Single machine batch scheduling problem with family setup times 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 single machine batch scheduling problem with family setup times 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 $O(mk^kP^{2k-1})$ 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 setup times of all the families and the processing times of all the jobs. 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; Release dates; Makespan

1 Introduction and Problem Formulation

In the single machine, family jobs, batch scheduling problem (see [2, 6]), we have n jobs $J_1, J_2, ..., J_n$ that are partitioned into m families $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_m$. Each job J_j has a processing time p_j, and each family $\mathcal{F}_f$ is associated with a setup time s_f. The jobs in a family are processed in batches, and each batch of jobs from family $\mathcal{F}_f$ will incur a setup

*Corresponding author

time s_f. The jobs in the same batch are required to be processed consecutively. For a batch B, the processing time of B is defined as $p_B = \sum_{J_j \in B} p_j$.

There are two ways to define the completion time of a job. One is the item availability model and the other is the batch availability model. In the item availability model, the completion time of a job J_j in a batch B of family $\mathcal{F}_f$ under a schedule π is defined as

$$C_j(\pi) = S_B(\pi) + s_f + \sum_{J_i \in \phi_j(\pi)} p_i,$$

where $S_B(\pi)$ is the starting time of the batch B under π, and $\phi_j(\pi) = \{J_i \in B : i = j$ or J_i is processed before $J_j\}$. In the batch availability model, the completion time of a job J_j in a batch B of family $\mathcal{F}_f$ under a schedule π is defined as the time when the last job in this batch has finished processing, i.e.,

$$C_B(\pi) = S_B(\pi) + s_f + \sum_{J_i \in B} p_i.$$

So, in the batch availability model, all the jobs in the same batch are completed together. Recent developments of family scheduling with both item and batch availability can be found in [7].

In the literature, the item availability model has been widely studied and is denoted by $1|s_f|V$, where V is the objective function to be minimized. But the batch availability model has mainly been studied for the special case with only one family of jobs. We will denote the batch availability model by $1|s_f; batch|V$.

The maximum lateness family scheduling problem $1|s_f|L_{\max}$ was first studied by Bruno and Downey [2]. They gave a binary NP-hardness proof for the problem $1|s_f|L_{\max}$. By Bruno and Downey [2], $1|s_f|L_{\max}$ is NP-hard even for either two distinct due dates, two jobs per family, or three distinct due dates, three jobs per family, and equal setup times; however, it is pseudo-polynomially solvable for a fixed number of due dates. The best algorithm for the problem $1|s_f|L_{\max}$ is a dynamic programming algorithm given by Ghosh and Gupta [5] with a time bound $O(m^2M^m)$, where $M = \frac{1}{m}\sum_{1\le f\le m}|\mathcal{F}_f| + 1$. Bruno and Downey [2] first posed the question of whether the problem $1|s_f|L_{\max}$ is strongly NP-hard. Ghosh and Gupta [5] pointed out that the long-standing question as to whether $1|s_f|L_{\max}$ is strongly NP-hard had remained open. Recently, Cheng et al. [3] proved that the family scheduling problem $1|s_f|L_{\max}$ is strongly NP-hard.

Suppose that each job J_j has a release date r_j. We require that, in any schedule, the setup of the batch that contains job J_j cannot start before time r_j. It is natural to define the release date of a batch B as

$$r_B = \max\{r_j : J_j \in B\}.$$

With release dates, the makespan minimization problem will be denoted by

$$1|s_f, r_j|C_{\max}.$$

It is obvious that $1|s_f, r_j|C_{\max}$ is equivalent to $1|s_f; batch; r_j|C_{\max}$.

The makespan scheduling problem is closely related to the maximum lateness scheduling problem. In fact, it is easy to see that $1|s_f, r_j|C_{\max}$ is equivalent to $1|s_f; batch|L_{\max}$, since there is a schedule for $1|s_f, r_j|C_{\max}$ with makespan at most C^* if and only if there is a schedule for $1|s_f; batch|L_{\max}$ with maximum lateness at most 0, where

$$d_j = C^* - r_j, \text{ for every job } J_j.$$

However, this relation cannot be directly established between $1|s_f, r_j|C_{\max}$ and $1|s_f|L_{\max}$. Hence, the strong NP-hardness of $1|s_f|L_{\max}$ does not necessarily imply the strong NP-hardness of $1|s_f, r_j|C_{\max}$ or, equivalently, $1|s_f; batch|L_{\max}$.

We show in this paper that the problem $1|s_f, r_j|C_{\max}$ is strongly NP-hard even if the processing times of the jobs are unit and the setup times of the families are identical. (We would like to remark here that whether $1|s_f = s|L_{\max}$ is strongly NP-hard remains open. This explains the difference between $1|s_f, r_j|C_{\max}$ and $1|s_f|L_{\max}$.) We give an $O\left(n\left(\frac{n}{m}+1\right)^m\right)$ time dynamic programming algorithm and an $O(mk^kP^{2k-1})$ 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 setup times of all the families and the processing times of all the jobs. We further give a heuristic with a performance ratio 2 for the problem. 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 $1|s_f; r_j|C_{\max}$, there is an optimal batch sequence $BS = (B_1, B_2, ..., B_b)$ such that if there exist two jobs J_i and J_j belonging to the same family, where $J_i \in B_x$ and $J_j \in B_y$ with $x < y$, then $r_i < r_j$.

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 $r_i \geq r_j$, $J_i \in B_x$ and $J_j \in B_y$ with $x < y$. We obtain a new batch sequence BS' by shifting the job J_j from B_y to B_x, i.e., $BS' = (B'_1, B'_2, ..., B'_b)$, such that

$$B'_i = \begin{cases} B_k, & i \notin \{x, y\}, \\ B_x \cup \{J_j\}, & i = x, \\ B_y \setminus \{J_j\}, & i = y. \end{cases}$$

In the case $B_y = \{J_j\}$, $B'_y = \emptyset$ is assumed to be a dummy batch that still incurs the setup time s_f but has a release date 0. Since $r_{B_x} \geq r_i \geq r_j$, $S_{B'_x}(BS') = S_{B_x}(BS)$, and so the h-th batch B'_h has completion time $C_{B'_h}(BS') \leq C_{B'_h}(BS) + p_j$ for $x \leq h \leq y - 1$

under the new batch sequence BS'. By the fact that the starting time of B_y under BS is $S_{B_y} \geq r_i$, the y-th new batch has completion time $C_{B'_y}(BS') \leq \max\{C_{B'_{y-1}}(BS'), S_{B_y}\} + s_f + P_{B_y} - p_j \leq \max\{C_{B_{y-1}}(BS) + s_f + P_{B_y}, S_{B_y} + s_f + P_{B_y} - p_j\} \leq C_{B_y}(BS)$, where $P_{B_y} = \sum_{J_i \in B_y} p_i$. It follows that BS' is an optimal batch sequence, too.

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

□

Corollary 2.2 There is an optimal batch sequence $BS = (B_1, B_2, ..., B_b)$ for the problem $1|s_f; r_j|C_{\max}$ such that each batch B_x of family $\mathcal{F}_f$ is of the form $B_x = \{J_j \in \mathcal{F}_f : l \leq r_j \leq u\}$ 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 [4], we have

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

Theorem 3.2 The problem $1|s_f = s, r_j|C_{\max}$ is strongly NP-hard.

Proof: The decision version of the problem is clearly in NP. To prove the 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 $1|s_f, r_j|C_{\max}$ as follows.

- $3t(t+1)$ jobs: $J_{(i,j)}$, $1 \leq i \leq 3t$, $1 \leq j \leq t+1$;
- $3t$ families $\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_{3t}$, where
$$\mathcal{F}_i = \{J_{(i,j)} : 1 \leq j \leq t+1\}, \quad 1 \leq i \leq 3t;$$
- Processing times of the jobs are defined as
$$p_{(i,j)} = Z + a_i, \ 1 \leq i \leq 3t, 1 \leq j \leq t+1, \text{ where } Z = t(t+2)B;$$
- Setup times of the families are defined as
$$s_i = X, \ 1 \leq i \leq 3t, \text{ where } X = (3t^2 + 3t + 1)Z;$$
- Release dates of the jobs are defined as
$$r_{(i,j)} = 3(j-1)X + \frac{3}{2}j(j-1)Z + \frac{1}{2}j(j-1)B, \ 1 \leq i \leq 3t, 1 \leq j \leq t+1;$$

• Threshold value of the makespan is defined as

$$Y = 6Xt + 3t(t+1)Z + t(t+1)B = 2\sum_{1\le i\le 3t} s_i + \sum_{1\le i\le 3t, 1\le j\le t+1} p_{(i,j)}.$$

The decision version of the problem $1|s_f, r_j|C_{\max}$ 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 $r^{(j)} = 3(j-1)X + \frac{3}{2}j(j-1)Z + \frac{1}{2}j(j-1)B$, $1 \le j \le t+1$. 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 our scheduling problem as follows.

Each family $\mathcal{F}_f$, $1 \le f \le 3t$ 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

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

The batches are processed according to the following order under BS:

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

The jobs in each batch are sequenced in any order under BS.

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

Now suppose that our scheduling problem has the 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 $BS = (B_1, B_2, ..., B_b)$ for 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 containing $J_{(f,j)}$, 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 $BS = (B_1, B_2, ..., B_b)$ be the required batch sequence of the scheduling problem that satisfies the properties in Claim 1. We need more properties of BS.

Let $m_j = |\{B_x : r_{B_x} \geq r^{(j)}, 1 \leq x \leq b\}|$, $1 \leq j \leq t+1$.

Claim 2 $m_j \leq 3(2t - j) + 3$, $1 \leq j \leq t+1$.

Suppose to the contrary that $m_j \geq 3(2t-j)+4$ for some j with $1 \leq j \leq t+1$. Since the earliest starting time of the m_j batches in $\{B_x : r_{B_x} \geq r^{(j)}, 1 \leq x \leq b\}$ is at least $r^{(j)} \geq 3(j-1)X$ and each batch has a setup time X, the makespan is estimated as

$$C_{\max}(BS) > r^{(j)} + m_j X \geq 3(j-1)X + 3(2t-j)X + 4X = 6tX + X > Y.$$

This contradicts our assumption.

By Claim 2, we have

Claim 3 $b \leq 6t$.

Let $N = \sum_{x:\, r_{B_x} \geq r^{(t+1)}} |B_x|$.

Claim 4 $N \leq \frac{3}{2}t(t+1)$.

Otherwise, the $3t$ batches with release date $r^{(t+1)}$ must contain at least $\frac{3}{2}t(t+1)+1$ jobs. Then

$$C_{\max} > r^{(t+1)} + 3tX + \frac{3}{2}t(t+1)Z + Z > 6tX + 3t(t+1)Z + Z > Y,$$

a contradiction.

Suppose that $\mathcal{A}_f$ is the batch of family $\mathcal{F}_f$ under BS such that the job $J_{(f,t+1)} \in \mathcal{A}_f$, $1 \leq f \leq 3t$. Furthermore, we re-lable the indices of $\mathcal{F}_1, ..., \mathcal{F}_{3t}$ such that

$$|\mathcal{A}_1| \geq |\mathcal{A}_2| \geq ... \geq |\mathcal{A}_{3t}|.$$

Let

$$\mathcal{B}_f = \mathcal{F}_f \setminus \mathcal{A}_f$$

and $b_f = |\mathcal{B}_f| \leq t$, for $1 \leq f \leq 3t$. Then, by Claim 1(2),

$$\mathcal{B}_f = \{J_{(f,1)}, J_{(f,2)}, ..., J_{(f,b_f)}\}, \text{ for } 1 \leq f \leq 3t.$$

Claim 5 $b_{3k} \leq k$, for $1 \leq k \leq t$.

Otherwise, let h be the maximum index such that $b_{3h} \geq h+1$. Since $b_{3h} \leq t$, by the maximality of h, $m_{h+1} \geq 3t + (3t - 3h + 1) = 3(2t-h)+1$, a contradiction to Claim 2.

Claim 6 $b_{3k-2} = b_{3k-1} = b_{3k} = k$, for $1 \leq k \leq t$.

From Claim 4, we have

$$\sum_{1 \leq k \leq 3t} b_k = 3t(t+1) - N \geq \frac{3}{2}t(t+1).$$

By Claim 5, we also have $b_{3k-2}, b_{3k-1}, b_{3k} \leq k$, and thus

$$\sum_{1\leq k\leq 3t} b_k \leq \frac{3}{2}t(t+1).$$

This just implies that $b_{3k-2} = b_{3k-1} = b_{3k} = k$ for $1 \leq k \leq t$. The proof of Claim 6 is completed.

It is implied in Claim 6 that each family $\mathcal{F}_i$, $1 \leq i \leq 3t$, is divided into at least two batches under BS. If some family is divided into at least three batches, then there are at least $6t+1$ batches (under BS), contradicting Claim 3. Hence, we conclude that each family $\mathcal{F}_i$ is divided exactly into two batches $\mathcal{B}_i$ and $\mathcal{A}_i$ under BS.

Furthermore, from Claim 1(2) and Claim 6, 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

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

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

$$r^{(k)} = 3(k-1)X + \frac{3}{2}k(k-1)Z + \frac{1}{2}k(k-1)B.$$

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

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

For each k with $2 \leq k \leq t+1$, we consider the batches

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

where, when $k = t+1$, the considered batches are $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_{3t}$. 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 $r^{(k)}$, the setup times of these batches and the processing times of the jobs in these batches. Now,

$$r^{(k)} = 3(k-1)X + \frac{3}{2}k(k-1)Z + \frac{1}{2}k(k-1)B,$$

the sum of the setup times of these batches is $3(2t-k+1)X$, and the sum of the processing times of the jobs in these batches is

$$3t(t+1)Z + t(t+1)B - \frac{3}{2}k(k-1)Z - \sum_{1\leq i\leq k-1} i\alpha_i.$$

Hence,

$$C_{\max}(BS) \geq 6Xt + 3t(t+1)Z + t(t+1)B + \sum_{1\leq i\leq k-1} iB - \sum_{1\leq i\leq k-1} i\alpha_i.$$

By the assumption $C_{\max}(BS) \leq Y = 6Xt + 3t(t+1)Z + t(t+1)B$, we deduce that

$$\sum_{1\leq i\leq k-1} i\alpha_i \geq \sum_{1\leq i\leq k-1} iB,\ 2 \leq k \leq t+1,$$

or equivalently, we have the following t inequalities (I_k), $1 \leq k \leq t$:

$$(I_k): \qquad \sum_{1\leq i\leq k} i\alpha_i \geq \sum_{1\leq i\leq k} iB,\ 1 \leq k \leq t.$$

Set $\lambda_k = \frac{1}{k} - \frac{1}{k+1} = \frac{1}{k(k+1)}$ for $1 \leq k \leq t-1$, and set $\lambda_t = \frac{1}{t}$. Because each λ_k is positive, the linear combination of the above t inequalities (I_k), $1 \leq r \leq t$, yields the following inequality $(*)$.

$$(*): \quad \sum_{k=1}^{t} \lambda_k \sum_{i=1}^{k} i\alpha_i \geq \sum_{k=1}^{t} \lambda_k \sum_{i=1}^{k} iB.$$

One can easily verify that the left hand side of the inequality $(*)$ is $\sum_{k=1}^{t} \alpha_k$, and the right hand side of the inequality $(*)$ is tB. By the fact that $\sum_{k=1}^{t} \alpha_k = tB$, we deduce that equality always holds for the inequality $(*)$. Since the inequality $(*)$ is a positive linear combination of the t inequalities I_k, $1 \leq k \leq t$, we deduce that equality always holds for each of the t inequalities (I_r), $1 \leq r \leq t$, i.e.,

$$\sum_{1\leq i\leq k} i\alpha_i = \sum_{1\leq i\leq k} iB,\ 1 \leq k \leq t.$$

From these equalities, we can trivially deduce that

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

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

The problem $1|s_f = s, r_j|C_{\max}$ and $1|s_f = s, r_j, p_j = 1|C_{\max}$ have the same complexity status in a unary sense. In fact, each job with $p_j > 1$ can be split into p_j small jobs, each with a unit processing time and release date r_j. By Lemma 2.1, these small jobs split from J_j can be included in the same batch in an optimal batch sequence. Thus, the set of these small jobs will act as J_j. Hence, we have

Theorem 3.3 $1|s_f = s, r_j, p_j = 1|C_{\max}$ is strongly NP-hard.

Recall the following NP-complete Equal-size 2-Partition problem [4].

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 3.4 The problem $1|s_f = s, r_j|C_{\max}$ is NP-hard even when the jobs have at most 3 distinct release dates.

Proof Similar to the proof of Theorem 3.2. □

Theorem 3.5 The problem $1|s_f, r_j|C_{\max}$ is NP-hard even when the jobs have at most 2 distinct release dates.

Proof For a given instance of the Equal-size 2-Partition problem with $a_1, a_2, ..., a_{2t}$, where $\sum_{i=1}^{2t} a_i = 2B$, we construct an instance of the decision version of the problem $1|s_f, r_j|C_{\max}$ with two distinct release dates as follows.

- $4t$ jobs: $J_{(i,j)}$, $1 \leq i \leq 2t$, $1 \leq j \leq 2$;
- $2t$ families $\mathcal{F}_1, \mathcal{F}_2, ...\mathcal{F}_{2t}$, where

$$\mathcal{F}_i = \{J_{(i,j)} : 1 \leq j \leq 2\}, \ \ 1 \leq i \leq 2t;$$

- Processing times of the jobs are defined as

$$p_{(i,j)} = Z + a_i, \ 1 \leq i \leq 2t, 1 \leq j \leq 2, \text{ where } Z = 3tB;$$

- Setup times of the families are defined as

$$s_i = X + a_i, \ 1 \leq i \leq 2t, \text{ where } X = 4tZ;$$

- Release dates of the jobs are defined as

$$r_{(i,1)} = 0 \text{ and } r_{(i,2)} = tX + tZ + 2B, \ \ 1 \leq i \leq 2t;$$

- Threshold value of the makespan is defined as

$$Y = 3tX + 4tZ + 7B.$$

The decision version of the problem $1|s_f, r_j|C_{\max}$ asks whether there is a batch sequence BS such that the makespan $C_{\max}(BS) \leq Y$.

The construction is done in polynomial time. Let $BS = (B_1, B_2, ..., B_b)$ be a batch sequence for the scheduling instance such that $C_{\max}(BS)$ is minimum. We will lose nothing by re-sequencing the batches by the earliest release date first (ERD) rule. Hence, we can suppose that $r_{B_1} \leq r_{B_2} \leq ... \leq r_{B_b}$. Let k be the maximum index such that $r_{B_k} = 0$ (in case that $r_{B_1} > 0$, we set $k = 0$). Then

$$b = 2t + k \text{ and } |B_1| = |B_2| = ... = |B_k| = 1.$$

Suppose that the jobs in $B_1 \cup B_2 \cup ... \cup B_k$ are $J_{(i_1,1)}, J_{(i_2,1)}, ..., J_{(i_k,1)}$, and set $D = \sum_{1 \leq e \leq k} a_{i_e}$. Then, the makespan $C_{\max}(BS)$ can be calculated by

$$\max\{tX + tZ + 2B, kX + kZ + 2D\} + 2tX + (4t - k)Z + 6B - D.$$

From this we can deduce that $C_{\max}(BS) \leq Y$ if and only if $k = t$ and $D = B$. Consequently, the instance of the Equal-size 2-Partition problem has a solution if and only if there is a batch sequence BS for the scheduling instance such that the makespan $C_{\max}(BS) \leq Y$. Hence, $1|s_f, r_j|C_{\max}$ with at most two distinct release dates is NP-hard.

□

4 Algorithms

Consider the problem $1|s_f, r_j|C_{\max}$. By Lemma 2.1, we can combine the jobs with the same release date in the same family into a big job. This procedure requires only $O(n)$ time. Hence, we can suppose that any two jobs in the same family have different release dates.

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

4.1 A general dynamic programming algorithm

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

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

Suppose further that the jobs are numbered such that

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

For a nonnegative integer x, write

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

When $x = 0$, these sets are empty, and so no setup is needed.

For m integers $x_1, x_2, ..., x_m$ with $0 \leq x_i \leq n_i$, let $R(x_1, x_2, ..., x_m)$ be the minimum makespan of the problem $1|s_f, r_j|C_{\max}$ 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 $1|s_f, r_j|C_{\max}$ restricted to the families $\mathcal{F}_1^{(x_1)}, \mathcal{F}_2^{(x_2)}, ..., \mathcal{F}_m^{(x_m)}$ 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

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

where $r_{B_b} = r_{(i,x_i)}$ and P_{B_b} is the sum of the processing times of the jobs in B_b. Hence, our dynamic programming recursion can be given by

$$\begin{aligned} & R(x_1, x_2, ..., x_m) = \\ & \min_{1 \le i \le m, 0 \le y_i \le x_i - 1} \max\{r_{(i,x_i)}, R(x_1, ..., x_{i-1}, y_i, x_{i+1}, ..., x_m)\} + s_i + P^{(i)}(x_i) - P^{(i)}(y_i), \end{aligned}$$

where, for any integer y with $1 \le y \le n_i$,

$$P^{(i)}(y) = \sum_{1 \le j \le y} p_{(i,j)}.$$

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. The computation of $P^{(i)}(y)$, for all i and y with $1 \le i \le m$ and $1 \le y \le n_i$, can be taken before the dynamic programming recursion, which needs only $O(n)$ time, since

$$P^{(i)}(y + 1) = P^{(i)}(y) + p_{(i,y)}.$$

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 serial batching scheduling problem to minimize makespan, i.e., $1|s\text{-}batch, r_j|C_{\max}$, which can be solved in $O(n^2)$ time [1, 7].

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

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

Let $g(a_1, \ldots, a_k)$ denote the minimum makespan of the schedule 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 ($i = 1, 2, \ldots, k$). 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 $P = \sum_{f=1}^{m} s_f + \sum_{j=1}^{n} p_j$. Then, $(a_1, \ldots, a_k)$ 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 $h(j; l_1, \ldots, l_k)$ ($0 \le j \le m$) 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 setup and processing times 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) $0 \le l_i \le P$ ($1 \le i \le k$);

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

$h(m; l_1, \ldots, l_k)$ 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 setup and processing requirement of the batch of $\mathcal{F}_j$ starting in $[R_i, R_{i+1})$ and satisfies $0 \le l_i - \delta_i \le R_{i+1} - a_i$ if $\delta_i \ne 0$. Let $\Delta_i = l_i - \delta_i$, $1 \le i \le k$. Then, for $j = 1, 2, \ldots, m$,

$$h(j; l_1, \ldots, l_k) = \min_{(\Delta_1,\ldots,\Delta_k)} \{h(j-1; \Delta_1, \ldots, \Delta_k) + l_k - \Delta_k\},$$

and Δ_i ($i = 1, 2, \ldots, k$) satisfies $0 \le \Delta_i \le R_{i+1} - a_i$ if $\Delta_i \ne l_i$. Since there are at most $O(n_j^k) \le O(k^k)$ ways to partition the jobs of $\mathcal{F}_j$ into k sets, $(\delta_1, \ldots, \delta_k)$ has at most $O(k^k)$ configurations, and so $(\Delta_1, \ldots, \Delta_k)$ has at most $O(k^k)$ configurations. Therefore, each recursion requires $O(k^k)$ 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 $O(mk^kP^k)$ time. The globally optimal schedule is obtained by considering all the configurations of $(a_1, \ldots, a_k)$, which requires $O(mk^kP^{2k-1})$ time. Clearly, this algorithm is pseudopolynomial when k is fixed.

4.3 A heuristic

Consider the following heuristic for the problem $1|s_f, r_j|C_{\max}$.

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

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

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

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

Theorem 4.3.2 The Family Batching Rule is a 2-approximation algorithm for the problem $1|s_f, r_j|C_{\max}$.

Proof Let $C_{\max}^{opt}$ be the minimum makespan for the problem $1|s_f, r_j|C_{\max}$. Two obvious lower bounds for $C_{\max}^{opt}$ are

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

and

$$C_{\max}^{opt} \ge \sum_{1\le i\le m} s_i + \sum_{1\le j\le n} p_j.$$

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

$$C_{\max}(BS) \le \max\{r_j : 1 \le j \le n\} + \sum_{1\le i\le m} s_i + \sum_{1\le j\le n} p_j,$$

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

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

□

The performance ratio of 2 for the Family Batching Rule cannot be further refined. To see this, let $\varepsilon > 0$ be any small positive number. We will construct an instance I of the problem $1|s_f, r_j|C_{\max}$ 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 $m \ge \frac{3}{\varepsilon}$. Each family $\mathcal{F}_i$ has two jobs, i.e.,

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

The processing times of the jobs are defined as

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

The release dates of the jobs are defined as

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

The setup times of the families are defined as

$$s_i = 1, \quad 1 \le i \le m.$$

One can verify that one of the optimal batch sequences is

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

and the minimum makespan is given by $C_{\max}^{opt} = m^2+3m$. But the batch sequence obtained by the Family Batching Rule is

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

and the makespan of BS is

$$C_{\max}(BS) = 2m^2 + 3m > (2-\varepsilon)(m^2+3m) = (2-\varepsilon)C_{\max}^{opt}.$$

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, 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:

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

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 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 $\nu = \epsilon P/(mk+n+1)$. Let

$$r_i^* = \lceil r_i/\nu\rceil\,,\ p_i^* = \lfloor p_i/\nu\rfloor \quad (i = 1, 2, \ldots, n)$$
$$s_f^* = \lfloor s_f/\nu\rfloor \quad (f = 1, 2, \ldots, m).$$

Suppose that we have found an optimal schedule and its objective value $C^*_{\max}$ for the problem with the scaled parameters s^*_f, r^*_i and p^*_i by the dynamic program in Section 4.2. Note that the schedule has at most mk batches, and hence contains at most mk setups. Increase each setup s^*_f by $s_f/\nu - s^*_f$ and each processing time p^*_i by $p_i/\nu - p^*_i$, which increases the objective value by at most $mk + n$. Now consider $1/\nu$ to be a unit of time. Then we can get a schedule with respect to s_f, r_i and p_i, and its objective value is given by

$$C_{\max} \leq \nu C^*_{\max} + (mk + n)\nu \, .$$

Also, consider an optimal schedule with respect to s_f, r_i and p_i, the objective value of which is denoted by $C^{opt}_{\max}$. 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

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

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

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

The time complexity of the approximation scheme is dominated by the step to solve the problem with the scaled parameters. Let $P^* = \sum_{f=1}^{m} s^*_f + \sum_{i=1}^{n} p^*_i$. Clearly, it holds that

$$P^* \leq \sum_{f=1}^{m} \frac{s_f}{\nu} + \sum_{i=1}^{n} \frac{p_i}{\nu} = \frac{P}{\nu} = \frac{mk + n + 1}{\epsilon} \, .$$

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

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

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

4.5 A remark on the problem with only two distinct release dates

Suppose the job system has only two distinct release dates R_1 and R_2. When the setup times of the families are the same, i.e., $s_f = s$ for every family $\mathcal{F}_f$, we can easily show that the problem $1|s_f, r_j|C_{\max}$ can be polynomially solved by the following algorithm (the proof is routine and omitted). Together with Theorem 3.4 and 3.5, this can help to deduce the complexity status of the problem $1|s_f, r_j|C_{\max}$.

Algorithm 4.5.1 Batching rule under two distinct release dates.

(1) Renumber the families with two jobs in non-decreasing order of the processing times of the jobs with release date R_1.

(2) Each family that contains just one job with release date R_1 acts as a batch and is scheduled first in any order.

(3) At any decision time t with $R_1 \leq t < R_2$, if $t + s < R_2$, pick a family $\mathcal{F}_i$ with two jobs such that

$$p_{(i,1)} = \max\{p_{(f,1)} : |\mathcal{F}_f| = 2\}$$

(if any) and let $\{p_{(i,1)}\}$ act as a batch starting at t.

(4) The remaining families or subfamilies, each of which acts as a batch, are scheduled in an arbitrary order beginning at $\max\{t, R_2\}$, where t is the current decision time.

It is easy to see that the complexity of the algorithm is $O(n) + O(m \log m)$. □

Acknowledgements

This research was supported in part by The Hong Kong Polytechnic University under grant numbers G-S818. The first author was also supported by NSFC(10371112), NSFHN(0411011200) and SRF for ROCS, SEM. The second author was also supported by NSFC(10101007).

References

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

[2] J. Bruno and P. Downey, Complexity of task sequencing with deadlines, setup times and changeover costs, *SIAM Journal on Computing*, **7**(1978), 393-404.

[3] T.C.E. Cheng, C.T. Ng and J.J. Yuan, The single machine batching problem with family setup times to minimize maximum lateness is strongly NP-hard, *Journal of Scheduling*, **6**(2003), 483-490.

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

[5] J.B. Ghosh and J.N.D. Gupta, Batch scheduling to minimize maximum lateness, *Operations Research Letters*, **21**(1997), 77-80.

[6] C.L. Monma and C.N. Potts, On the complexity of scheduling with batch setup times, *Operations Research*, **37**(1988), 798-804.

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