Annals of Operations Research 133, 149–161, 2005


# Two-Machine Flowshop Batching and Scheduling

B.M.T. LIN bmtlin@mail.nctu.edu.tw
*Department of Information and Finance Management, National Chiao Tung University, Hsinchu, Taiwan 545*

T.C.E. CHENG LGTcheng@inet.polyu.edu.hk
*Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong*

**Abstract.** We consider in this paper a two-machine flowshop scheduling problem in which the first machine processes jobs individually while the second machine processes jobs in batches. The forming of each batch on the second machine incurs a constant setup time. The objective is to minimize the makespan. This problem was previously shown to be NP-hard in the ordinary sense. In this paper, we first present a strong NP-hardness result of the problem. We also identify a polynomially solvable case with either anticipatory or non-anticipatory setups. We then establish a property that an optimal solution for the special case is a lower bound for the general problem. To obtain near-optimal solutions for the general problem, we devise some heuristics. The lower bound is used to evaluate the quality of the heuristic solutions. Results of computational experiments reveal that the heuristics produce solutions with small error ratios. They also suggest that the lower bound is close to the optimal solution.

**Keywords:** production scheduling, flowshop, batch processing, makespan, strong NP-hardness, lower bound, heuristics

## 1. Introduction

Flowshop scheduling, initiated by Johnson (1954), is one of the most extensively studied topics in scheduling research. In a recent paper, Cheng and Wang (1998) consider batch scheduling problems in a two-machine flowshop which comprises a discrete processor and a batch processor. There is a set of jobs simultaneously available for processing in the flowshop. All jobs are first processed by the discrete processor, which processes one job at a time. The batch processor processes the jobs, transferred from the discrete processor, in batches. The forming of each batch on the batch processor incurs a constant setup time. The processing time of a batch on the batch processor is the sum of the constant setup time and the processing times of all jobs belonging to it. All jobs in a batch have a common completion time which is equal to the completion time of the last job of the batch. The objective is to batch, as well as schedule, the jobs so as to minimize the makespan.

This scheduling problem arises from the manufacturing of custom-built very large-scale integrated circuits by flexible manufacturing cells organized into a two-stage flow-line. In the first stage, chips of various types are picked and placed on a circuit board according to its individual technical specifications by a pick and insertion machine –

the discrete processor. Each circuit board is unique and represents a discrete job. Upon completion of this operation, the circuit board is loaded onto a pallet to be transferred to the second stage for further processing. Circuit boards loaded on the same pallet correspond to a batch. Pallets are installed and removed by a robot before and after processing on the second machine. The fixed time incurred in removing a previous pallet and installing a new one is the constant setup time. In the second stage, each pallet will have its circuit boards soldered and tested one at a time by an integrated soldering and testing equipment – the batch processor. This is a highly sophisticated and expensive piece of equipment and so operations on it are performed in batches to minimize idle time caused by frequent setups. Thus, the batch processing time is equal to the sum of the setup time and the individual soldering and testing time for each circuit board loaded on the same pallet. The objective of the scheduling problem is to determine the optimal sequence and batching compositions so as to minimize the makespan.

In the literature, most of the previous research on batch scheduling problems is related to the single machine case to minimize the total flow time. An $O(n)$ algorithm is first given by Naddef and Santos (1988) for the case where all jobs have the same processing time. Coffman, Nozari and Yannakakis (1989) later devise an improved $O(\sqrt{n})$ algorithm. Albers and Brucker (1993) identify many NP-hard problems in which precedence constraints between jobs are imposed. They also show that many problems can be transformed into the shortest path problem, which is solvable by an algorithm with a time complexity linear in the number of vertices visited. In (Cheng, Kovalyov and Lin, 1997), the problem of minimizing the sum of total weighted job earliness and mean batch delivery penalties is investigated. On $m$ parallel machines, Cheng et al. (1996) present an $O(n^{m+2})$ dynamic programming algorithm to determine the minimum total completion time.

Considering the two-machine flowshop environment, Ahmadi et al. (1992) study a class of batching and scheduling problems. In their model, a batch can accommodate all of its jobs simultaneously and the batch processing time is constant and independent of the batch size. A study similar to the problem of interest in this paper is conducted by Cheng, Lin and Toker (2000). In the two-machine flowshop environment they consider, both machines process the jobs in batches. After processing on the first machine, a batch is transferred to the second machine with the batch composition preserved. They prove that the general problem is strongly NP-hard and propose polynomial algorithms for some special cases. Glass, Potts and Strusevich (2001) consider a similar problem where setups on the second machine are anticipatory. The reader is referred to (Cheng, Gupta and Wang, 2000; Allahverdi, Gupta and Aldowaisan, 1999; Potts and Kovalyov, 2000) for comprehensive surveys on batch scheduling in flowshop environments.

The rest of this paper is organized as follows. In section 2, we introduce the notation that will be used throughout this paper. In section 3, we present the strong NP-hardness result by a reduction from 3-PARTITION. We identify special cases that are polynomially solvable in section 4. Besides, we also establish an interesting property concerning a lower bound for the general problem. In section 5, four heuristic methods are given to generate sub-optimal solutions. A series of computational experiments is

conducted, and the results demonstrate the practical effectiveness of the proposed heuristics. Finally, some conclusions are given in section 6.

## 2. Problem formulation

In this section, we present the notation that will be used throughout this paper.

**Notation.**

- $N = \{1, 2, \ldots, n\}$: the job set to be processed;
- $p_i$: the processing time of job $i$ on the discrete processor, $1 \leqslant i \leqslant n$;
- $q_i$: the processing time of job $i$ on the batch processor, $1 \leqslant i \leqslant n$;
- $s$: batch setup time;
- $B_l$: batch $l$;
- $P_l$: total processing time of batch $B_l$ on the discrete processor;
- $Q_l$: total processing time of batch $B_l$ on the batch processor;
- $Z(S)$: makespan of schedule $S$;
- $S^*$: optimal schedule;
- $Z^*$: optimal makespan.

We adopt the three-field notation $F(\delta \rightarrow \beta)//C_{\max}$ used by Cheng and Wang (1998) to denote the general problem. In this notation, $\delta$ and $\beta$ stand for the discrete processor and the batch processor, respectively. In this paper, both cases with non-anticipatory and anticipatory setups, denoted by $F(\delta \rightarrow \beta)/ns/C_{\max}$ and $F(\delta \rightarrow \beta)/as/C_{\max}$, respectively, are considered. Non-anticipatory setups mean that all jobs forming a batch must be present at the batch processor for batch initialization, and anticipatory setups mean that the setup for the batch processor can be performed prior to arrivals of jobs which form a batch from the discrete processor. Due to the batch considerations, to compose a schedule, we needs to determine how the jobs are grouped into batches and how the batches are sequenced. In this paper, we consider permutation sequences only, i.e. the processing sequences on both machines are the same.

## 3. Strong NP-hardness

In this section, we show that both $F(\delta \rightarrow \beta)/ns/C_{\max}$ and $F(\delta \rightarrow \beta)/as/C_{\max}$ are strongly NP-hard by a reduction from 3-PARTITION, which is known to be NP-hard in the strong sense (see (Garey and Johnson, 1979)).

3-PARTITION: Given an integer $M$ and a set $A$ of $3n$ positive integers $\{x_1, x_2, \ldots, x_{3n}\}$, $M/4 < x_i < M/2,\ 1 \leqslant i \leqslant 3n$, such that $\sum_{i=1}^{3n} x_i = nM$, does there exist a partition $A_1, A_2, \ldots, A_n$ of the set $A$ such that $|A_l| = 3$ and $\sum_{x_i \in A_l} x_i = M,\ 1 \leqslant l \leqslant n$?

Before proceeding to the main result, we first assume without loss of generality that $M > n + 1$ in 3-PARTITION. If it is not the case, we may scale the instance by multiplying the value of each $x_i$ with integer $n$.

**Theorem 1.** The $F(\delta \rightarrow \beta)/ns/C_{\max}$ problem is strongly NP-hard even if $p_i \leqslant q_i$ for all job $i$.

*Proof.* It is clear that the decision version of $F(\delta \rightarrow \beta)/ns/C_{\max}$ is in NP. We next perform a polynomial-time reduction from 3-PARTITION. Given an instance of 3-PARTITION, we construct an instance of $F(\delta \rightarrow \beta)/ns/C_{\max}$ consisting of $4n + 1$ jobs as follows:

Initial job: $p_0 = 0,\ q_0 = \omega_1 + \omega_2 + 2M^2$;
Ordinary jobs: $p_i = x_i M,\ q_i = 2x_i M,\ 1 \leqslant i \leqslant 3n$;
Enforcer jobs: $p_{3n+i} = \omega_1 + i\omega_2 + M^2 + 1,\ q_{3n+i} = \omega_1 + (i+1)\omega_2,\ 1 \leqslant i \leqslant n$,
where $\omega_2 > 4nM^2$ and $\omega_1 > 2n\omega_2$;
Setup time $s = 1$.

Note that, in the above instance, $p_i \leqslant q_i$ for any job $i,\ 1 \leqslant i \leqslant 4n + 1$. The sum of the processing times of all jobs on the discrete processor is $2nM^2 + n + n\omega_1 + n(n+1)\omega_2/2$; the sum of the processing times of all jobs on the batch processor is $(n+1)\omega_1 + (n+1)\omega_2 + 2(n+1)M^2 + n(n+1)\omega_2/2$. We claim that there is a desired partition for the set $A$ if and only if there exists an optimal schedule for the instance of $F(\delta \rightarrow \beta)/ns/C_{\max}$ with a makespan $C_{\max} \leqslant (n+1)\omega_1 + (n+1)\omega_2 + 2(n+1)M^2 + n(n+1)\omega_2/2 + (n+1)$.

($\Longrightarrow$) Let the subsets $A_1, A_2, \ldots, A_n$ be a partition as specified for set $A$ in 3-PARTITION. Let the initial job be the first job, and let it solely form batch $B_0$ on the batch processor. Let job $3n + l$ and the jobs corresponding to the elements of $A_l$ form batch $B_l,\ 1 \leqslant l \leqslant n$. Figure 1 illustrates the configuration of the schedule. In the derived schedule, the makespan is $(n+1)\omega_1 + (n+1)\omega_2 + 2(n+1)M^2 + n(n+1)\omega_2/2 + (n+1)$,

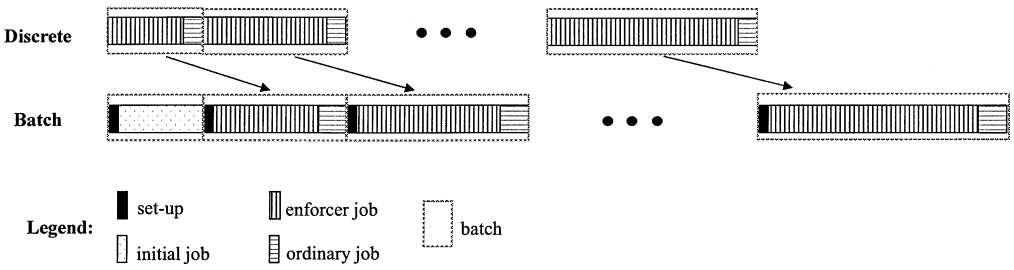


Figure 1. Configuration of the optimal schedule in theorem 1.

where the last item $(n+1)$ corresponds to the sum of the setup times incurred by the $n+1$ batches.

($\Longleftarrow$) Suppose that there is an optimal schedule $S$ for $F(\delta \to \beta)/ns/C_{\max}$ whose completion time is no more than $(n+1)\omega_1+(n+1)\omega_2+2(n+1)M^2+n(n+1)\omega_2/2+(n+1)$. For schedule $S$, the sum of the idle times and setup times on the batch processor cannot be greater than $n+1$.

First of all, we know that the number of batches on the batch processor must be less than or equal to $n+1$. Furthermore, because any job, except the initial job, has a processing time longer than $n+1$ on the discrete processor, the first batch must contain only the initial job.

Recall that $P_l$ and $Q_l$, $0 \leqslant l \leqslant n$, denote the processing times, including the setup time, of batch $B_l$ on the discrete processor and the batch processor, respectively.

**Fact 1.** In schedule $S$, each batch contains at most one enforcer job.

*Proof.* Assume batch $B_l$ is the first batch containing two or more enforcer jobs. The processing time of $B_l$ will be no less than $2\omega_1$. Let $\eta$ be the total idle time before the completion of all predecessor batches $B_0, B_1, B_2, \ldots, B_{l-1}$ on the batch processor. That is, these predecessor batches finish their processing on the batch processor at time $\eta+\sum_{i=0}^{l-1} Q_i$. Therefore, the total idle time before the completion of batch $B_l$ on the batch processor is no less than $\eta+\max\{0, 2\omega_1+\sum_{i=0}^{l-1} P_i-(\eta+\sum_{i=0}^{l-1} Q_i)\}$. If $2\omega_1+\sum_{i=0}^{l-1} P_i-(\eta+\sum_{i=0}^{l-1} Q_i) \geqslant 0$, the total idle time is no less than $2\omega_1-\sum_{i=0}^{l-1}(Q_i-P_i)$. On the other hand, if $2\omega_1+\sum_{i=0}^{l-1} P_i-(\eta+\sum_{i=0}^{l-1} Q_i) < 0$, the total idle time is no less than $\eta$, which then is greater than $2\omega_1-\sum_{i=0}^{l-1}(Q_i-P_i)$. In other words, the total idle time before the completion of batch $B_l$ on the batch processor is no less than $2\omega_1-\sum_{i=0}^{l-1}(Q_i-P_i)$.

The value $\sum_{i=0}^{l-1}(Q_i-P_i)$ is maximized by the contributions made by the initial jobs, all ordinary jobs, the $n-2$ enforcer jobs, and the setup times of up to $n$ batches. It is evaluated to be $(\omega_1+\omega_2+2M^2)+nM^2+(n-2)(\omega_2-M^2-1)+n$. By subtracting this number from $2\omega_1$, we have

$$\begin{aligned}
&\omega_1-\left[(n-1)\omega_2+4M^2+2\right]\\
&\quad>(n+1)\omega_2-4M^2-2\\
&\quad>4n^2M^2+4nM^2-4M^2-2\\
&\quad>4n^2M^2-2\\
&\quad>n+1 \quad \text{(because } M>n+1\text{)}.
\end{aligned}$$

In other words, the induced idle time is greater than $n+1$, a contradiction. Therefore, any batch cannot contain more than one enforcer job. □

With the above observations, we have that schedule $S$ must contain exactly $n+1$ batches, say $B_0, B_1, B_2, \ldots, B_n$, such that the initial job is the only element in $B_0$ and the enforcer job $3n+l$, $1 \leqslant l \leqslant n$, belongs to batch $B_l$. Besides, no idle time is permitted in schedule $S$.

**Fact 2.** In schedule $S$, $P_l = \omega_1 + l\omega_2 + 2M^2 + 1,\ 1 \leqslant l \leqslant n$.

*Proof.* Because on the discrete processor a batch containing the enforcer job $3n+i$ has a shorter processing time than any other batch containing the enforcer job $3n + j$ with $1 \leqslant i < j \leqslant n$, we assume that in schedule $S$ the enforcer job $3n + l$ is included in batch $B_l$, $1 \leqslant l \leqslant n$. Let $P_l = \omega_1 + i\omega_2 + M^2 + \rho_l + 1,\ 1 \leqslant l \leqslant n$, where $\rho_l$ corresponds to the sum of the processing times of the ordinary jobs in batch $B_l$. We now show that $\rho_1 = \rho_2 = \cdots = \rho_n = M^2$ must hold.

Assume that $\rho_1 \leqslant \rho_2 \leqslant \cdots \leqslant \rho_n$, for otherwise we may exchange, without increasing the makespan, all ordinary jobs of any two consecutive batches not satisfying the inequality. Let $k,\ k > 1$, be the first batch such that $\rho_k > M^2$. We now determine the idle time incurred before batch $B_k$ on the batch processor. On the discrete processor, the total processing time of jobs in $B_0, B_1, B_2, \ldots, B_k$ is

$$\begin{aligned}&\big(\omega_1 + \omega_2 + M^2 + \rho_1 + 1\big) + \big(\omega_1 + 2\omega_2 + M^2 + \rho_2 + 1\big) + \cdots \\ &\quad + \big(\omega_1 + k\omega_2 + M^2 + \rho_k + 1\big). \end{aligned} \tag{1}$$

On the batch processor, the total processing time of jobs in $B_0, B_1, B_2, \ldots, B_{k-1}$ is

$$\begin{aligned}&\big(\omega_1 + \omega_2 + 2M^2 + 1\big) + \big(\omega_1 + 2\omega_2 + 2\rho_1 + 1\big) + \big(\omega_1 + 3\omega_2 + 2\rho_2 + 1\big) + \cdots \\ &\quad + \big(\omega_1 + k\omega_2 + 2\rho_{k-1} + 1\big). \end{aligned} \tag{2}$$

Subtracting (1) from (2) yields

$$\begin{aligned}&\big(\omega_1 + \omega_2 + 2M^2 + 1\big) + \big(\omega_2 - M^2 + \rho_1\big) + \big(\omega_2 - M^2 + \rho_2\big) + \cdots \\ &\quad + \big(\omega_2 - M^2 + \rho_{k-1}\big) - \big(\omega_1 + k\omega_2 + M^2 + \rho_k + 1\big) \\ &= \big(\rho_1 - M^2\big) + \big(\rho_2 - M^2\big) + \cdots + \big(\rho_{k-1} - M^2\big) - \rho_k + M^2 \\ &< 0 \quad (\text{because } \rho_k > M^2). \end{aligned}$$

In other words, nonzero idle time will be incurred before batch $B_k$ on the batch processor, a contradiction. Therefore, $\rho_l = M$ must hold for $l = 1, 2, \ldots, n$ and the proof is concluded. □

*Proof of 3*-PARTION. With fact 2, a partition for the set $A$ is obtained by letting the elements corresponding to the ordinary jobs in batch $B_l,\ 1 \leqslant l \leqslant n$, to form the subset $A_l$. Then, $\sum_{x_i \in A_l} x_i = M$, and $|A_l| = 3$ because $M/4 \leqslant x_i \leqslant M/2$. □

Referring to figure 1 for the proof of $F(\delta \to \beta)/ns/C_{\max}$, we see that in the optimal schedule, there is no idle time incurred on the batch processor. That is, the proof is valid regardless of whether anticipatory or non-anticipatory setups are considered.

**Corollary 1.** The $F(\delta \to \beta)/as/C_{\max}$ problem is strongly NP-hard even if $p_i \leqslant q_i$ for all job $i$.

## 4. Polynomially solvable cases and their implications

In this section, we consider two special cases where a total order on the job set is satisfied and show their polynomial solvability. We first focus the study on $F(\delta \to \beta)/ns/C_{\max}$. The results are similarly valid for the special case of $F(\delta \to \beta)/as/C_{\max}$.

Define a total order $\preceq$ on the set of jobs as a relation between any pair of jobs $i$ and $j$ such that $i \preceq j$ if and only if $p_i \leqslant p_j$ and $q_i \geqslant q_j$. For example, the relation $\preceq$ is satisfied in the case where $p_i = p$ for all $i$. We denote the special case that satisfies the relation $\preceq$ by $F(\delta \to \beta)/ns, \preceq /C_{\max}$.

**Lemma 1.** For $F(\delta \to \beta)/ns, \preceq /C_{\max}$, there is an optimal schedule in which for jobs $i$ and $j$, if $i \preceq j$, then either job $i$ precedes job $j$ or they are in the same batch.

*Proof.* Suppose that there are jobs $i \in B_l$ and $j \in B_m$ in some optimal schedule with $i \preceq j$ and $l > m$. By swapping the positions of jobs $i$ and $j$, the idle time before any batch on the batch processor may be decreased or unchanged. Therefore, the total idle time will not be increased by the job interchange operation. □

By virtue of lemma 1, we may renumber the jobs in $F(\delta \to \beta)/ns, \preceq /C_{\max}$ such that $1 \preceq 2 \preceq 3 \preceq \cdots \preceq n$.

**Lemma 2.** For $F(\delta \to \beta)/ns, \preceq /C_{\max}$ with the job set $N'$, let $S'^*$ be optimal among all schedules for the subset $N' - \{r, r+1, \ldots, n\}$. Then, for the set $N'$, schedule $S'^* \cup B_k$, where $B_k$ is the batch comprising jobs $r, r+1, \ldots, n$, is optimal among all schedules with $B_k$ as the last batch.

*Proof.* Let $S'$ be any schedule for $N' - B_k$. Batches $S'$ and $S'^*$ have the same completion time on the discrete processor. Furthermore, the completion time of $S'^*$ on the batch processor is the same or smaller than that of $S'$. It is evident that $Z(S'^* \cup B_k)$ is no greater than $Z(S' \cup B_k)$. The proof readily follows. □

Based on lemmas 1 and 2, we can construct a simple dynamic programming algorithm to solve $F(\delta \to \beta)/ns, \preceq /C_{\max}$. Let $C(i)$ denote the makespan of an optimal solution for scheduling jobs in $\{1, 2, \ldots, i\}$. An optimal schedule for $F(\delta \to \beta)/ns, \preceq /C_{\max}$ with the job set $N'$ can be obtained by the following algorithm.

**Algorithm NS.**
*Line 1*: $C(0) = 0$.
*Line 2*: For $i = 1$ to $n$ do *line 3*.
*Line 3*: $C(i) = \min_{l=1,2,\ldots,i}\{\max\{\sum_{j=1}^{i} p_j, C(i-l)\} + (s + \sum_{j=i-l+1}^{i} q_j)\}$.
*Line 4*: Return $C(n)$.
*Line 5*: Stop.

In the recurrence relation in *line 3*, variable $l$ indicates the number of jobs to be included in the last batch $B_k$. If an $O(n^2)$ preprocessing procedure is used to compute the values of $\sum_{j=1}^{i} p_j$ and $\sum_{j=i-l+1}^{i} q_j$, each iteration of *line 3* can be done in $O(n)$ time. As *line 3* iterates $n$ times, the overall time complexity of *algorithm NS* is $O(n^2)$.

**Theorem 2.** The problem $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ can be solved in $O(n^2)$ time.

Next, we give an insight into the relationship between $F(\delta \rightarrow \beta)/ns/C_{\max}$ and $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$. Let $N$ be a job set for $F(\delta \rightarrow \beta)/ns/C_{\max}$. We transform $N$ into a job set $N'$ for $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ by defining $p'_i$ as the $i$th smallest $p_i$ in $N$ and $q'_i$ as the $i$th largest $q_i$ in $N$. Clearly, the job set $N'$ satisfies the total order $\preceq$ in $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$.

**Lemma 3.** For any schedule $S$ for $F(\delta \rightarrow \beta)/ns/C_{\max}$ with the job set $N$, there is a schedule $S'$ for $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ with the job set $N'$ such that $Z(S') \leqslant Z(S)$.

*Proof.* Given any schedule $S$ for $F(\delta \rightarrow \beta)/ns/C_{\max}$ with the job set $N$, we make the following transformation: For any jobs $i$ and $j$ in schedule $S$, if job $j$ precedes job $i$ and $p_i \leqslant p_j$, then we swap the operations of $i$ and $j$ on the discrete processor but leave the operations on the batch processor unchanged. That is, the processing times of job $i$ on both machines are $p_j$ and $q_i$, and the processing times of job $j$ are $p_i$ and $q_j$. It is not hard to see that the makespan will not increase. Continuing the interchange process, we will have a new schedule in which all jobs on the discrete processor are arranged in nondecreasing order of their processing times. With the derived schedule, we further adjust the operations on the batch processor such that the operations are arranged in nonincreasing order of their processing times. Similarly, the makespan will not increase. Note that the jobs in the final schedule make up the job set $N'$. Therefore, the proof is complete. □

This lemma suggests that, given an instance of $F(\delta \rightarrow \beta)/ns/C_{\max}$ with $N$, we can find an optimal schedule $S'^*$ for an instance of the problem $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ with $N'$, and $Z(S'^*)$ is a lower bound for the optimal solution for the original problem. Therefore, we conclude with the following theorem.

**Theorem 3.** For any schedule $S$ for $F(\delta \rightarrow \beta)/ns/C_{\max}$ with the job set $N$ and an optimal schedule $S'^*$ for the corresponding $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ with the job set $N'$, $Z(S'^*) \leqslant Z(S)$.

When considering the other case $F(\delta \rightarrow \beta)/as, \preceq /C_{\max}$, it is not hard to see that the above results are still valid if we replace the recurrence formula in *line 3* of

*algorithm NS* by

$$C(i) = \min_{l=1,2,\ldots,i} \left\{ \max \left\{ \sum_{j=1}^{i} p_j,\ C(i-l) + s \right\} + \sum_{j=i-l+1}^{i} q_j \right\}.$$

The corresponding solution method is denoted by *algorithm AS*. We have the results for the problem $F(\delta \to \beta)/as, \preceq /C_{\max}$ in the following corollaries.

**Corollary 2.** The problem $F(\delta \to \beta)/as, \preceq /C_{\max}$ can be solved in $O(n^2)$ time.

**Corollary 3.** For any schedule $S$ for $F(\delta \to \beta)/as/C_{\max}$ with the job set $N$ and an optimal schedule $S'^*$ for the corresponding $F(\delta \to \beta)/as, \preceq /C_{\max}$ with the job set $N'$, $Z(S'^*) \leqslant Z(S)$.

## 5. Heuristics

In this section, we present heuristics to find approximation solutions for $F(\delta \to \beta)/ns/C_{\max}$ and $F(\delta \to \beta)/as/C_{\max}$. We focus our study first on $F(\delta \to \beta)/ns/C_{\max}$. Given an instance of $F(\delta \to \beta)/ns/C_{\max}$ with $N$, we first transform it into an instance of $F(\delta \to \beta)/ns, \preceq /C_{\max}$ with $N'$. *Algorithm NS* is used to find an optimal solution for $F(\delta \to \beta)/ns, \preceq /C_{\max}$, which is a lower bound for $F(\delta \to \beta)/ns/C_{\max}$ and can be used to evaluate the quality of the heuristic solutions.

From the results presented in the previous section, we know that given a job sequence, we can find an optimal batching policy using *algorithm NS*. Therefore, our heuristic methods simply foucs on determining job sequences. We use $Z_{ns}^{Hi}$ to denote the makespan reported by sequencing rule $NS_{Hi}$, $i = 1, 2, 3$. $Z_{ns}^{H4}$ is defined as $\min\{Z_{ns}^{H1}, Z_{ns}^{H2}, Z_{ns}^{H3}\}$.

**Rule** $NS_{H1}$: Order the jobs by nondecreasing order of $p_i$.
**Rule** $NS_{H2}$: Order the jobs by non-increasing order of $q_i$.
**Rule** $NS_{H3}$: Order the jobs by Johnson's rule.

With a slight modification, we have another set of methods, $AS_{H1}$, $AS_{H2}$, $AS_{H3}$ and $AS_{H4}$, for the situations where anticipatory setups are allowed.

To evaluate the effectiveness of the heuristics, we conduct a series of computational experiments. Let $Z_{ns}^{*}$ and $Z_{as}^{*}$ denote the makespans of the schedules obtained from using *algorithm NS* and *algorithm AS* to solve $F(\delta \to \beta)/ns, \preceq /C_{\max}$ and $F(\delta \to \beta)/as, \preceq /C_{\max}$ with the job set $N'$, respectively. The error ratio of a heuristic is defined as

$$\frac{Z_x^y - Z_x^*}{Z_x^*} \times 100\%,$$

where $x \in \{ns, as\}$ and $y \in \{H1, H2, H3, H4\}$. Note that an optimal solution is encountered when the error ratio is zero.

Table 1
Numerical results for non-anticipatory setup with factor = 1.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 3.07 | 3.48 | 1.25 | 1.25 | 8.23 | 8.34 | 3.30 | 3.30 | 0 | 7 | 10 | 10 |
| 100 | 2.15 | 2.84 | 0.87 | 0.87 | 5.26 | 6.96 | 2.12 | 2.12 | 0 | 5 | 10 | 10 |
| 150 | 1.46 | 1.76 | 0.52 | 0.52 | 4.01 | 4.33 | 1.59 | 1.59 | 0 | 19 | 23 | 23 |
| 200 | 1.21 | 1.67 | 0.46 | 0.46 | 3.00 | 3.82 | 1.29 | 1.29 | 0 | 8 | 12 | 12 |
| 250 | 0.89 | 1.32 | 0.35 | 0.35 | 2.40 | 3.14 | 1.13 | 1.13 | 0 | 18 | 21 | 21 |
| 300 | 0.84 | 1.08 | 0.29 | 0.29 | 2.26 | 2.82 | 0.97 | 0.97 | 0 | 19 | 24 | 24 |
| 500 | 0.59 | 0.75 | 0.18 | 0.18 | 1.51 | 2.21 | 0.66 | 0.66 | 0 | 11 | 15 | 15 |

Table 2
Numerical results for non-anticipatory setup with factor = 2.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 4.13 | 5.14 | 2.19 | 2.19 | 9.10 | 9.32 | 5.10 | 5.10 | 1 | 4 | 6 | 6 |
| 100 | 2.60 | 3.47 | 1.23 | 1.23 | 7.59 | 7.69 | 3.44 | 3.44 | 0 | 6 | 10 | 10 |
| 150 | 2.34 | 3.05 | 1.04 | 1.04 | 5.94 | 6.48 | 2.51 | 2.51 | 0 | 7 | 7 | 8 |
| 200 | 1.88 | 2.65 | 0.88 | 0.88 | 4.58 | 5.40 | 2.38 | 2.38 | 0 | 5 | 9 | 9 |
| 250 | 1.62 | 2.04 | 0.64 | 0.64 | 4.01 | 4.70 | 1.77 | 1.77 | 1 | 12 | 13 | 13 |
| 300 | 1.49 | 0.99 | 0.61 | 0.61 | 3.62 | 4.07 | 1.59 | 1.59 | 0 | 6 | 7 | 7 |
| 500 | 1.03 | 1.35 | 0.38 | 0.38 | 2.57 | 2.98 | 1.12 | 1.12 | 0 | 4 | 6 | 6 |

Table 3
Numerical results for non-anticipatory setup with factor = 3.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 4.72 | 6.17 | 2.74 | 2.74 | 10.76 | 10.35 | 6.40 | 6.40 | 0 | 2 | 2 | 2 |
| 100 | 3.44 | 4.42 | 1.85 | 1.85 | 8.36 | 7.40 | 4.86 | 4.86 | 0 | 3 | 5 | 5 |
| 150 | 2.85 | 3.58 | 1.36 | 1.36 | 6.16 | 6.37 | 3.19 | 3.19 | 0 | 3 | 4 | 4 |
| 200 | 2.65 | 3.41 | 1.31 | 1.31 | 5.59 | 6.22 | 2.75 | 2.75 | 0 | 3 | 4 | 4 |
| 250 | 2.33 | 3.07 | 1.10 | 1.10 | 4.70 | 5.13 | 2.16 | 2.16 | 0 | 2 | 3 | 3 |
| 300 | 2.02 | 2.68 | 0.96 | 0.96 | 5.00 | 5.54 | 2.57 | 2.57 | 0 | 5 | 7 | 7 |
| 500 | 1.26 | 1.68 | 0.51 | 0.51 | 3.30 | 3.60 | 1.43 | 1.43 | 1 | 8 | 8 | 8 |

In the experiment setting, the processing times $p_i$ and $q_i$ are set to be uniformly distributed over the interval [0, 100]. The problem size $n$ takes values from {50, 100, 150, 200, 250, 300, 500}. To contrast the impacts the setup time might have, we randomly select $s$ from $[0, 100 \cdot factor]$, where *factor* is 1, 2 or 3. For each combination of $s$ and $n$, each proposed method runs through 100 input instances. We keep track of the average error ratio, the largest error ratio, and the number of optimal solutions found amongst the 100 instances. The results for non-anticipatory setups are shown in tables 1–3. Tables 4–6 contain the results for the situations where anticipatory setups are allowed.

Table 4
Numerical results for anticipatory setup with factor = 1.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 2.75 | 3.31 | 1.27 | 1.27 | 6.99 | 8.11 | 4.90 | 4.90 | 1 | 10 | 15 | 15 |
| 100 | 1.84 | 2.25 | 0.75 | 0.75 | 5.09 | 5.19 | 2.28 | 2.28 | 1 | 5 | 11 | 12 |
| 150 | 1.62 | 1.87 | 0.56 | 0.56 | 3.55 | 4.18 | 1.61 | 1.61 | 0 | 11 | 14 | 14 |
| 200 | 1.11 | 1.25 | 0.38 | 0.38 | 2.89 | 3.12 | 1.22 | 1.22 | 0 | 15 | 23 | 23 |
| 250 | 1.03 | 1.32 | 0.39 | 0.39 | 2.95 | 3.18 | 1.26 | 1.26 | 0 | 11 | 15 | 15 |
| 300 | 0.87 | 1.05 | 0.31 | 0.31 | 2.14 | 2.70 | 1.08 | 1.08 | 0 | 16 | 20 | 20 |
| 500 | 0.58 | 0.66 | 0.18 | 0.18 | 1.59 | 1.83 | 0.72 | 0.72 | 0 | 25 | 29 | 30 |

Table 5
Numerical results for anticipatory setup with factor = 2.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 4.83 | 4.12 | 2.04 | 2.04 | 9.58 | 8.40 | 6.69 | 6.69 | 2 | 4 | 10 | 10 |
| 100 | 2.88 | 3.33 | 1.36 | 1.36 | 6.30 | 6.97 | 3.61 | 3.61 | 0 | 5 | 8 | 8 |
| 150 | 1.97 | 2.22 | 0.84 | 0.84 | 5.10 | 4.56 | 2.33 | 2.33 | 0 | 6 | 7 | 7 |
| 200 | 1.88 | 2.22 | 0.77 | 0.77 | 5.00 | 4.37 | 2.23 | 2.23 | 0 | 2 | 5 | 5 |
| 250 | 1.65 | 2.13 | 0.71 | 0.71 | 4.03 | 3.93 | 1.74 | 1.74 | 0 | 5 | 9 | 9 |
| 300 | 1.56 | 1.77 | 0.57 | 0.57 | 3.46 | 3.88 | 1.59 | 1.59 | 0 | 6 | 7 | 7 |
| 500 | 0.98 | 1.20 | 0.36 | 0.36 | 2.39 | 2.67 | 1.24 | 1.24 | 0 | 7 | 7 | 7 |

Table 6
Numerical results for anticipatory setup with factor = 3.

| | Average error (%) | | | | Largest error (%) | | | | Number of optimal solutions | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ | $H1$ | $H2$ | $H3$ | $H4$ |
| 50 | 3.17 | 3.97 | 1.73 | 1.73 | 7.65 | 7.81 | 7.10 | 7.10 | 6 | 2 | 21 | 21 |
| 100 | 3.56 | 4.11 | 2.10 | 2.10 | 5.89 | 6.83 | 5.29 | 5.29 | 0 | 2 | 5 | 5 |
| 150 | 2.73 | 3.29 | 1.40 | 1.40 | 6.60 | 6.26 | 3.59 | 3.59 | 0 | 4 | 7 | 7 |
| 200 | 2.42 | 2.72 | 0.06 | 1.06 | 5.72 | 4.97 | 2.74 | 2.74 | 0 | 6 | 6 | 6 |
| 250 | 2.23 | 2.60 | 0.98 | 0.98 | 4.31 | 4.93 | 2.06 | 2.06 | 0 | 3 | 4 | 4 |
| 300 | 1.89 | 2.35 | 0.91 | 0.91 | 4.12 | 4.13 | 2.04 | 2.04 | 0 | 1 | 2 | 2 |
| 500 | 1.39 | 1.67 | 0.58 | 0.58 | 3.06 | 3.33 | 1.52 | 1.52 | 0 | 1 | 3 | 3 |

From the numerical results, we clearly see that Johnson's rule ($H3$) almost totally dominates the other two methods. From the point of view of error ratios, all algorithms have better performance when the setup time is relatively small, i.e., *factor* $= 1$. Furthermore, the error ratios decrease when the problem sizes grow. By and large, the solutions generated by the heuristic algorithms are close to optimal solutions. This observation also suggests that the solutions for the total order problem with the job set $N'$ provide nontrivial lower bounds for the original problem with the job set $N$. In other

words, we may easily spend $O(n^2)$ time to obtain near-optimal solutions for the problems $F(\delta \rightarrow \beta)/as/C_{\max}$ and $F(\delta \rightarrow \beta)/ns/C_{\max}$.

## 6. Conclusions

In this paper, we have shown that, by a polynomial-time reduction from 3-PARTITION, $F(\delta \rightarrow \beta)/as/C_{\max}$ and $F(\delta \rightarrow \beta)/ns/C_{\max}$ are strongly NP-hard. In other words, it is very unlikely that polynomial or pseudo-polynomial algorithms for these problems can be found. We have also identified two polynomially solvable problems, $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ and $F(\delta \rightarrow \beta)/as, \preceq /C_{\max}$, by giving an $O(n^2)$ solution algorithm for each problem. The solution for $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ (or $F(\delta \rightarrow \beta)/as, \preceq /C_{\max}$) with $N'$ is a lower bound for the original problem with $N$.

To derive approximation solutions, we have designed some heuristics for determining initial job sequences and conducted computational experiments to test their effectiveness. Using the lower bounds obtained from solving $F(\delta \rightarrow \beta)/ns, \preceq /C_{\max}$ and $F(\delta \rightarrow \beta)/as, \preceq /C_{\max}$, we evaluate the error ratios of the heuristic solutions. The results show that the error ratios are small. They not only establish the effectiveness of the proposed heuristics, but also reveal that the lower bounds are close to the optimal solutions.

An important implication of our study is that the transformation of a hard problem to an easy one may provide a viable approach to dealing with different variants of flowshop-type problems. Following this line of transformation, Lin and Wu (2002) has designed a lower bound for the development of branch-and-bound algorithms for the classical flowshop scheduling to minimize the total completion time. Their algorithms can solve several instances with up to 65 jobs, while the best algorithm known in the literature can solve instances with 35 jobs.

## Acknowledgments

The first author was partially supported by the National Science Council of the R.O.C. under contract number 90-2416-H-260-013-C. The second author was partially supported by The Hong Kong Polytechnic University under a grant from the *ASD in China Business Services*.

## References

Ahmadi, J.H., R.H. Ahmadi, S. Dasu, and C.S. Tang. (1992). "Batching and Scheduling on Batch and Discrete Processors." *Operations Research* 40, 750–763.

Albers, A. and P. Brucker. (1993). "The Complexity of One-Machine Batching Problem." *Discrete Applied Mathatmematics* 47, 87–107.

Allahverdi, A., J.N.D. Gupta, and T. Aldowaisan. (1999). "A Review of Scheduling Research Involving Setup Considerations." *Omega* 27, 219–239.

Cheng, T.C.E., Z.-L. Chen, M.Y. Kovalyov, and B.M.T. Lin. (1996). "Parallel-Machine Batching and Scheduling to Minimize Total Completion Time." *IIE Transactions* 28, 953–956.

Cheng, T.C.E., J.N.D. Gupta, and G. Wang. (2000). "A Review of Flowshop Scheduling Research with Setup Times." *Production and Operations Management* 9, 262–282.

Cheng, T.C.E., M.Y. Kovalyov, and B.M.T. Lin. (1997). "Single Machine Scheduling to Minimize Batch Delivery and Job Earliness Penalties." *SIAM Journal on Optimization* 7, 547–559.

Cheng, T.C.E., B.M.T. Lin, and A. Toker. (2000). "Makespan Minimization in the Two-Machine Flowshop Batch Scheduling Problem." *Naval Research Logistics* 47, 128–144.

Cheng, T.C.E. and G. Wang. (1998). "Batching and Scheduling to Minimize the Makespan in the Two-Machine Flowshop." *IIE Transactions* 30, 447–453.

Coffmann, E.G., A. Nozari, and M. Yannakakis. (1989). "Optimal Scheduling of Products with Two Subassemblies on a Single Machine." *Operations Research* 37, 426–436.

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

Glass, C.A., C.N. Potts, and A.V. Strusevich. (2001). "Scheduling Batches with Sequential Job Processing for Two-Machine Flow and Open Shops." *INFORMS Journal on Computing* 13, 120–137.

Johnson, S.M. (1954). "Optimal Two- and Three-Stage Production Schedules with Setup Times Included." *Naval Research Logistics Quarterly* 1, 61–67.

Lin, B.M.T. and J.M. Wu. (2002). "A New Lower Bound Scheme for the Two-Machine Flowshop Scheduling to Minimize the Total Completion Time." In *Electronic Proceedings of the Fourth Asia-Pacific Conference on Industrial Engineering and Management Systems*, Taipei, Taiwan, December 2002.

Naddef, D. and C. Santos. (1988). "One-Pass Batching Algorithms for the One-Machine Problem." *Discrete Applied Matthematics* 21, 133–146.

Potts, C.N. and M.K. Kovalyov. (2000). "Scheduling with Batching: A Review." *European Journal of Operational Research* 102, 228–249.