is divided into two batches $\mathcal{B}_f$ and $\mathcal{A}_f$ such that

$$\mathcal{B}_f = \{J_{(f,j)} : 1 \le j \le \lceil \frac{1}{3} f \rceil\}$$

and

$$\mathcal{A}_f = \{J_{(f,j)} : \lceil \frac{1}{3} f \rceil < j \le t\}.$$

Each family $\mathcal{F}_f$, $f = 3t-2, 3t-1, 3t$, acts as a batch. We write $\mathcal{B}_f = \mathcal{F}_f$ for $f = 3t-2, 3t-1, 3t$ in the following. The batches are processed according to the following order under BS:

$$\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, ..., \mathcal{B}_{3i-2}, \mathcal{B}_{3i-1}, \mathcal{B}_{3i}, ..., \mathcal{B}_{3t-2}, \mathcal{B}_{3t-1}, \mathcal{B}_{3t}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, ..., \mathcal{A}_{3t-5}, \mathcal{A}_{3t-4}, \mathcal{A}_{3t-3}.$$

It is not hard to verify that, under the above batch sequence BS, $C_{\max}(BS) = Y$. Hence, the scheduling problem has the required batch sequence.

Now suppose that the scheduling problem has a required batch sequence. We need to show that the 3-Partition problem has a solution. By Lemma 2.1, we have the following claim.

Claim 1 There is a required batch sequence $BS = (B_1, B_2, ...B_m)$ of the scheduling problem such that

(1) for every two jobs $J_{(f,i)}$ and $J_{(f,j)}$ of any family $\mathcal{F}_f$ with $i < j$, either $J_{(f,i)}$ and $J_{(f,j)}$ are included in the same batch, or $J_{(f,i)}$ is included in a batch with an index smaller than that of the batch that contains $J_{(f,j)}$, i.e., $C_{(f,i)}(BS) \le C_{(f,j)}(BS)$;

(2) the job indices in each batch are consecutive, i.e., if $\mathcal{B}$ is a batch of family $\mathcal{F}_f$ under BS, then for every two jobs $J_{(f,i)}, J_{(f,j)} \in \mathcal{B}$ with $i < j$, $\{J_{(f,k)} : i \le k \le j\} \subseteq \mathcal{B}$.

Let $BS = (B_1, B_2, ...B_m)$ be a batch sequence of the scheduling problem that satisfies the properties in Claim 1. We need more properties of BS.

For each i, $1 \le i \le 3t$, let $\mathcal{B}_i$ be the batch in BS such that $J_{(i,1)} \in \mathcal{B}_i$. Then, $\mathcal{B}_i \subseteq F_i$. Let $m_i = |\mathcal{B}_i|$. Then, by Claim 1, we have

$$\mathcal{B}_i = \{J_{(i,j)} : 1 \le j \le m_i\}.$$

For convenience, we re-enumerate the families of jobs such that

$$m_1 \le m_2 \le ... \le m_{3t}.$$

Claim 2 $m_{3i} \le i$ for $1 \le i \le t$.

Suppose to the contrary that $m_{3i} \geq i+1$ for some i, $1 \leq i \leq t$. Then the earliest starting time of the batches in $\{\mathcal{B}_x : 3i \leq x \leq 3t\}$ is at least $r^{(i+1)} = 3iX + i\Delta B > 3iX$. Hence, the makespan is estimated as

$$C_{\max}(BS) > r^{(i+1)} + (3t - 3i + 1)X \geq 3tX + X > Y.$$

This contradicts our assumption.

Claim 3 $m_i = \lceil \frac{1}{3} i \rceil$ for $1 \leq i \leq 3t$.

In fact, by Claim 2, each family $\mathcal{F}_i$ with $m_i \leq 3(t-1)$ is splitted into at least two batches such that the sum of the processing times of these batches is at least $(X + \Delta a_i) + (t - m_i)(Z + a_i) > X + (t - m_i)Z$. So, the makespan can be estimated as

$$C_{\max}(BS) > 3tX + 3t^2 Z - \sum_{1 \leq i \leq 3t} m_i Z.$$

Since $C_{\max}(BS) \leq Y = 3tX + t\Delta B + \frac{1}{2}(t^2 - t)(3Z + B)$, we have

$$3tX + 3t^2 Z - \sum_{1 \leq i \leq 3t} m_i Z < 3tX + t\Delta B + \frac{1}{2}(t^2 - t)(3Z + B),$$

i.e.,

$$\sum_{1 \leq i \leq 3t} m_i Z > \frac{3}{2}(t^2 + t)Z - t\Delta B - \frac{1}{2}(t^2 - t)B = \frac{3}{2}(t^2 + t)Z - Z + 1.$$

So,

$$\sum_{1 \leq i \leq 3t} m_i > \frac{3}{2}(t^2 + t) - 1 + \frac{1}{Z}.$$

By the integrality of m_i, we deduce that

$$\sum_{1 \leq i \leq 3t} m_i \geq \frac{3}{2}(t^2 + t).$$

Since

$$\frac{3}{2}(t^2 + t) = \sum_{1 \leq i \leq 3t} \lceil \frac{1}{3} i \rceil,$$

we then have

$$\sum_{1 \leq i \leq 3t} m_i \geq \sum_{1 \leq i \leq 3t} \lceil \frac{1}{3} i \rceil,$$

or equavilently,

$$\sum_{1 \leq i \leq 3t} (m_i - \lceil \frac{1}{3} i \rceil) \geq 0.$$

By Claim 2, $m_i \leq \lceil \frac{1}{3} i \rceil$, and so $m_i - \lceil \frac{1}{3} i \rceil \leq 0$ for each i. This implies that the only possiblity is that $m_i = \lceil \frac{1}{3} i \rceil$ for each i, as required.

Claim 4 Each family $\mathcal{F}_i$, $1 \leq i \leq 3(t-1)$, is divided into just two batches under π.

Otherwise, some family $\mathcal{F}_k$ ($1 \leq k \leq 3(t-2)$) is divided into at least three batches under π. By Claim 3, two of these batches have processing times $X + \Delta a_k > X$ and $(t-m_k)(Z+a_k) > (t-m_k)Z$, and each of the remaining batches has at least a processing time $Z + a_k \geq Z$. So, the sum of the processing times of the batch $\mathcal{F}_k$ is greater than $Z + X + (t-m_k)Z$. The sum of the processing times of the batches of any other family $\mathcal{F}_i$ is at least $(X + ta_i) + (t-m_i)(Z+a_i) > X + (t-m_i)Z$. Hence, the makespan of BS is estimated as

$$\begin{array}{rl} & C_{\max}(BS) \\ > & Z + \sum_{1\leq i\leq 3t}(X + (t-m_i)Z) \\ = & Z + 3tX + 3t^2Z - \sum_{1\leq i\leq 3t}\lceil \frac{1}{3}i\rceil Z \\ = & Z + 3tX + \frac{3}{2}(t^2-t)Z \\ > & 3tX + t\Delta B + \frac{1}{2}(t^2-t)(3Z+B) \\ = & Y, \end{array}$$

a contradiction.

Set $\mathcal{A}_i = \mathcal{F}_i \setminus \mathcal{B}_i$, for $1 \leq i \leq 3t$. By Claim 1(2), Claim 3 and Claim 4, each family $\mathcal{F}_i$, $1 \leq i \leq 3(t-1)$, is divided into two batches $\mathcal{B}_i$ and $\mathcal{A}_i$ under BS, and we have

$$\mathcal{B}_i = \{J_{(i,j)} : 1 \leq j \leq \lceil \frac{i}{3} \rceil\}, \text{ for } 1 \leq i \leq 3t,$$

and

$$\mathcal{A}_i = \{J_{(i,j)} : \lceil \frac{i}{3} \rceil + 1 \leq j \leq t\}, \text{ for } 1 \leq i \leq 3(t-1).$$

Now, for $1 \leq k \leq t$, write $\alpha_k = a_{3k-2} + a_{3k-1} + a_{3k}$. Then, $\sum_{k=1}^{t} \alpha_k = tB$. We notice the following facts:

(1) The common release date of $\mathcal{B}_{3k-2}$, $\mathcal{B}_{3k-1}$ and $\mathcal{B}_{3k}$ is $r^{(k)} = 3(k-1)X + (k-1)\Delta B$.

(2)The common release date of $\mathcal{A}_i$, $1 \leq i \leq 3(t-1)$, is $r^{(t)} > r^{(k)}$, $1 \leq k \leq t-1$.

(3) $|\mathcal{B}_{3k-2}| = |\mathcal{B}_{3k-1}| = |\mathcal{B}_{3k}| = k$ for $1 \leq k \leq t$; and $|\mathcal{A}_{3k-2}| = |\mathcal{A}_{3k-1}| = |\mathcal{A}_{3k}| = t-k$ for $1 \leq k \leq t-1$.

For each k with $1 \leq k \leq t$, we consider the batches

$$\mathcal{B}_{3k-2}, \mathcal{B}_{3k-1}, ..., \mathcal{B}_{3t}, \mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_{3(t-1)}.$$

Since the minimum release date of these batches is $r^{(k)}$, the makespan $C_{\max}(BS)$ is greater than or equal to the value obtained by summing up $r^{(k)}$ and the processing times of these batches. Now,

$$r^{(k)} = 3(k-1)X + (k-1)\Delta B,$$

and the sum of the processing times of the jobs in these batches is

$$3(t-k+1)X + \Delta \sum_{k\leq i\leq t} \alpha_i + \sum_{1\leq i\leq t} i(3Z + \alpha_i).$$

Hence,

$$C_{\max}(BS) \geq 3Xt + (k-1)\Delta B + \Delta \sum_{k \leq i \leq t} \alpha_i + \frac{3}{2}t(t-1)Z + \sum_{1 \leq i \leq t} (t-i)\alpha_i.$$

By the assumption of $C_{\max}(BS) \leq Y = 3tX + t\Delta B + \frac{3}{2}t(t-1)Z + \frac{1}{2}t(t-1)B$, we deduce that

$$(*): \qquad \Delta \sum_{k \leq i \leq t} \alpha_i \leq \Delta(t-k+1)B + \sum_{1 \leq i \leq t} (t-i)(B-\alpha_i).$$

This implies that

$$\Delta \sum_{k \leq i \leq t} \alpha_i \leq \Delta(t-k+1)B + \Delta - 1, \quad 1 \leq k \leq t,$$

or equivalently

$$\sum_{k \leq i \leq t} \alpha_i \leq (t-k+1)B + 1 - \frac{1}{\Delta}.$$

By the integrality of α_i, we deduce the following t inequalities (I_k), $1 \leq k \leq t$:

$$(I_k): \qquad \sum_{k \leq i \leq t} \alpha_i \leq (t-k+1)B.$$

By setting $k = 1$ in the inequality $(*)$, we obtain

$$\sum_{1 \leq i \leq t} (t-i)(B-\alpha_i) \geq 0, \ 1 \leq k \leq t.$$

This can be rewriten as

$$\sum_{1 \leq i \leq t} i\alpha_i \geq \sum_{1 \leq i \leq t} iB,$$

i.e.,

$$\sum_{1 \leq k \leq t} \sum_{k \leq i \leq t} \alpha_i \geq \sum_{1 \leq k \leq t} (t-k+1)B.$$

Combining this with the t inequalities I_k $(1 \leq k \leq t)$, we deduce that

$$\sum_{k \leq i \leq t} \alpha_i = (t-k+1)B, \text{ for } 1 \leq k \leq t.$$

Consequently,

$$\alpha_k = B, \text{ for } 1 \leq k \leq t.$$

Hence, the 3-Partition problem has a solution. The result follows. □

Recall the following NP-complete Equal-size 2-Partition problem [8].

Equal-size 2-Partition Given a set of $2t$ positive integers $a_1, a_2, ..., a_{2t}$ such that $\sum_{i=1}^{2t} a_i = 2B$, is there a partition of the a_i's into 2 groups of t, each summing exactly to B?

By using the NP-complete Equal-size 2-Partition problem for the reduction, we can further prove the following two results.

Theorem 3.3 The problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ is NP-hard even when the jobs have at most 2 distinct release dates.

Proof Similar to the proof of Theorem 3.2. □

4 Algorithms

Consider the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$. By Corollary 2.2, if there are two jobs J_i and J_j in the same family such that $r_i \le r_j$ and $p_i \le p_j$, we can delete the job J_i from the job system. This procedure requires only $O(n)$ time. Hence, in the sequel, we can suppose that any two jobs J_i and J_j in the same family have different release dates and different processing times, and furthermore $r_i < r_j$ implies $p_i > p_j$.

The algorithms presented in this section include: an $O\left(n\left(\frac{n}{m}+1\right)^m\right)$ time dynamic programming algorithm; an $O(mk^{k+1}P^{2k-1})$ time dynamic programming algorithm, where n is the number of jobs, m is the number of families, k is the number of distinct release dates and P is the sum of the processing times of all families; a heuristic with a performance ratio 2; and a polynomial time approximation scheme.

4.1 A genaral dynamic programming algorithm

Suppose that we have m families $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_m$, and each family is in the form

$$\mathcal{F}_i = \{J_{(i,1)}, J_{(i,2)}, ..., J_{(i,n_i)}\}, \ \ 1 \le i \le m.$$

Suppose further that the jobs are enumerated such that

$$r_{(i,1)} < r_{(i,2)} < ... < r_{(i,n_i)}, \ \ 1 \le i \le m,$$

and

$$p_{(i,1)} > p_{(i,2)} > ... > r_{(i,n_i)}, \ \ 1 \le i \le m.$$

For a nonnegative integer x, write

$$\mathcal{F}_i^{(x)} = \{J_{(i,j)} : 1 \le j \le x\}, \ \ 1 \le i \le m.$$

For m integers $x_1, x_2, ..., x_m$ with $0 \le x_i \le n_i$, $1 \le i \le m$, let $R(x_1, x_2, ..., x_m)$ be the minimum makespan of the problem $1|\textit{family-jobs; p-batch; } r_j|C_{\max}$ restricted to the m subfamilies of jobs

$$\mathcal{F}_1^{(x_1)}, \mathcal{F}_2^{(x_2)}, ..., \mathcal{F}_m^{(x_m)}.$$

Consider an optimal batch sequence BS for the problem $1|$*family-jobs; p-batch;* $r_j|C_{\max}$ restricted to the fimilies $\mathcal{F}_1^{(x_1)}, \mathcal{F}_2^{(x_2)}, ..., \mathcal{F}_m^{(x_m)}$ such that BS satisfies the property described in Lemma 2.1. If the last batch B_b in BS is a subset of $\mathcal{F}_i$ and the maximum release date of the jobs in B_b is $r_{B_b} \in \{r_{(i,j)} : 1 \le j \le x_i\}$, then we have

$$\begin{aligned} & R(x_1, x_2, ..., x_m) \\ = \; & \max\{r_{B_b}, R(x_1, ..., x_{i-1}, x_i - |B_b|, x_{i+1}, ..., x_m)\} + P_{B_b}, \end{aligned}$$

where $r_{B_b} = r_{(i,x_i)}$ is the release date of the last job in B_b and $P_{B_b} = p_{(i,x_i-|B_b|+1)}$ is the processing time of the first job in B_b. Hence, our dynamic programming recursion can be expressed as

$$\begin{aligned} & R(x_1, x_2, ..., x_m) = \\ & \min_{1 \le i \le m, 1 \le y_i \le x_i - 1} \max\{r_{(i,x_i)}, R(x_1, ..., x_{i-1}, y_i, x_{i+1}, ..., x_m)\} + p_{(i,y_i+1)}. \end{aligned}$$

The initial condition is given by

$$R(0, 0, ..., 0) = 0.$$

The dynamic programming function has at most

$$(n_1 + 1)(n_2 + 1)...(n_m + 1) \le \left(\frac{n}{m} + 1\right)^m$$

states. Each recursion runs only $O(n)$ time, since we have at most $O(n)$ choices for (i, y_i) with $1 \le i \le m$ and $1 \le y_i \le n_i$. Hence, the overall complexity of the above dynamic programming recursion is $O\left(n\left(\frac{n}{m} + 1\right)^m\right)$.

One interesting corollary of the above discussion is that, when $m = 1$, the problem becomes the unbouned parallel batch scheduling problem to minimize makespan, i.e., $1|$*p-batch*, $r_j|C_{\max}$ [2, 13], and can be solved in $O(n^2)$ time. The complexity of this problem was posed as open in [3], but in fact, an $O(n^2)$ time algorithm was implied in [11].

4.2 A pseudopolynomial dynamic programming formulation under fixed number of release dates

Let k distinct release dates be given: $R_1, R_2, \dots, R_k$, satisfying $R_1 < R_2 < \cdots < R_k$. Then, the interval $[R_1, +\infty)$ is divided into k segments $[R_1, R_2), [R_2, R_3), \dots, [R_k, R_{k+1})$, where $R_{k+1} = +\infty$.

Let $g(a_1, \dots, a_k)$ denote the minimum makespan of the schedules for the n jobs, subject to the constraint that the first batch starting in $[R_i, R_{i+1})$ (if it exists) starts at time a_i ($i = 1, 2, \dots, k$). Clearly, we may require $a_1 = R_1$ and $R_i \le a_i < \min\{R_{i+1}, R_i + P\}$ ($2 \le i \le k$), where $P = \sum_{f=1}^{m} \max_{j=1}^{n_f} p_{(f,j)}$. Then, $(a_1, \dots, a_k)$ has at most $O(P^{k-1})$ configurations.

Now assume that some $(a_1, \ldots, a_k)$ is given. To compute $g(a_1, \ldots, a_k)$, we further introduce $h(j; l_1, \ldots, l_k)$ $(0 \leq j \leq m)$ as the minimum makespan to schedule the jobs of families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_j$, subject to the constraints that, for $i = 1, 2, \ldots, k$,

(i) the first batch starting in $[R_i, R_{i+1})$ (if it exists) starts at a_i;

(ii) the total processing time of the batches starting in $[R_i, R_{i+1})$ is l_i.

Then, $g(a_1, \ldots, a_k) = \min_{(l_1,\ldots,l_k)} h(m; l_1, \ldots, l_k)$, where $(l_1, \ldots, l_k)$ satisfies

(i) $0 \leq l_i \leq P$ $(1 \leq i \leq k)$;

(ii) $a_i + l_i \leq a_{i+u}$ if $l_{i+1} = l_{i+2} = \cdots = l_{i+u-1} = 0$ and $l_{i+u} \neq 0$ $(1 \leq i \leq k-1,$ $1 \leq u \leq k-i)$.

$h(m; l_1, \ldots, l_k)$ can be computed recursively. Initially, we define $h(0; 0, \ldots, 0) = a_k$ and for other cases, $h(0; l_1, \ldots, l_k) = +\infty$. Then, for $j = 1, 2, \ldots, m$,

$$h(j; l_1, \ldots, l_k) = \min_{(\delta_1,\ldots,\delta_k)} \{h(j-1; l_1 - \delta_1, \ldots, l_k - \delta_k) + \delta_k\},$$

where δ_i $(i = 1, 2, \ldots, k)$ is the processing time of the batch of $\mathcal{F}_j$ starting in $[R_i, R_{i+1})$ and satisfies $0 \leq l_i - \delta_i \leq R_{i+1} - a_i$ if $\delta_i \neq 0$. Since there are at most $O(n_j^k) \leq O(k^k)$ ways to partition the jobs of $\mathcal{F}_j$ into k sets, i.e., $(\delta_1, \ldots, \delta_k)$ has at most $O(k^k)$ configurations, and each recursion requires $O(k^{k+1})$ time. The size of the domain of $h(j; l_1, \ldots, l_k)$ is $O(mP^k)$. Thus, $g(a_1, \ldots, a_k)$ is obtained in $O(mk^{k+1}P^k)$ time. The globally optimal schedule is obtained by considering all configurations of $(a_1, \ldots, a_k)$, which requires $O(mk^{k+1}P^{2k-1})$ time.

4.3 A heuristic

Consider the following heuristic for the problem $1|$*family-jobs; p-batch;* $r_j|C_{\max}$.

Algorithm 4.3.1 Family Batchng Rule: Each family acts as a batch.

First, we re-enumerate the families $\mathcal{F}_1, ..., \mathcal{F}_m$ such that

$$r_{\mathcal{F}_1} \leq r_{\mathcal{F}_2} \leq ... \leq r_{\mathcal{F}_m}.$$

Then, set $BS = (\mathcal{F}_1, ..., \mathcal{F}_m)$.

Theorem 4.3.2 Family Batching Rule is a 2-approximation algorithm for the problem $1|$*family-jobs; p-batch;* $r_j|C_{\max}$.

Proof Let $C_{\max}^{opt}$ be the minimum makespan for the problem $1|$*family-jobs; p-batch;* $r_j|C_{\max}$. Two obvious lower bounds for $C_{\max}^{opt}$ are

$$C_{\max}^{opt} \geq \max\{r_j : 1 \leq j \leq n\} = \max\{r_{\mathcal{F}_i} : 1 \leq i \leq m\}$$

and

$$C_{\max}^{opt} \geq \sum_{1\leq i\leq m} p_{\mathcal{F}_i}.$$

Furthermore, an obvious upper bound for $C_{\max}(BS)$ is

$$C_{\max}(BS) \leq \max\{r_j : 1 \leq j \leq n\} + \sum_{1\leq i\leq m} p_{\mathcal{F}_i},$$

where BS is the batch sequence obtained by the Family Batching Rule (Algorithm 4.3.1). It follows that

$$C_{\max}(BS) \leq 2C_{\max}^{opt}.$$

□

The result of Theorem 4.3.2 is the best possible. To see this, let $\varepsilon > 0$ be any small positive number. We will construct an instance I of the problem $1|$*family-jobs; p-batch;* $r_j|C_{\max}$ such that the batch sequence obtained by the Family Batching Rule on I is not a $(2-\varepsilon)$-approximation solution.

We have m families $\mathcal{F}_1, ..., \mathcal{F}_m$, where $m \geq \frac{2}{\varepsilon}$. Each family $\mathcal{F}_i$ has two jobs, i.e.,

$$\mathcal{F}_i = \{J_{(i,1)}, J_{(i,2)}\}.$$

The proceesing times of the jobs are defined as

$$p_{(i,1)} = m \text{ and } p_{(i,2)} = 1, \quad 1 \leq i \leq m.$$

The release dates of the jobs are defined as

$$r_{(i,1)} = 0 \text{ and } r_{(i,2)} = m^2, \quad 1 \leq i \leq m.$$

One can verify that one of the optimal batch sequences is

$$(\{J_{(1,1)}\}, \{J_{(1,1)}\}, ..., \{J_{(m,1)}\}, \{J_{(1,2)}\}, \{J_{(2,2)}\}, ..., \{J_{(m,2)}\},$$

and the minimum makespan is given by $C_{\max}^{opt} = m^2 + m$. But the batch sequence obtained by the Family Batching Rule is

$$BS = (\mathcal{F}_1, ..., \mathcal{F}_m),$$

and the makespan of BS is

$$C_{\max}(BS) = 2m^2 > (2-\varepsilon)(m^2+m) = (2-\varepsilon)C_{\max}^{opt}.$$

Hence, the result of Theorem 4.3.2 is the best possible.

4.4 A polynomial time approximation scheme

In this section we will derive a polynomial time approximation scheme for the scheduling problem. First, we give a lemma that allows us to focus on the special case with a constant number of distinct release dates.

Lemma 4.4.1 Given a PTAS for the special case with a constant number of distinct release dates, then there exists a PTAS for the general problem.

Proof Let $\epsilon > 0$ be given. Define $r_{\max} = \max_{1\le i\le n} r_i$ and $\delta = \epsilon r_{\max}/2$. Note that $\delta \le \epsilon C_{\max}^{opt}/2$, since $r_{\max} \le C_{\max}^{opt}$, where $C_{\max}^{opt}$ denotes the optimal objective value. Round each release date r_i down to the nearest multiple of δ, i.e.,

$$r_i^* = \delta\lfloor r_i/\delta\rfloor \quad (i = 1, 2, \ldots, n)\,.$$

Clearly, the number of distinct r_i^* is no more than $1+r_{\max}/\delta = 1+2/\epsilon$, which is a constant number for a given ϵ. Let $C_{\max}^*$ denote the optimal objective value for the problem with the scaled release dates r_i^*. Consider a $(1 + \epsilon/2)$-approximation solution to the problem with the scaled release dates. Add δ to each batch's start time in the solution. Then we get a feasible schedule with respect to the release dates r_i, the $C_{\max}$ of which is bounded by

$$(1 + \epsilon/2)C_{\max}^* + \delta \le (1 + \epsilon)C_{\max}^{opt}\,,$$

where $C_{\max}^* \le C_{\max}^{opt}$ is applied. □

In the following, we present an FPTAS for the special case with a constant number k of distinct release dates. This is done by applying the well-known rounding technique to the dynamic programming formulation in Section 4.2.

Given $\epsilon > 0$, we define $\nu = \epsilon P/(n + 1)$. Let

$$\begin{aligned} r_i^* &= \lceil r_i/\nu\rceil\,, \\ p_i^* &= \lfloor p_i/\nu\rfloor \quad (i = 1, 2, \ldots, n). \end{aligned}$$

Suppose that we have found an optimal schedule and its objective value $C_{\max}^*$ for the problem with the scaled parameters r_i^* and p_i^* by the dynamic program in Section 4.2. Increase each processing time p_i^* by $p_i/\nu - p_i^*$, which increases the objective value by at most n. Now consider $1/\nu$ to be a unit of time. Then we can get a schedule with respect to r_i and p_i, and its objective value is given by

$$C_{\max} \le \nu C_{\max}^* + n\nu\,.$$

Also, consider an optimal schedule with respect to r_i and p_i, the objective value of which is denoted by $C_{\max}^{opt}$. Delay the starting of each batch by ν units of time in the schedule, and consider ν to be a unit of time. We obtain a schedule with the objective value

$$C'_{\max} = 1 + C_{\max}^{opt}/\nu\,.$$

Obviously, $C^*_{\max} \le C'_{\max}$ holds. Thus,

$$C_{\max} \le C^{opt}_{\max} + (n+1)\nu = C^{opt}_{\max} + \epsilon P \le (1+\epsilon)C^{opt}_{\max}\,.$$

The time complexity of the approximation scheme is dominated by the step to solve the problem with the scaled parameters. Let $P^* = \sum_{f=1}^{m} \max_{j=1}^{n_f} p_{(f,j)}{}^*$. Clearly, it holds that

$$P^* \le \frac{P}{\nu} = \frac{n+1}{\epsilon}\,.$$

Thus, the running time of the approximation scheme is bounded by

$$O\left(mk^{k+1}P^{*2k-1}\right) \le O\left(mk^{k+1}\left(\frac{n+1}{\epsilon}\right)^{2k-1}\right),$$

which is polynomial for given k and $1/\epsilon$. In other words, the approximation scheme is an FPTAS.

Acknowledgements

This research was supported in part by the Research Grant Council of Hong Kong under grant numbers PolyU B.23.37.Q454 and G-YW59. The first two authors were also supported by the National Natural Science Foundation of China under grant numbers 10371112 and 10101007.

References

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

[2] P. Brucker, *Scheduling Algorithms*, Springer-Verlag, Berlin, 2001.

[3] P. Brucker and S. Knust, Complexity results for scheduling problems, *http://www.mathematik.uni-osnabrueck.de/research/OR/class/*, 2003.

[4] T.C.E. Cheng, Z.H. Liu and W.C. Yu, Scheduling jobs with release dates and deadlines on a batch processing machine, *IIE Transactions*, **33**(2001), 685-690.

[5] E.G. Coffman, M. Yannakakis, M.J. Magazine and C. Santos, Batch sizing and sequencing on a single machine, *Annals of Operations Research*, **26**(1990), 135-147.

[6] X. Deng and Y.Z. Zhang, Minimizing mean response time for batch processing systems, *Lecture Notes in Computer Science*, **1627**(1999), 231-240.

[7] X. Deng, C.K. Poon and Y.Z. Zhang, Approximation algorithms in batch processing, *Lecture Notes in Computer Science*, **1741**(2000), 153-162.

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[9] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems, *Annals of Discrete Mathematics,* **1**(1977), 343-362.

[10] C.-Y. Lee, R. Uzsoy and L.A. Martin-Vega, Efficient algorithms for scheduling semiconductor burn-in operations, *Operations Research*, **40**(1992), 764-775.

[11] Z.H. Liu and W.C. Yu, Scheduling one batch processor subject to job release dates, *Discrete Applied Mathematics*, **105**(2000), 129-136.

[12] Z.H. Liu, J.J. Yuan and T.C.E. Cheng, On scheduling an unbounded batch machine, *Operation Research Letters*, **31**(2003), 42-48.

[13] C.N. Potts and M.Y. Kovalyov, Scheduling with batching: a review, *European Journal of Operational Research*, **120**(2000), 228-249.