Batch scheduling with incompatible job families and common due date

Lili Liu1 *

1 College of Arts and Sciences, Shanghai Polytechnic University, Shanghai 201209, China
*Corresponding author’s e-mail: llliu@sspu.edu.cn

Abstract. This paper explores incompatible batch scheduling with common due date on a single batch processing machine. The problem of minimizing the number of tardy jobs is proved to be NP-hard with respect to id-encoding, and pseudo-polynomial time algorithms for minimizing the number of tardy jobs and the weighted number of tardy jobs are presented, respectively.

1. Introduction
For the burn-in operation of the final testing stage in the very large-scale integrated circuit manufacturing, a machine can process several jobs simultaneously, then batch scheduling problem is brought about. The processing time of each batch is equal to the longest processing time of the jobs assigned to it. If jobs having different processing times cannot be assigned to the same batch, we call it incompatible batch scheduling. We say that jobs having same processing time belong to the same family, where jobs of different families cannot be processed together. For batch scheduling with incompatible job families, Uzsoy[1] explores several scheduling objectives on single batch processing machine and parallel batch processing machines. The problem of minimizing the total tardiness is proved to be strongly NP-hard by Mehta and Uzsoy[2] and a dynamic programming algorithm and approximation algorithms are developed. Jolai[3] examines the problem of minimizing the number of tardy jobs, he provides NP-hardness proof and a dynamic programming algorithm. Perez et al.[4] try to study the problem of minimizing the total weighted tardiness, they present several heuristic algorithms for this problem.

We shall first describe some notations that we will use for the rest of this paper. There are some jobs awaiting processing on a single batch processing machine. These jobs comprise \( m \) different job families. For \( i = 1, \cdots, m \), we denote the number of jobs of family \( i \) by \( n_i \), the common processing time of family \( i \) by \( p_i \), and the common due date of all the jobs by \( d \). For \( i = 1, \cdots, m; j = 1, \cdots, n_i \), job \( J_{ij} \) of family \( i \) may also have a weight \( w_{ij} \), it indicates the importance of job \( J_{ij} \) relative to the other jobs. A batch processing machine can process at most \( B \) jobs at the same time, if a batch contains exactly \( B \) jobs, we call it a full batch, otherwise, it is a partial batch. For batch \( Q \), we use \( p(Q) \) to denote its processing time and \( |Q| \) to denote the number of jobs in it. In a feasible schedule, let \( C_{ij} \) denote the completion time of job \( J_{ij} \), \( U_{ij} \) its unit tardy penalty which is defined as \( U_{ij} = 1 \) if \( C_{ij} > d_{ij} \) and zero otherwise. This paper examines incompatible batch scheduling with common due date...
date on a single batch processing machine to minimize the number of tardy jobs and the weighted number of tardy jobs.

For problems under consideration, in Section 2 we show that minimizing the number of tardy jobs on a single incompatible batch processing machine with common due date is NP-hard with respect to id-encoding, and pseudo-polynomial time dynamic programming algorithms are developed to minimize the number of tardy jobs and the weighted number of tardy jobs, respectively.

2. Minimizing the (weighted) number of tardy jobs

2.1 Minimizing the number of tardy jobs

For incompatible batch scheduling with a common due date on a single machine, we show that minimizing the number of tardy jobs is NP-hard with respect to id-encoding [5] by reducing from the Even-Odd Partition problem.

**Even-Odd Partition:** There are $2t$ positive integers $\{a_1, \ldots, a_{2t}\}$ with $a_1 < a_2 < \cdots < a_{2t-1} < a_{2t}$ and $\sum a_j = 2A$, is there a partition of the index set $\{1, \ldots, 2t\}$ into $A_1, A_2$ such that $\sum_{j \in A_1} a_j = \sum_{j \in A_2} a_j = A$ and $A_i$ contains exactly one of $\{2i-1, 2i\}$ for $i = 1, \ldots, t$?

**Theorem 1** Minimizing the number of tardy jobs on a batch processing machine with incompatible job families and common due date is NP-hard with respect to id-encoding.

**Proof** Given an instance of the Even-odd Partition problem, we can create an instance of incompatible batch scheduling with $t$ families. For $i = 1, \ldots, t$, create $a_i$ jobs with processing times $a_i$. Let the capacity of a batch processing machine $B = a_{2t}$ and the common due date of all the jobs $d = A$. We will prove that a schedule with $\sum U_j \leq A$ exists for the scheduling instance if and only if the Even-Odd Partition instance has a solution.

Suppose the Even-Odd Partition instance has a solution with $\sum_{i \in A_1} a_i = \sum_{i \in A_2} a_i = A$. Since $B = a_{2t}$, for $i = 1, \ldots, t$, assign the $a_i$ jobs with processing time $a_i$ to batch $B_i$ before the common due date $d = A$ in an arbitrary order if $i \in A_1$, and process batch $B_i$ on or after the common due date $d = A$ in an arbitrary order if $i \in A_2$. Thus the $a_i$ jobs in batches $B_i (i \in A_1)$ are on time and the $a_i$ jobs in batches $B_i (i \in A_2)$ are tardy. We have $\sum U_j = \sum_{i \in A_1} a_i = A$. Hence, if the solution of Even-odd Partition instance exists, a feasible schedule with $\sum U_j \leq A$ also exists for the created scheduling instance.

Conversely, if there exists a feasible schedule for the created scheduling instance such that $\sum U_j \leq A$, there must exist a feasible schedule where the jobs in the same family are assigned to one batch since $B = a_{2t}$. If the jobs with processing time $a_i$ are finished before or on common due date $d = A$, denote the set of indices $i (i = 1, \ldots, t)$ as $A_1$; if the $a_i$ jobs with processing time $a_i$ are finished after the common due date $d = A$, denote the set of indices $i (i = 1, \ldots, t)$ as $A_2$. Hence, we have $\sum U_j = \sum_{i \in A_1} a_i \leq A$. Because the common due date is $A$, we also have $\sum_{i \in A_2} a_i \leq A$. Therefore, $A_1$ and $A_2$ constitute a solution of the Partition instance.

Since we copy job’s parameter to create identical jobs with the same characteristics, the problem reduction takes pseudo-polynomial time, to minimize the number of tardy jobs is NP-hard with respect
to id-encoding. We give two very useful properties of an optimal schedule as follows.

**Lemma 1** [6] There must exist an optimal schedule where the batches of each family are full except possibly the last one.

**Lemma 2** There exists an optimal schedule such that the full batches are arranged in non-decreasing order of their processing times.

*Proof* Consider an optimal schedule with full batch $B_i$ processed before full batch $B_k$ and $p(B_i) > p(B_k)$. Denote the completion times of batches $B_i$ and $B_k$ by $C_1$ and $C_2$. Exchange batches $B_i$ with $B_k$, since jobs in the same family have identical processing times, the completion times of the other batches will not increase. Let $C'_1$ and $C'_2$ denote the updated completion times of batches $B_i$ and $B_k$. We obtain $C'_1 \leq C_1$ and $C'_2 = C_2$. Since all the jobs have common due date and the moved batches are both full, the number of tardy jobs will not increase after the batches interchange.

Basing on Lemma 1 and Lemma 2, we will develop a dynamic programming algorithm. For each family, schedule the jobs in an arbitrary order, then from the beginning assign adjacent batches into a batch until all the jobs have been batched. For $i = 1, \cdots, m$, there are exactly $\left\lfloor \frac{n_i}{B} \right\rfloor$ full batches and one possible partial batch with processing time $p_i$, where $\left\lfloor r \right\rfloor$ denotes the largest integer smaller than or equal to $r$. We partition all the batches into two subsets: full batches and partial batches. According to Lemma 2, there must exist an optimal schedule such that the full batches are arranged in non-decreasing order of their processing times. Order all the full batches in non-decreasing order of their processing times, denote the sum of processing times of the first $k$ full batches as $P_k \left( k = 1, \cdots, \sum_{i=1}^{m} \left\lfloor \frac{n_i}{B} \right\rfloor \right)$. For the partial batches, we next present a dynamic programming algorithm.

Order the partial batches in non-decreasing order of their processing times, denote the partial batch with processing time $p_i$ as $B_i$. Let $F(j, t)$ denote the minimum number of tardy jobs when partial batches $B_1, \cdots, B_j$ are scheduled, and the completion time of the last on time batch is $t$. We have

$$F(0, t) = \begin{cases} 0, & \text{if } t = 0, \\ \infty, & \text{otherwise.} \end{cases}$$

For $j = 1, \cdots, m$ and $t = 0, \cdots, d$, the recursive equation is

$$F(j, t) = \min \left\{ F\left( j-1, t-p_j \right), F\left( j-1, t \right) + \left\lfloor \frac{n_j}{B} \right\rfloor \right\}.$$ 

Basing on the properties of an optimal schedule and the dynamic programming algorithm of the partial batches, the minimum number of tardy jobs for all the jobs is equal to

$$\min_{0 \leq t \leq d} \left\{ F(m, t) + \left( \sum_{i=1}^{m} \left\lfloor \frac{n_i}{B} \right\rfloor - k \right) B t + P_k \leq d \right\},$$

and the optimal schedule can be obtained by backtracking. The worst case time complexity of this algorithm is $O(md)$, which runs in pseudo-polynomial time.

### 2.2 Minimizing the weighted number of tardy jobs

The problem of minimizing the weighted number of tardy jobs with common due date on a single unit capacity machine is NP-hard[7], hence the single machine incompatible batch scheduling with common due date to minimize the weighted number of tardy jobs is at least binary NP-hard. We next develop a pseudo-polynomial time dynamic programming algorithm, which indicates that it cannot be strongly NP-hard unless P=NP.
A schedule is said in batch LWF-order[6] if there are two batches \( P \) and \( Q \) with batch \( P \) processed before batch \( Q \) and there does not exist two jobs \( J_k, J_l \) with \( J_k \in P, J_l \in Q \) and \( w_{j_k} < w_{j_l} \).

**Lemma 3** [6] For single machine incompatible batch scheduling to minimize the weighted number of tardy jobs, if all the jobs are subject to a common due date, there must exist an optimal schedule such that the batches of each family are in batch LWF-order.

According to Lemma 1 and Lemma 3, for \( i = 1, \ldots, m \), sequence the jobs of family \( i \) in non-increasing order of their weights, then from the beginning assign adjacent \( B \) jobs into a batch until all the jobs in family \( i \) have been batched. The number of batches of family \( i \) is equal to \( \lceil \frac{n_i}{B} \rceil \), where \( \lceil r \rceil \) denotes the smallest integer larger than or equal to \( r \). Then order all the batches in non-decreasing order of their processing times, denote the \( j \)-th batch as \( B_j \), and the total weight of jobs in batch \( B_j \) as \( W(B_j) \). Let \( F(j, t) \) be the minimum weighted number of tardy jobs when batches \( B_1, \ldots, B_j \) are assigned, and the completion time of the last on time batch is \( t \). We propose the following dynamic programming algorithm to sequence all the batches.

\[
F(0, t) = \begin{cases} 
0, & \text{if } t = 0, \\
\infty, & \text{otherwise.}
\end{cases}
\]

For \( j = 1, \ldots, \sum_{i=1}^{m} \left\lceil \frac{n_i}{B} \right\rceil \) and \( t = 0, \ldots, d \), the recursive relation is

\[
F(j, t) = \min \left\{ F(j-1, t-p(B_j)), F(j-1, t) + W(B_j) \right\}.
\]

The dynamic programming algorithm runs in at most \( O(d \sum_{i=1}^{m} \left\lceil \frac{n_i}{B} \right\rceil) \) time, which is pseudo-polynomial time.

3. **Conclusion**

This paper explores the single machine incompatible batch scheduling problems with common due date. We prove that minimizing the number of tardy jobs is NP-hard with respect to id-encoding, and we develop dynamic programming algorithms for minimizing the (weighted) number of tardy jobs.

Developing approximation algorithms for the problems under study and investigating their special cases are very interesting directions for our future research.

**Acknowledgements**

This work is supported in part by the National Natural Science Foundation of China (No. 11601316) and is also supported by discipline “Applied Mathematics” of Shanghai Second Polytechnic University (No. XXKPY1604).

**References**

[1] Uzsoy R. (1995) Scheduling batch processing machines with incompatible job families, International Journal of Production Research, 33, 2685-2708.

[2] Mehta S., Uzsoy R. (1998) Minimizing total tardiness on a batch processing machine with incompatible job families, IIE Transactions 30, 165-178.

[3] Jolai F. (2005) Minimizing number of tardy jobs on a batch processing machine with incompatible job families, European Journal of Operational Research, 162, 184-190.

[4] Perez I., Fowler J., Carlyle W. (2005) Minimizing total weighted tardiness on a single batch process
machine with incompatible job families, Computers and Operations Research, 32, 327-341.

[5] Rinnooy Kan, A.H.G. (1976) Machine Scheduling Problems: Classification, Complexity and Computations. The Hague: Nijhoff.

[6] Liu L.L., Fu C.Y. (2020) Batch scheduling with incompatible job families and common due date in automatic drug dispensing system. Algorithmic Aspects in Information and Management, Springer Science and Business Media LLC, 537-543.

[7] Karp, R.M. (1972) Reducibility among combinatorial problems. Complexity in Computer Computations, 85-103. New York: Plenum Press.