An approximation scheme for two-machine flowshop scheduling with setup times and an availability constraint

Xiuli Wang T. C. Edwin Cheng #

Department of Logistics
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong

Abstract

This paper studies the two-machine permutation flowshop scheduling problem with anticipatory setup times and an availability constraint imposed only on the first machine. The objective is to minimize the makespan. Under the assumption that interrupted jobs can resume their operations, we present a polynomial-time approximation scheme (PTAS) for this problem.

Keywords: Flowshop scheduling; approximation scheme; availability constraint

#Corresponding author.

1. Introduction

Machine scheduling problems with availability constraints have received increasing attention from researchers in the last decade. When scheduling jobs over a planning period, we need to take into consideration the unavailability of machines for processing, which arises due to such causes as scheduled preventive maintenance, prior assignment of some fixed jobs or overlapping with the previous planning period. Surveys of the latest research results on this subject have been given by Lee et al. [1], Sanlaville and Schmidt [2], and Schmidt [3].

Although the classical two-machine flowshop scheduling problem with the objective of minimizing the makespan is polynomial-time solvable, the problem even with an unavailable interval becomes NP-hard (see [4]). Under the assumption that the jobs are resumable, i.e., an unfinished job can continue after the machine becomes available again, Lee [4] proposed pseudo-polynomial dynamic programming algorithms to solve the problem optimally. He also developed two heuristics with a worst-case error bound of 3/2 and 4/3 for the cases where the unavailable interval is on machines 1 and 2, respectively. Cheng and Wang [5] developed an improved heuristic with a worst-case bound of 4/3 for the problem with an unavailable interval on machine 1. Breit [6] presented an improved heuristic with a worst-case bound of 5/4 for the problem with an unavailable interval on machine 2. Ng and Kovalyov [7] provided fully polynomial-time approximation schemes for the problems with an unavailable interval on machine 1 or 2. Under the no-wait processing environment, Cheng and Liu [8] developed a polynomial-time approximation scheme (PTAS) for the problem.

The above-mentioned scheduling models only consider job processing times; in other words, setup times are assumed to be included in processing times. However, in many industrial settings it is necessary to treat setup times as separated from processing times. For example, the production of steel tubes mainly consists of two stages. First, pre-heated pillared billets are made into tubes by a rolling machine, which can control the outer-diameter and inner-diameter of the tubes by assembling different sizes of machine-frames and mandrels. Each order has special requirements for the outer-diameter and inner-diameter of the tubes. So setting the types of machine-frame and mandrel must be performed before fabricating the tubes. Once all the tubes of an order have been produced, a chemical disposal operation to remove the phosphor on the surface of the tubes is performed. So an order is taken as a job. Then a chasing lathe is used to make screw threads on the two ends of a steel tube and on the inner wall of a steel-hoop. The chasing lathe needs to have its tools adjusted before working on tubes in different diameters. Making adjustments of the tools may be anticipatory. In order to reduce intermediate inventories, orders are in turn processed in two stages. As a heavy machine, the rolling machine needs periodic maintenance such as replacing worn-out parts and lubricating the axles once a month. In the floor shop, a production plan usually spans two weeks. Thus, there exists at most an unavailable interval on the first machine over a scheduling period. To the best of our knowledge, only Wang and Cheng [9] have considered the scheduling problem with separated setups and availability constraints. In their paper, they studied two-machine flowshop scheduling with anticipatory setup times and a resumable availability constraint imposed on only one of the machines. They presented two heuristics and showed that their worst-case error bounds are no larger than 5/3.

Motivated by the above example, we consider the two-machine flowshop scheduling problem with anticipatory setup times, where an availability constraint is imposed only on the first machine. A setup is performed on a machine before processing a job. The setup times are anticipatory, i.e., the setup for the second operation of any job on machine 2 can start before the completion of its first operation on machine 1 whenever there is some idle time on machine 2. We assume that the processing order of the jobs is the same on each machine. That is, we confine ourselves to finding solutions that are permutation schedules for the problem. We also assume that all the jobs and their setups are resumable. The objective is to minimize the makespan. It is evident from Lee [4] that our problem is NP-hard. It is very unlikely to develop an algorithm for solving the problem optimally in polynomial time. In this paper we propose a PTAS for the problem.

The rest of this paper is organized as follows. In the next section, we introduce the notation and some preliminaries and investigate some optimal properties of the problem. In Section 3 we give an algorithm for a class of special instances of the problem and prove that it can generate an optimal schedule. In Section 4 we develop a PTAS based on the algorithm in Section 3 for our problem. Some conclusions are given in the last section.

2. Notation and preliminaries

For the problem under consideration, we first introduce the following notation to be used in this paper.

$N = \{J_1, \cdots, J_n\}$: a set of n jobs;

M_1, M_2: machine 1 and machine 2;

$\Delta_1 = t_2 - t_1$: the length of the unavailable interval on M_1, where M_1 is unavailable from time t_1 to t_2;

s_i^1, s_i^2: setup times of J_i on M_1 and M_2, respectively;

a_i, b_i: processing times of J_i on M_1 and M_2, respectively;

X: the job set in which all the jobs are finished before the unavailable interval on M_1;

$\overline{X}$: a sequence of X;

Y: the job set in which all the jobs are finished after the unavailable interval on M_1;

$\overline{Y}$: a sequence of Y;

Φ: the empty set;

π: a permutation schedule;

$C_i(\pi)$: completion time of J_i on M_2 in schedule π;

$C(\pi)$: the makespan of schedule π;

C^*: the optimal makespan.

The classical two-machine permutation flowshop scheduling problem with setup times, denoted as $F2|permu, setup|C_{max}$, can be optimally solved by the Yoshida and

Hitomi rule (YHR) [10], which can be stated as follows:

In an optimal schedule, if

$$\min\{s_i^1 + a_i - s_i^2, b_j\} \le \min\{s_j^1 + a_j - s_j^2, b_i\} \quad (1)$$

holds, then job J_i should be sequenced before job J_j.

Adopting the notation introduced by Lee [1], we denote the problem under study as $F2|permu, setup, r\text{-}a(M_1)|C_{max}$, i.e., the makespan minimization problem in a two-machine permutation flowshop with setup times and a resumable availability constraint on M_1.

For the problem under study, we assume that all the parameters s_i^1, s_i^2, a_i, b_i of job J_i are positive numbers and $\Delta_1 > 0$. For a non-empty subset Q of N, we define $a(Q) = \sum_{J_i \in Q}(s_i^1 + a_i)$ and $b(Q) = \sum_{J_i \in Q}(s_i^2 + b_i)$. If $Q = \Phi$, we set $a(Q) = 0$ and $b(Q) = 0$. The special case of $a(N) \le t_1$ can be optimally solved by YHR. We therefore assume in the following that $a(N) > t_1$ in our problem.

In order to obtain a schedule, we may first divide N into two disjoint subsets X and Y, then sequence all the jobs in X and Y as $\overline{X}$ and $\overline{Y}$, respectively. Since $X \cup Y = N$ and $X \cap Y = \Phi$, $[\overline{X}, \overline{Y}]$ constitutes a schedule for $F2 \mid permu, setup, r - a(M_1) \mid C_{max}$, where for the first job in $\overline{Y}$, its setup and processing may start on M_1 before t_1, but the job must be finished after time t_2. For schedule $[\overline{X}, \overline{Y}]$, we have the following lemma.

Lemma 1. For the problem $F2 \mid permu, setup, r - a(M_1) \mid C_{max}$, there exists no idle time between the last job of $\overline{X}$ and the first job of $\overline{Y}$ on M_1 if schedule [$\overline{X}$, $\overline{Y}$] is an optimal schedule.

Proof. The result is trivial. □

For the problem $F2 \mid r - a(M_1) \mid C_{max}$, Lee [4] proved that the jobs before or after the unavailable interval should be sequenced by Johnson's rule to obtain an optimal schedule. For the problem $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$, we can obtain a similar result, which is stated as follows:

Lemma 2. There exists an optimal schedule for the problem $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$ in which all the jobs in *X* follow YHR, and so are the remaining jobs in *Y*.

Proof. Obviously, if the jobs in *X* are not sequenced by YHR, the schedule may be improved by reordering the jobs according to YHR (this case is the same as the classical counterpart of the problem).

Let π denote a schedule of $X \cup Y$ that satisfies the conditions of Lemma 2 and *C*(*X*) the completion time of *X* on machine 2 in π. Without loss of generality, we assume that $\min_{J_i \in Y}\{s_i^1 + a_i\} > t_1 - a(X)$. Suppose that the first and second jobs of *Y* in π are J_p and J_q, respectively. We will prove that YHR is optimal for these two jobs by the job swapping argument.

Let π' be a schedule obtained by only swapping the first two jobs of *Y* in π. Then, we have

$$C_p(\pi) = \max\{a(X) + \Delta_1 + s_p^1 + a_p, C(X) + s_p^2\} + b_p,$$

$$\begin{aligned} C_q(\pi) &= \max\{a(X) + \Delta_1 + s_p^1 + a_p + s_q^1 + a_q, C_p(\pi) + s_q^2\} + b_q \\ &= \max\{a(X) + \Delta_1 + s_p^1 + a_p + s_q^1 + a_q + b_q, C(X) + s_p^2 + b_p + s_q^2 + b_q, \\ &\quad a(X) + \Delta_1 + s_p^1 + a_p + b_p + s_q^2 + b_q\}; \end{aligned} \tag{2}$$

similarly, for schedule π', we have

$$\begin{aligned} C_p(\pi') = \max\{& a(X) + \Delta_1 + s_p^1 + a_p + s_q^1 + a_q + b_p, C(X) + s_q^2 + b_q + s_p^2 + b_p, \\ & a(X) + \Delta_1 + s_q^1 + a_q + b_q + s_p^2 + b_p\}. \end{aligned} \tag{3}$$

From (2) and (3), we can verify that $C_q(\pi) \le C_p(\pi')$ if (1) holds. For other adjacent job pairs of *Y* in π, it is obvious that (1) holds in an optimal schedule (this case is also the same as the classical counterpart of the problem). □

We define (Q_1, Q_2) as a partition of set Q if $Q_1 \cup Q_2 = Q$ and $Q_1 \cap Q_2 = \Phi$ hold. For a given problem, once N is partitioned into two disjoint subsets X and Y, Lemma 2 shows that the jobs in X and Y are respectively sequenced by YHR in order to find an optimal solution. However, there are at most $2^n - 1$ partitions of N even if under the restriction of $a(N) > t_1$. So, we should focus on identifying some optimal properties to reduce the number of partitions.

An instance of our problem can be defined by a given set of numbers: $\{ (s_i^1, a_i, s_i^2, b_i), (t_1, t_2) \mid J_i \in N\}$. A lower bound LB for the optimal objective C^* is given as follows:

$$C^* \geq LB = \frac{(a(N) + \Delta_1) + b(N)}{2}.$$

Using the above lower bound LB, for any $\varepsilon > 0$, we may define the following subsets of N.

$$U = \{J_i \mid J_i \in N, \max\{s_i^1 + a_i, s_i^2 + b_i\} \geq \varepsilon LB\}, \quad (4)$$

$$V = N \setminus U = \{J_i \mid J_i \in N, \max\{s_i^1 + a_i, s_i^2 + b_i\} < \varepsilon LB\},$$

$$V_1 = \{J_i \mid J_i \in V, \ s_i^1 + a_i \leq s_i^2 + b_i\}, \quad (5)$$

$$V_2 = \{J_i \mid J_i \in V, s_i^1 + a_i > s_i^2 + b_i\}. \quad (6)$$

According to the above definitions, we call the jobs in U and V as large jobs and small jobs, respectively. If $\Delta_1 > \varepsilon LB$, then $|U| \leq \left\lfloor \frac{2}{\varepsilon} - 1 \right\rfloor$, where $\lfloor x \rfloor$ denotes the largest integer that is no larger than x; otherwise, we have

$$\frac{a(U) + b(U)}{2} + \frac{\Delta_1}{2} > \frac{2 - \varepsilon}{\varepsilon} \frac{\varepsilon LB}{2} + \frac{\varepsilon LB}{2} = LB = \frac{(a(U) + \Delta_1) + b(U)}{2},$$

a contradiction.

For our problem with the instance set Π given by

$\{ (s_i^1, a_i, s_i^2, b_i), (t_1, t_2) \mid J_i \in N\}$, we construct problem P_0 with the instance set Π_0, where $\Pi_0 = \Psi(\Pi)$, and Ψ is a map from $\{ (s_i^1, a_i, s_i^2, b_i), (t_1, t_2) \mid J_i \in N\}$ to $\{ (\bar{s}_i^1, \bar{a}_i, \bar{s}_i^2, \bar{b}_i), (t_1, t_2) \mid J_i \in N\}$, defined by

$$\begin{cases} \bar{s}_i^1 = s_i^1, \bar{a}_i = a_i, \bar{s}_i^2 = s_i^2, \bar{b}_i = b_i, & \text{if } J_i \in U \\ \bar{s}_i^1 = s_i^1, \bar{a}_i = a_i, \bar{s}_i^2 = s_i^1 + a_i, \bar{b}_i = (s_i^2 + b_2) - \bar{s}_i^2, & \text{if } J_i \in V_1 \\ \bar{s}_i^1 = s_i^1, \bar{a}_i = a_i, \bar{s}_i^2 = s_i^2 + b_i - \varepsilon_0, \bar{b}_i = \varepsilon_0, & \text{if } J_i \in V_2 \end{cases},$$

where ε_0 is a very tiny positive number.

Obviously, we have $\Pi_0 \subset \Pi$. For the problem P_0, we assume that a job J_i in V_1 can be split into two jobs J_{i1} and J_{i2} in Π_0, such that $\bar{s}_{i1}^1 + \bar{s}_{i2}^1 = \bar{s}_i^1$, $\bar{a}_{i1} + \bar{a}_{i2} = \bar{a}_i$, $\bar{s}_{i1}^2 + \bar{s}_{i2}^2 = \bar{s}_i^2$, $\bar{b}_{i1} + \bar{b}_{i2} = \bar{b}_i$ and $(\bar{s}_{i1}^2 + \bar{b}_{i1}) / (\bar{s}_{i1}^1 + \bar{a}_{i1}) = (\bar{s}_{i2}^2 + \bar{b}_{i2}) / (\bar{s}_{i2}^1 + \bar{a}_{i2})$. In the following, we will develop an optimal solution scheme for problem P_0.

3. An exact algorithm for P_0

Since $\Pi_0 \subset \Pi$, for the sake of convenience, we may denote an instance $\{ (\bar{s}_i^1, \bar{a}_i, \bar{s}_i^2, \bar{b}_i), (t_1, t_2) \mid J_i \in N\}$ of problem P_0 as $\{ (s_i^1, a_i, s_i^2, b_i), (t_1, t_2) \mid J_i \in N\}$. Thus, a partition ($U$, V_1, V_2) of N as defined by (4), (5) and (6) is also a partition for problem P_0.

We define $W(\sigma_1)$ as an ordered set in which all the jobs are consecutively scheduled by YHR, and $W(\sigma_2)$ as an ordered set in which all the jobs are consecutively scheduled in nonincreasing order of $(s_i^2 + b_i)/(s_i^1 + a_i)$. We call a schedule an effective schedule if it could end up as an optimal schedule.

For an instance of problem P_0 and given $\varepsilon > 0$, an exact algorithm is performed as follows.

Algorithm H_0

Step 1. For $\varepsilon > 0$, determine sets U, V_1 and V_2 according to (4), (5) and (6). If $U = \Phi$, then generate an effective schedule $\pi_{\mathrm{H}_0} = [V_1(\sigma_2), V_2(\sigma_2)]$, where if a job J_i of V_1 is over the unavailable interval in π_{H_0}, then the split job J_{i1} must satisfy that $a(X \cup \{J_{i1}\}) = t_1$, and stop. Otherwise, go to the next step.

Step 2. If all the partitions of U are checked, then go to Step 5; otherwise, for each possible partition (U_1, U_2) of U that has not been checked, if $a(U_1) > t_1$, this partition should be discarded; otherwise, proceed to the next step.

Step 3. Divide U_1 into two subsets U_{11} and U_{12} such that $U_{11} = \{J_i \mid s_i^1 + a_i - s_i^2 \le 0, J_i \in U_1\}$ and $U_{12} = U_1 \setminus U_{11}$. Let $X = U_{11} \cup V_1 \cup U_{12}$ and $Y = U_2 \cup V_2$. Let $\overline{X} = [U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1)]$ and $\overline{Y} = [U_2(\sigma_1), V_2(\sigma_2)]$.

Step 4. According to the value of $a(X)$, proceed with one of the following two cases:

1) Case $a(X) \le t_1$

If $U_2 = \Phi$, then generate an effective schedule $[U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1), V_2(\sigma_2)]$; otherwise, if no idle time exists between the last job of $\overline{X}$ and the first job of $\overline{Y}$ on M_1, then generate an effective schedule $[\overline{X}, \overline{Y}]$; else, discard $[\overline{X}, \overline{Y}]$. Go to Step 2.

2) Case $a(X) > t_1$

Divide U_2 into two subsets U_{21} and U_{22} such that $U_{21} = \{J_i \mid s_i^1 + a_i - s_i^2 \le 0, J_i \in U_2\}$ and $U_{22} = U_2 \setminus U_{21}$. Repeatedly remove the last job of $V_1(\sigma_2)$ until $a(X') = t_1$, where $X' = U_1 \cup V_{11}$, V_{12} and V_{11} denote the sets of the removed jobs and the remaining jobs, respectively. And the last job of $V_{11}(\sigma_2)$

and the first job of $V_{12}(\sigma_2)$ may be two split jobs J_{i1} and J_{i2} of J_i. Let $\bar{X}' = [U_{11}(\sigma_1), V_{11}(\sigma_2),\ U_{12}(\sigma_1)]$ and $\bar{Y}' = [U_{21}(\sigma_1), V_{12}(\sigma_2), U_{22}(\sigma_1),\ \ V_2(\sigma_2)]$. Then produce an effective schedule $[\bar{X}', \bar{Y}']$. Go to Step 2.

Step 5. Choose the schedule with the minimum makespan from all the generated effective schedules and denote it as π_{H_0}. Stop.

For any given $\varepsilon > 0$, there are at most $2^{\lfloor 2/\varepsilon - 1 \rfloor}$ partitions of set *U*. For each partition, the computational time to generate a schedule is no more than *O*(*n*log*n*). Hence, the complexity of Algorithm H_0 is $O(2^{\lfloor 2/\varepsilon - 1 \rfloor} n \log n)$.

A critical job is defined as the last job whose finishing time on M_1 is equal to its starting processing time on M_2 in a schedule. From Algorithm H_0, we have the following optimal properties for problem P_0.

Lemma 3. For problem P_0, if $U = \Phi$ in Algorithm H_0, then $\pi_{H_0} = [V_1(\sigma_2), V_2(\sigma_2)]$ is an optimal schedule.

Proof. For schedule $\pi_{H_0} = [V_1(\sigma_2), V_2(\sigma_2)]$, if no idle time exists on M_2, then $C(\pi_{H_0})$ = *b*(*N*), so π_{H_0} is an optimal schedule. Otherwise, if the critical job $J_k \in V_2$, denote $\hat{V}_2$ as the set of all the jobs sequenced after J_k in $V_2(\sigma_2)$, then since ε_0 is a very tiny positive number, we have

$$C(\pi_{H_0}) \le a(V_1) + \Delta_1 + \sum\nolimits_{J_i \in V_2 \setminus \hat{V}_2} (s_i^1 + a_i) + \sum\nolimits_{J_i \in \hat{V}_2} (s_i^2 + \varepsilon_0) + \varepsilon_0$$

$$\le a(V_1) + \Delta_1 + a(V_2) + \varepsilon_0 \le C^*.$$

In this situation, $C(\pi_{H_0})$ is also a lower bound for the problem. Hence π_{H_0} is optimal.

If the critical job $J_k \in V_1$, since $s_i^1 + a_i = s_i^2$ holds for all the jobs in V_1 and there exists idle time on M_2, then job J_k must be finished after t_2. Hence, the idle time $I(\sigma_2)$ on M_2 is

$$I(\sigma_2) = \Delta_1 + \sum_{J_i \in \hat{V}_1} [(s_i^1 + a_i) - (s_i^2 + b_i)] = \Delta_1 - \sum_{J_i \in \hat{V}_1} b_i \,,$$

where $\hat{V}_1$ denotes the set of all the jobs that are finished before t_1 on M_1. Since the jobs in $V_1(\sigma_2)$ are sequenced in nonincreasing order of $(s_i^2 + b_i)/(s_i^1 + a_i)$, the idle time $I(\sigma_2)$ is the smallest one among all the sequences of V_1.

When the critical job $J_k \in V_1$, if we insert a job J_j of V_2 before J_k, then the idle time on M_2 becomes $I'(\sigma_2)$, and we have

$$I'(\sigma_2) \geq \Delta_1 + \sum_{J_i \in \hat{V}_1} [(s_i^1 + a_i) - (s_i^2 + b_i)] + [(s_j^1 + a_j) - (s_j^2 + b_j)] > I(\sigma_2) \,.$$

If we insert a job J_j of V_2 in $V_1(\sigma_2)$ after J_k, since $s_j^1 + a_j > s_j^2$, then the makespan becomes C', and we have

$$C' \geq \Delta_1 + \sum_{J_i \in V_1 \setminus \hat{V}_1} (s_i^1 + a_i) + b_k + \sum_{J_i \in \hat{V}_1 \cup V_2} (s_i^2 + b_i) = C(\pi_{\mathrm{H}_0}) \,.$$

Hence, in this situation, π_{H_0} is an optimal schedule, too. □

Lemma 4. For problem P_0, if $U_2 = \Phi$ and $a(X) \leq t_1$ in Algorithm H_0, an optimal schedule is sequenced as $[U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1), V_2(\sigma_2)]$.

Proof. Denote schedule $[U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1), V_2(\sigma_2)]$ as π. If no idle time exists on M_2 or the critical job $J_k \in V_2$ in π, then, similar to the proof of Lemma 3, we can show that $C(\pi)$ is equal to a lower bound for π. Hence, π is an optimal schedule.

If there exists idle time on M_2 and the critical job $J_k \notin V_2$ in π, then the critical job only belongs to U_{12} because $s_h^1 + a_h \leq s_h^2$ holds for $J_h \in V_1 \cup U_{11}$. We have

$$C(\pi) = \sum_{J_i \in U_{11} \cup V_1 \cup (U_{12} \setminus \hat{U}_{12})} (s_i^1 + a_i) + b_k + \sum_{J_i \in \hat{U}_{12} \cup V_2} (s_i^2 + b_i) \,,$$

where $\hat{U}_{12}$ is a subset of U_{12}, whose jobs are sequenced after J_k.

According to Lemma 2, we only consider a schedule π' obtained by shifting a job $J_p \in V_1$ to the position where J_p is finished just after t_1. We have

$$C(\pi') = C(\pi) + (s_p^2 + b_p) - (s_p^1 + a_p) = C(\pi) + b_p > C(\pi).$$

So, in this situation, π is an optimal schedule. □

Lemma 5. For problem P_0, if $a(X) \le t_1$ and $U_2 \ne \Phi$ in Algorithm H$_0$, an optimal schedule is sequenced as $[\overline{X}, \overline{Y}]$, where $\overline{X} = [U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1)]$ and $\overline{Y} = [U_2(\sigma_1), V_2(\sigma_2)]$.

Proof. Once we have determined $V_1 \cup U_1 = X$ and $U_2 \cup V_2 = Y$, if $a(X) \le t_1$, an optimal schedule should be $[\overline{X}, \overline{Y}]$ according to Lemma 2. In the following, we will prove that it is necessary to have $V_1 \subset X$ and $V_2 \subset Y$ in an optimal schedule.

For schedule $\pi = [\overline{X}, \overline{Y}]$, suppose that there exists no idle time on M_2, or the critical job belongs to $V_2(\sigma_2)$. In either of these situations, similar to Lemma 3, we can easily prove that $[\overline{X}, \overline{Y}]$ is an optimal schedule.

In the situation that there exists idle time on M_2 and the critical job does not belong to V_2, since $s_i^1 + a_i - s_i^2 \le 0$ for $J_i \in U_{11}$ and $s_i^1 + a_i = s_i^2$ for $J_i \in V_1$, then the critical job $J_k \in U_{12}$ or U_2. So, we focus on analyzing the following two cases.

Case 1: Critical job $J_k \in U_{12}$

We denote all the jobs of U_{12} sequenced after J_k as set $\hat{U}_{12}$. Then the makespan of $[\overline{X}, \overline{Y}]$ is

$$C(\pi) = \sum_{J_i \in U_{11} \cup V_1 \cup (U_{12} \setminus \hat{U}_{12})} (s_i^1 + a_i) + b_k + \sum_{J_i \in \hat{U}_{12} \cup U_2 \cup V_2} (s_i^2 + b_i).$$

If we move job J_p of $V_1(\sigma_2)$ to the position between $U_{21}(\sigma_1)$ and $U_{22}(\sigma_1)$, where $U_{21} = \{J_i \mid s_i^1 + a_i - s_i^2 \le 0, J_i \in U_2\}$ and $U_{22} = U_2 \setminus U_{21}$, then we obtain a schedule $\pi' = [U_{11}(\sigma_1), V_1(\sigma_2) \setminus J_p, U_{12}(\sigma_1), U_{21}(\sigma_1), J_p, U_{22}(\sigma_1), V_2(\sigma_2)]$. Since $s_p^1 + a_p = s_p^2$, we have $C_k(\pi') = C_k(\pi) - (s_p^1 + a_p)$. Then

$$C(\pi') = C(\pi) + (s_p^2 + b_p) - (s_p^1 + a_p) = C(\pi) + b_p > C(\pi).$$

If we remove job J_q of $V_2(\sigma_2)$ and insert it after $U_{12}(\sigma_1)$, then we obtain a schedule $\pi'' = [U_{11}(\sigma_1), V_1(\sigma_2), U_{12}(\sigma_1), J_q, U_2(\sigma_1), V_2(\sigma_2) \setminus J_q]$. Since $s_q^1 + a_q > s_q^2$, shifting job J_q forward may result in idle time on *M*$_2$ just before job J_q in π''. Thus, $C(\pi'') \ge C(\pi)$.

Case 2: Critical job $J_k \in U_2$

We denote all the jobs of U_2 sequenced after J_k in π as set $\hat{U}_2$. Then the makespan of $[\overline{X}, \overline{Y}]$ is

$$C(\pi) = \Delta_1 + \sum_{J_i \in X \cup (U_2 \setminus \hat{U}_2)} (s_i^1 + a_i) + b_k + \sum_{J_i \in \hat{U}_2 \cup V_2} (s_i^2 + b_i).$$

If we shift job J_p of $V_1(\sigma_2)$ to the position between $U_{21}(\sigma_1)$ and $U_{22}(\sigma_1)$, then we obtain a schedule π'. When the shifted job J_p is processed after J_k, since $s_p^1 + a_p = s_p^2$, the completion time $C_k(\pi')$ is no less than $C_k(\pi) - (s_p^1 + a_p)$. Then $C(\pi') \ge C(\pi) + (s_p^2 + b_p) - (s_p^1 + a_p) = C(\pi) + b_p > C(\pi)$. When the shifted job J_p is processed before J_k, then $C_k(\pi') \ge C_k(\pi)$, so $C(\pi') \ge C(\pi)$.

If we remove job J_q of $V_2(\sigma_2)$ and insert it after $U_{12}(\sigma_1)$, then we obtain a schedule π''. We have $C_k(\pi'') = C_k(\pi) + (s_q^1 + a_q)$. We assume that the job sequenced immediately before J_q in π is J_{q-1}. Obviously, no idle time exists between jobs J_k and J_{q-1} in π. Because $s_i^2 + b_i < s_i^1 + a_i$ and $b_i = \varepsilon_0$ hold for each job in V_2, no idle time occurs between job J_{q-1} and the last job in π'' due to the removal of J_q. Therefore, we have

$$C(\pi'') = C(\pi) + (s_q^1 + a_q) - (s_q^2 + b_q) > C(\pi) .$$

Summarizing the above discussions, when $a(X) \le t_1$, shifting the jobs of V_1 backward after the unavailable interval or shifting the jobs of V_2 forward before the unavailable interval will not result in a schedule with smaller makespan. Thus, we have reached the conclusion. □

Lemma 6. For problem P_0, if $a(X) > t_1$ in Algorithm H$_0$, an optimal schedule is sequenced as $[\overline{X}, \overline{Y}]$, where $\overline{X} = [U_{11}(\sigma_1), V_{11}(\sigma_2), U_{12}(\sigma_1)]$ and $\overline{Y} = [U_{21}(\sigma_1), V_{12}(\sigma_2), U_{22}(\sigma_1), V_2(\sigma_2)]$.

Proof. Similar to Lemma 5, we can prove the conclusion. □

According to Lemma 3, Algorithm H$_0$ generates an optimal schedule when $U = \Phi$. When $U \ne \Phi$, Algorithm H$_0$ has enumerated all the possible partitions (U_1, U_2) of set *U*. From Lemmas 4 to 6, we also notice that Algorithm H$_0$ has included all the divisions of V_1 and V_2 in *X* and *Y* that may generate optimal schedules. According to Lemmas 1 to 2, an optimal schedule should be an effective schedule. Hence, we have derived the following theorem.

Theorem 1. For problem P_0, π_{H_0} is an optimal schedule.

4. A PTAS

In this section we will develop a polynomial-time approximation scheme for the problem $F2 \mid permu, setup, r - a(M_1) \mid C_{max}$. For an instance of the problem, we first construct a special instance by using a map Ψ, if necessary. Then, we use Algorithm

H_0 to generate an optimal schedule for this special instance. Finally, we make this schedule feasible for the original problem.

Algorithm H

Step 1. For a given $\varepsilon > 0$, if $\Delta_1 \leq \varepsilon\, LB$, sequence all the jobs of N by YHR, and denote the generated schedule as π_{H}, stop; otherwise, go to the next step.

Step 2. Construct problem P_0 from the given problem, and apply Algorithm H_0 to obtain an optimal schedule π_{H_0} for problem P_0.

Step 3. Starting from the last job of schedule π_{H_0}, shift right $\varepsilon\, LB$ time units for each operation in turn of all the jobs on M_2. If there exists a job J_i in V_1 that is split into two jobs J_{i1} and J_{i2} in π_{H_0}, use J_i instead of J_{i1} and discard job J_{i2}. Then, shift left the operations of the jobs on M_2 such that each job is processed as early as possible. The generated schedule is denoted as π_{H}. Stop.

It is clear that Step 1 requires at most $O(nlogn)$ time. The complexity of Algorithm H_0 is $O(2^{\lfloor 2/\varepsilon - 1 \rfloor} n \log n)$ in Step 2. Shifting jobs in Step 3 is performed in $O(n)$ time. Thus, the complexity of Algorithm H is $O(2^{\lfloor 2/\varepsilon - 1 \rfloor} n \log n)$.

Theorem 2. For the problem $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$, $C(\pi_{\mathrm{H}}) \leq (1+\varepsilon)C^*$.

Proof. For the problem $F2 \mid permu, setup \mid C_{max}$, YHR can generate an optimal schedule. For the problem $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$, supposing that a schedule π is generated by YHR, we can easily prove that $C(\pi) \leq C^* + \Delta_1$. So, if $\Delta_1 \leq \varepsilon\, LB$, then $C(\pi_{\mathrm{H}}) \leq C^* + \varepsilon\, LB \leq (1+\varepsilon)C^*$.

Because Algorithm H_0 can produce an optimal schedule for problem P_0 and Step 3 of Algorithm H shifts right at most $\varepsilon\ LB$ time units each operation of all the jobs on M_2, it is clear that $C(\pi_{\mathrm{H}}) \le (1+\varepsilon)C^*$. In the following, we show that the schedule generated by Algorithm H is a feasible schedule.

If no job is split in Algorithm H_0, from the definition of map Ψ, we notice that the differences between the problems $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$ and P_0 may be in the parameters of the small jobs. Without loss of generality, we assume that there exists no idle time between the setup and processing of a job on M_2 in schedule π_{H_0}. When problem P_0 reversely maps to the original problem, the overlapping processing time of job J_i on M_2 and M_1 in set V_1 is no more than $s_i^1 + a_i$, and the overlapping processing time of job J_j on M_2 and M_1 in set V_2 is less than b_j. Therefore, in Step 3 of Algorithm H, shifting right $\varepsilon\ LB$ time units each operation of all the job on M_2 will result in a feasible schedule for the problem.

If there exists a job J_i in V_1 that is split into two jobs J_{i1} and J_{i2}, when J_{i1} and J_{i2} are sequenced just before and after the unavailable interval for the case of $U = \Phi$, respectively, use J_i instead of J_{i1} and discard J_{i2}, then the completion time of J_i on M_2 is equal to the sum of the completion time of J_{i2} on M_2 and $\bar{b}_{i1}$. Since $(\bar{s}_{i1}^2 + \bar{s}_{i2}^2) + (\bar{b}_{i1} + \bar{b}_{i2}) = \bar{s}_i^2 + \bar{b}_i = s_i^2 + b_i \le \varepsilon LB$, moving right εLB time units the starting time for processing J_i on M_2 does not result in overlapping with the next job on M_2. For the other jobs, similar to the case of no split jobs, the feasibility of the schedule generated by Algorithm H can be proved. When J_{i2} is not sequenced just after the unavailable interval for the case of $a(X) > t_1$, similar to the above, we can also show that the schedule generated by Algorithm H is feasible. □

For any given $\varepsilon > 0$, the complexity of Algorithm H is $O(nlogn)$. Hence, Algorithm H is a PTAS for the problem $F2 \mid permu, setup, r-a(M_1) \mid C_{max}$.

5. Conclusions

In this paper we studied the two-machine permutation flowshop scheduling problem with anticipatory setup times and a resumable availability constraint imposed only on the first machine. We developed a polynomial-time approximation scheme for this problem.

Acknowledgments

We are grateful to an anonymous referee for his/her helpful comments on an earlier version of this paper. This research was supported in part by The Hong Kong Polytechnic University under a grant from the *Area of Strategic Development in China Business Services*.

References

[1] C.-Y. Lee, L. Lei, M. Pinedo, Current trends in deterministic scheduling, Annals of Operations Research, 70 (1997) 1-41.

[2] E. Sanlaville, G. Schmidt, Machine scheduling with availability constraints, Acta Informatica, 35 (1998) 795-811.

[3] G. Schmidt, Scheduling with limited availability, European Journal of Operational Research, 121 (2000) 1-15.

[4] C.-Y. Lee, Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint, Operations Research Letters, 20 (1997) 129-139.

[5] T.C.E. Cheng, G. Wang, An improved heuristic for two-machine flowshop scheduling with an availability constraint, Operations Research Letters, 26 (2000) 223-229.

[6] J. Breit, An improved approximation algorithm for two-machine flow shop scheduling with an availability constraint, Information Processing Letters, 90

(2004) 273-278.

[7] C.T. Ng, M.Y. Kovalyov, An FPTAS for scheduling a two-machine flowshop with one unavailability interval, Naval Research Logistics, 51 (2004) 307-315.

[8] T.C.E. Cheng, Z. Liu, Approximability of two-machine no-wait flowshop scheduling with availability constraints, Operations Research Letters, 31 (2003) 319-322.

[9] X. Wang, T.C.E. Cheng, Heuristics for two-machine flowshop scheduling with setup times and an availability constraint, Computers and Operations Research (2005), accepted.

[10] T. Yoshida, K. Hitomi, Optimal two-stage production scheduling with setup times separated, AIIE Transactions, 11 (1979) 261-263.