Approximation Schemes for Flow Shop Scheduling Problems with Machine Availabilty Constraints

Mikhail A. Kubzin
School of Computing and Mathematical Sciences,
University of Greenwich, London SE10 9LS, U.K.

Chris N. Potts
Faculty of Mathematical Studies, University of Southampton,
Southampton SO17 1BJ, U.K.

Vitaly A. Strusevich
School of Computing and Mathematical Sciences,
University of Greenwich, London SE10 9LS, U.K.

May 1, 2010

Abstract

This paper considers two-machine flow shop scheduling problems with machine availability constraints. When the processing of a job is interrupted by an unavailability period of a machine, we consider both the resumable scenario in which the processing can be resumed when the machine next becomes available, and the semi-resumable scenarios in which some proportion of the processing is repeated but the job is otherwise resumable. For the resumable scenario, problems with non-availability intervals on one of the machines are shown to admit fully polynomial-time approximation schemes that are based on an extended dynamic programming algorithm. For the problem with several non-availability intervals on the first machine, we present a fast 3/2-approximation algorithm. For the problem with one non-availability interval under the semi-resumable scenario, polynomial-time approximation schemes are developed.
(Production-Scheduling; Flow-Shop; Analysis of Algorithms; Computational Complexity; Suboptimal Algorithms)

This paper has appeared as a journal publication in:
Computers & Operations Research, Volume 36, Issue 2, pages 379-390, February 2009.
For information about the journal, see:
http://www.elsevier.com/wps/find/journaldescription.cws_home/300/description#description

1 Introduction

In real industrial settings, it is often necessary to address situations where the machines used for processing become unavailable during the planning period due to planned maintenance, machine breakdowns, or the use of such facilities for other activities that conflict with planning decisions. Relevant scheduling models with machine non-availability periods have recently been studied in numerous papers. For recent surveys of the related results, we refer to Sanlaville and Schmidt [16] and Schmidt [17].

This paper studies the two-machine flow shop scheduling problem under various assumptions about the jobs affected by a non-availability period, and the structure of non-availability intervals.

In the classical two-machine flow shop scheduling problem, we are given a set of n jobs and two machines. Each job has to be processed on the first machine and then on the second machine. We refer to the processing of a job on a machine as an *operation*. The processing times of all operations are known. Every job is to be processed on at most one machine at a time, and each machine processes no more than one job at a time. In the models studied in this paper, the machines are not continuously available for processing. Further, the precise time of each interval of machine unavailability is known in advance. In all problems discussed in this paper, the goal is to minimize the makespan, i.e., the maximum completion time of all jobs on all machines.

Henceforth, we often refer to a non-availability period on a machine as a *hole*. As introduced by Lee [14], we consider various scenarios relating to the processing of an operation that is interrupted by a hole.

Resumable Scenario. In this case, the total processing time of the operation interrupted by a non-availability interval remains equal to its original processing time, i.e., the processing of the operation is interrupted by the hole and is resumed when the machine next becomes available. For example, this scenario applies to a typist who has gone to a lunch break and then resumes the typing of a manuscript from the last typed page.

Semi-Resumable Scenario. This case is similar to the resumable scenario, but the portion of an operation performed before the hole has to be partially reprocessed after the hole. Thus, the total processing time of an operation becomes greater than its original processing time. This scenario is applicable if, for example, the machine must heat the job to the required temperature, so that the cooling of the job during the non-availability period necessitates re-heating to the temperature at the point of interruption.

Non-Resumable Scenario. For this scenario, after an interruption the total processing of the interrupted operation is equal to its original processing time, so effectively the operation restarts from scratch. We can regard this is a special case of the semi-resumable scenario when an operation has to be completely reprocessed. An example where such a situation arises is in situations related to downloading files from the Internet: if the connection is lost during the download, the process must be restarted when the connection is restored.

It is well-known that the two-machine flow shop problem with no availability constraints is solvable in $O(n \log n)$ time by an algorithm of Johnson [8]. However, the complexity of the problem changes if the machines are not continuously available. Lee (1997, 1999) studies the problem with a single hole for all three scenarios: resumable, non-resumable and semi-resumable. He proves that the problem is NP-hard for each of these scenarios, and presents pseudopolynomial dynamic programming algorithms. For the resumable scenario, Kubiak et al. [9] show that the problem with a variable number of holes on one of the machines is NP-hard in the strong sense. For the both the semi-resumable scenario and the non-resumable scenario, Lee [12] shows that minimizing the makespan on a single machine with a variable number of holes is NP-hard in the strong sense.

The complexity status of scheduling with machine unavailability has stimulated research on approximability of the problems. It appears that for the problem under consideration, as well as some other scheduling problems with machine availability constraints, there is a sharp borderline between those variants of the problem for which it is possible to design fast algorithms that provide a provably close approximation to the optimum, and those variants for which finding an approximate solution that is close enough to the optimum is

theoretically no easier than determining the optimum exactly.

A polynomial-time algorithm that creates a schedule with the makespan that is at most ρ times the optimal value (where $\rho \geq 1$), is called a ρ-approximation algorithm; the value of ρ is called a *worst-case ratio bound*. If a problem admits a ρ-approximation algorithm, then it is said to be *approximable within a factor* ρ. A family of ρ-approximation algorithms is called a *polynomial-time approximation scheme*, or a *PTAS*, if $\rho = 1 + \varepsilon$ for any fixed $\varepsilon > 0$ and the running time is polynomial in the length of the problem input. If additionally the running time of a PTAS is polynomial with respect to $1/\varepsilon$, it is called a *fully polynomial-time approximation scheme*, or an *FPTAS*.

The resumable two-machine flow shop problem becomes non-approximable within a fixed factor in polynomial time, unless $\mathcal{P} = \mathcal{NP}$, provided that there are either two holes on the second machine or one hole on each machine (Lee 1999, Kubiak et al. 002). Breit et al. (2001) show that for the semi-resumable scenario as well as for the non-resumable scenario, the single machine problem with two holes is non-approximable within a fixed factor, unless $\mathcal{P} = \mathcal{NP}$. This implies that the two-machine flow shop problem may admit a ρ-approximation algorithm for finite ratio ρ only if there is exactly one hole (for all scenarios), or if there are several holes on the first machine for the resumable scenario only.

The following approximation results for the two-machine flow shop problem with a single hole on one of the machines are known. For the resumable scenario, Lee [13] gives a (4/3)-approximation algorithm if the hole is on the second machine, and a (3/2)-approximation algorithm if the hole is on the first machine. For the former problem, Breit (2004) develops an improved (5/4)-approximation algorithm, while for the latter problem, Cheng and Wang [3] propose an improved (4/3)-approximation algorithm. Finally, it has been shown that the two-machine flow shop with a single hole under the resumable scenario admits a PTAS [1] and an FPTAS [**?**].

For the semi-resumable scenario (and also the non-resumable scenario), Lee [14] gives a (3/2)-approximation algorithm if the hole is on the second machine and a 2-approximation algorithm if the hole is on the first machine. If there are two holes, one on the first and the other on the second machine, and these holes are consecutive, i.e., the second hole starts exactly when the first hole ends, Cheng and Wang [2] give a (5/3)-approximation algorithm.

Our contribution ...

The remainder of this paper is organized as follows. Section 2 contains a formal description of the two-machine flow shop problem under consideration, and introduces some notation and terminology. In Section **??**, a dynamic programming algorithm for the resumable scenario and several holes on one of the machines is presented. Section **??** demonstrates how to use the available dynamic programming algorithms to create FPTAS's. Since the running time of these FPTAS's is fairly large, in Section **??** a fast heuristic algorithm with a worst-case ratio of 3/2 is presented for the problem with holes on the first machine and the semi-resumable scenario. Section 4 describes a PTAS for the problem with a single hole on one of the machines under the semi-resumable scenario. Some concluding remarks are given in Section 5.

2 Preliminaries

In this section, we formally describe the problem under consideration and introduce the required notation and terminology.

We are given a set of jobs $N = \{J_1, J_2, \ldots, J_n\}$. Each job J_j, for $j = 1, 2, \ldots, n$, consists of two operations O_{jA} and O_{jB} to be performed on machines A and B, respectively. We denote the processing time of operations O_{jA} and O_{jB} by a_j and b_j, respectively. We

assume that a_j and b_j are positive integers. We define $a_{\max} = \max_{1 \le j \le n} a_j$.

In the two-machine flow shop, each job is first processed on machine A and then on machine B. For a schedule S, we denote the value of the makespan by $C_{\max}(S)$. Following the notation of Lawler et al. (1993), this classical scheduling problem is usually denoted by $F2||C_{\max}$. Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be an arbitrary sequence of the job indices. In a schedule S associated with π, the jobs are processed on each machine according to π, and each operation starts as early as possible. It follows that

$$C_{\max}(S) = \max_{1 \le u \le n} \Big\{ \sum_{j=1}^{u} a_{\pi(j)} + \sum_{j=u}^{n} b_{\pi(j)} \Big\}. \tag{1}$$

A job $J_{\pi(u)}$ for which the maximum in the right-hand side of (1) is attained for index u is called *critical*. A critical job starts on machine B immediately after it is completed on machine A, and the starting times of either operation of a critical job cannot be delayed without increasing the makespan of the schedule under consideration.

It is well-known that problem $F2||C_{\max}$ is solvable in $O(n \log n)$ time due to Johnson's algorithm [8]. In this algorithm, an optimal sequence starts with the jobs for which $a_j \le b_j$ sorted in nondecreasing order of a_j, followed by the remaining jobs sorted in nonincreasing order of b_j. We refer to a sequence of jobs obtained by this rule as a *Johnson sequence*. The corresponding optimal schedule can be obtained by processing the jobs in a Johnson sequence on each machine, starting each operation as early as possible.

In this study, we are concerned with a modification of the classical problem $F2||C_{\max}$ in which there may be several non-availability intervals on each machine. Extending the standard notation, we denote the resulting problem by $F2|h(q_A, q_B), Sc|C_{\max}$, where q_A and q_B denote the number of holes on machines A and B, respectively, and $Sc \in \{Re, S\text{-}Re, N\text{-}Re\}$ denotes the scenario under consideration. Here, "Re" corresponds to the resumable scenario, "S-Re" denotes the semi-resumable scenario, and "N-Re" denotes the non-resumable scenario. The goal is to minimize the maximum completion time $C_{\max}$.

We now explain in more detail the three scenarios. Our description is given for the case that a hole occurs on machine A; the case of a hole on machine B is analogous. Consider a hole $[s, t]$ on machine A. Suppose that J_k is a job for which operation O_{kA} starts before time s but cannot be completed by time s. In what follows, we refer to J_k as a *crossover* job. The three scenarios that we study differ from each other in the way that crossover jobs are treated. Assume that a crossover job J_k is processed on machine A for x_k time units before the hole $[s, t]$. Under the resumable scenario, the processing of operation O_{kA} is interrupted at time s and is resumed at time t requiring a further $a_k - x_k$ time units of processing. Under the semi-resumable scenario, the crossover job J_k resumes at time t, and requires processing for a further $a_k - x_k + \alpha x_k$ time units, where α is a given constant and $0 \le \alpha \le 1$. The case $\alpha = 0$ corresponds to the resumable scenario, and the case $\alpha = 1$ to the non-resumable scenario.

If there is only one hole in the problem under consideration, then let the unavailability interval be $[s, t]$, and let $\Delta = t - s$ be the length of this interval. Otherwise, we denote the unavailability intervals by $[s_i, t_i]$ for $i = 1, 2, \ldots, q$, where q denotes the number of holes, and the lengths of the unavailability intervals by $\Delta_i = t_i - s_i$ for $i = 1, 2, \ldots, q$, respectively. We assume that s, t, s_i and t_i are non-negative integers.

This paper is mainly concerned with problem $F2|h(q_A, q_B), Sc|C_{\max}$ in which either $q_A = 0$ or $q_B = 0$, i.e., the holes appear on one machine only. It can be seen that for problems of this type, there exists an optimal schedule in which both machines process the jobs according to the same sequence. Additionally, the search for an optimal schedule can be restricted to a class of schedules that contains a critical job. As in the classical case, a

critical job starts on machine B immediately after it is completed on machine A and cannot be delayed. The makespan of a schedule that contains a critical job J_u is determined by the length of the *critical path*, which is the sum of the following components:

(i) the processing time of both operations of J_u;

(ii) the total processing time of all jobs that precede J_u on machine A;

(iii) total processing time of all jobs that follow J_u on machine B;

(iv) total length of all holes either before J_u on machine A or after J_u on machine B.

We exclude from further consideration the situation that all jobs complete before the first hole since in this case an optimal schedule can be found by Johnson's algorithm.

For any non-empty set $Q \subseteq N$, we define

$$a(Q) = \sum_{j \in Q} a_j, \quad b(Q) = \sum_{j \in Q} b_j,$$

and denote $a(\emptyset) = b(\emptyset) = 0$.

3 Resumable Scenario

In this section, we develop a fast heuristic algorithm that guarantees a solution fairly close to the optimum. For this problem with only one hole on machine A, Lee (1997) presents a $(3/2)-$approximation algorithm. We extend his heuristic to the case of several holes and simplify both the algorithm and the proof. Note that, to date, no approximation algorithm for the flow shop problems with more than one hole has been developed.

Our algorithm creates two schedules and outputs the better of them as a heuristic solution.

Algorithm H

1. Select a job with the largest processing time on machine B and place it into the first position in the processing sequence, followed by all other jobs in an arbitrary order. Let S_1 denote the schedule associated with that sequence.

2. Sequence all jobs j in non-increasing order of b_j/a_j and let S_2 denote the schedule associated with that sequence.

3. If $C_{\max}(S_1) \leq C_{\max}(S_2)$, then set $S_H = S_1$ as the heuristic schedule; otherwise, set $S_H = S_2$.

The running time of Algorithm H is $O(n \log n)$. Let S^* denote an optimal schedule. Our worst-case analysis of Heuristic H uses the following result.

Lemma 1 *If J_k is a critical job in schedule S_2 for problem $F2|h(q,0), Re|C_{\max}$, then*

$$C_{\max}(S_2) - C_{\max}(S^*) \leq b_k.$$

This lemma is proved by Lee [13] for the problem with a single hole. However, it is straightforward to extend his argument to the case of several holes. We now proceed to the main result in this section.

Theorem 1 *For the schedule S_H obtained by Algorithm H for problem $F2|h(q,0), Re|C_{\max}$,*

$$C_{\max}(S_H)/C_{\max}(S^*) \leq 3/2 \tag{2}$$

and this bound is tight.

Proof. It is sufficient to consider the case that $a(N) > s_1$; otherwise, Johnson's algorithm delivers an optimal solution in the polynomial time.

First, suppose that there is a job with processing time on machine B that is greater than $C_{\max}(S^*)/2$. Following a suitable relabelling of jobs, we may assume that this job is J_1. Since $b_1 > \frac{1}{2}C_{\max}(S^*)$, we observe that

$$\sum_{j=2}^{n} b_j < C_{\max}(S^*)/2. \tag{3}$$

In schedule S_1 found in Step 1 of Algorithm H, job J_1 is processed first. There are two separate cases depending on the position of the critical job in this schedule.

(i) Job J_1 is critical. Since J_1 is scheduled first, its completion time on machine B is a lower bound on the makespan of an optimal schedule, so that $C_{\max}(S_1) \leq C_{\max}(S^*) + \sum_{j=2}^{n} b_j$. Due to (3), we derive that (2) holds for $S_H = S_1$.

(ii) Job J_1 is not critical. Since the completion time of the critical job on machine A is a lower bound on the optimal makespan, we again obtain $C_{\max}(S_1) \leq C_{\max}(S^*) + \sum_{j=2}^{n} b_j$, so that the theorem holds.

Second, consider the remaining case that $b_j < C_{\max}(S^*)/2$ for all $j = 1, 2, \ldots, n$. We apply Lemma 1 to obtain $C_{\max}(S_2) \leq C_{\max}(S^*) + b_k$, which in turn implies that (2) holds. for $S_h = S_2$.

To establish that the bound (2) is tight, consider the following instance of problem $F2|h(q,0), Re|C_{\max}$. There are two jobs such that $a_1 = k+1, b_1 = k^2+3k+2$ and $a_2 = k, b_2 = k^2+k+1$, where k is an integer greater than 1. The hole on machine A occupies the interval $[k, k^2+k]$. Since $b_1 = k^2+3k+2 > k^2+k+1 = b_2$, it follows that in Step 1 of the algorithm we obtain schedule S_1 associated with the sequence (J_1, J_2). Since for $k > 1$ we have that

$$\frac{b_1}{a_1} = \frac{k^2+3k+2}{k+1} = k+2 > \frac{k^2+k+1}{k} = \frac{b_2}{a_2},$$

it follows that in Step 2 we obtain schedule S_2 associated with the same sequence. It is easy to verify that $C_{\max}(S_H) = 3k^2+5k+4$. On the other hand, for the optimal schedule S^* the sequence of jobs is (J_2, J_1), so that we have $C_{\max}(S^*) = 2k^2+5k+3$. Thus, as k approaches to infinity $C(S_H)/C(S^*)$ goes to 3/2. ∎

4 Semi-Resumable Scenario: PTAS

In this section, we consider the two-machine flow shop problem with a single hole on machine B, under the semi-resumable scenario, i.e., problem $F2|h(0,1), S\text{-}Re|C_{\max}$. The case of the hole on machine A is symmetric. Recall that under the semi-resumable scenario, the operation of the crossover job has to be partially reprocessed, and $\alpha \in [0,1]$ is a given parameter that determines the proportion of reprocessing that is required. We restrict our attention to instances in which some job completes after the hole so that $C_{\max}(S^*) > t$;

otherwise, the problem is trivially solved by sequencing the jobs according to Johnson's rule.

For each problem $F2|h(0,1), S\text{-}Re|C_{\max}$ and $F2|h(1,0), S\text{-}Re|C_{\max}$, we problem a PTAS. The best known results in this area available to date are a (3/2)-approximation algorithm for problem $F2|h(0,1), S\text{-}Re|C_{\max}$ and a 2-approximation algorithm for problem $F2|h(1,0), S\text{-}Re|C_{\max}$ [14].

Our PTAS has the following features. First, we follow the useful idea of Sevastianov and Woeginger (1998) of splitting the jobs into big, medium and small categories. We look for an approximate solution in one of three classes of schedules, depending on the position and the size of the crossover job. For each of these classes, we enumerate all schedules of the big jobs and attempt to schedule the small jobs in the gaps of that schedule by solving the linear programming relaxation of in integer programm. Those small jobs that cannot be fully processed in the existing gaps, plus all medium jobs, are appended.

We first specify how the big, medium and small jobs are defined. Let $T = a(N) + b(N)$. Consider any given ε, where $0 < \varepsilon < 1$. We define $\tilde{\varepsilon} = \varepsilon/10$, and introduce the sequence of real numbers $\delta_1, \delta_2, \ldots$, where $\delta_t = \tilde{\varepsilon}^{2^t}$. For each integer t, where $t \geq 1$, consider the set of jobs

$$N^t = \{J_j | j \in N,\ \delta_t^2 T < a_j + b_j \leq \delta_t T\}.$$

Note that the sets of jobs $N^1, N^2, \ldots$ are mutually disjoint. Thus, there exists a value $\tau \in \{1, \ldots, \lceil 1/\tilde{\varepsilon} \rceil\}$ such that $a(N^\tau) + a(N^\tau) \leq \tilde{\varepsilon} T$ holds; otherwise, $T \geq a(N^1) + b(N^1) + \cdots + a(N^{\lceil 1/\tilde{\varepsilon} \rceil}) + b(N^{\lceil 1/\tilde{\varepsilon} \rceil}) > \lceil 1/\tilde{\varepsilon} \rceil \tilde{\varepsilon} T$, which is impossible. We define $\delta = \delta_\tau$, and note that

$$\tilde{\varepsilon}^{2^{\lceil 1/\tilde{\varepsilon} \rceil}} \leq \delta \leq \tilde{\varepsilon}^2.$$

We now partition the jobs into *big jobs* W_b, *medium jobs* W_m and *small jobs* W_s by partitioning the index set N as follows:

$$\begin{aligned} W_b &= \{j | a_j + b_j > \delta T\}, \\ W_m &= \{j | \delta^2 T < a_j + b_j \leq \delta T\}, \\ W_s &= \{j | a_j + b_j \leq \delta^2 T\}. \end{aligned} \tag{4}$$

Note that, by definition, $W_m = N^\tau$, which implies

$$a(W_m) + b(W_m) \leq \tilde{\varepsilon} T. \tag{5}$$

Let n_b denote the number of big jobs. From the definition of W_b, each big job has a total processing time that exceeds δT. Since the total processing time of all jobs is equal to T, we deduce that $n_b < 1/\delta$. Moreover, $1/\delta \leq \tilde{\varepsilon}^{-2^{\lceil 1/\tilde{\varepsilon} \rceil}}$, which implies that n_b is fixed.

Our approximation scheme involves searching for an approximate solution in several specific classes of schedules. For notational convenience, we denote the big jobs by J'_k for $k = 1, \ldots, n_b$, and their processing times on machines A and B by a'_k and b'_k, respectively. Define $\mathcal{S}_0$, and $\bar{\mathcal{S}}_v$ and $\tilde{\mathcal{S}}_v$ for $v = 1, \ldots, n_b$ as follows:

$\mathcal{S}_0$ – the class of schedules in which all big jobs are completed before the hole;

$\bar{\mathcal{S}}_v$ – the class of schedules in which no big job is interrupted by the hole and a big job J'_v is the first big job that starts on B after the hole;

$\tilde{\mathcal{S}}_v$ – the class of schedules in which a big job J'_v is interrupted by the hole.

It is clear that an optimal schedule S^* belongs to one of these classes.

We introduce two dummy jobs J'_0 and J'_{n_b+1}, where J'_0 and J'_{n_b+1} each have a zero processing time on both machines. These dummy jobs are need to simplify the statement of an linear programming problem that forms a component of our algorithm.

Case 1

We first design an approximate solution in class $\mathcal{S}_0$. In a schedule of this class, the crossover job is not a big job. Further the jobs before the hole including all big jobs are sequenced by Johnson's rule. If necessary, renumber the big jobs so that a Johnson sequence of these jobs is given by $(J'_1, \ldots, J'_{n_b})$. After placing the two dummy jobs at the beginning and at the end, we obtain the sequence $(J'_0 J'_1, \ldots, J'_{n_b}, J'_{n_b+1})$.

Let n_s denote the number of small jobs and denote the small jobs by $J_1, \ldots, J_{n_s}$. We define variables

$$x_{jk} = \begin{cases} 1, & \text{if } J_j \text{ is scheduled between jobs } J'_{k-1} \text{ and } J'_k, \\ 0, & \text{otherwise,} \end{cases}$$

for $j \in W_s$ and $k = 1, \ldots, n_b + 1$. The following integer program is a relaxation of the problem of finding a schedule from class $\mathcal{S}_0$ for processing the big and the small jobs. The variable C provides a lower bound on the makespan of that partial schedule. We call this integer program IP(0).

Minimize C

subject to

$$\sum_{k=1}^{u}\Big(\sum_{j\in W_s} a_j x_{jk} + a'_k\Big) + b'_u + \sum_{k=u+1}^{n_b+1}\Big(\sum_{j\in W_s} b_j x_{jk} + b'_k\Big) \le C, \qquad u = 1, \ldots, n_b + 1; \tag{6}$$

$$\sum_{k=1}^{u}\Big(\sum_{j\in W_s} a_j x_{jk} + a'_k\Big) + b'_u + \sum_{k=u+1}^{n_b}\Big(\sum_{j\in W_s} b_j x_{jk} + b'_k\Big) \le s, \qquad u = 1, \ldots, n_b; \tag{7}$$

$$\sum_{k=1}^{n_b+1} x_{jk} = 1, \qquad j \in W_s; \tag{8}$$

$$x_{jk} \in \{0, 1\}, \qquad j \in W_s,\ k = 1, \ldots, n_b + 1. \tag{9}$$

Constraints (6) give lower bounds on the makespan, provided that big job J'_u is critical. Constraints (7) imply that all big jobs in the partial schedule must be completed on machine B before the hole. Constraints (8) ensure that each small job must be sequenced between some pair of big jobs, including the dummy jobs.

We solve the linear programming relaxation of this problem in which the constraints $x_{ij} \in \{0, 1\}$ in (9) are replaced by the non-negativity constraints $x_{ij} \ge 0$. Any small job J_j for which $x_{jk} \ne 1$ for any position k in this solution is called a *fractional* job. Note that, excluding non-negativity, there are $2n_b + n_s + 1$ constraints, and consequently $2n_b + n_s + 1$ basic variables, including C which must be basic. Moreover, each of the n_s assignment constraints (8) contains a distinct set of variables. Following the same type of analysis as that of Potts (1985), we establish that there are at most $2n_b$ fractional jobs.

Replace each fractional small job J_j with several pseudo-jobs J_j^k for all k such that $x_{jk} > 0$. A pseudo-job J_j^k is assigned to a position between jobs J'_{k-1} and J'_k, and its processing times on machines A and B are set equal to $a_j^k = a_j x_{jk}$ and $b_j^k = b_j x_{jk}$. Each non-fractional small job J_j with $x_{jk} = 1$ is assigned to a position between jobs J'_{k-1} and J'_k.

For $k = 1, \ldots, n_b + 1$, The small non-fractional jobs and pseudo-jobs assigned to positions between jobs J'_{k-1} and J'_k are sequenced according to Johnson's rule, for $k = 1, \ldots, n_b + 1$,

Let S^0_{LP} be a schedule associated with the job sequence constructed as described above, with any crossover job receiving the appropriate proportion of reprocessing. Remove all pseudo-jobs and assign all fractional small jobs and all medium jobs to be processed in an arbitrary order so that the first of these jobs starts on machine A at time $\max\{C_{\max}(S^0_{LP}), t\}$. Let S_ε denote resulting schedule.

We prove next that

$$C_{\max}(S_\varepsilon) \leq C_{\max}(S^*) + (2n_b + 2)\delta^2 T + \tilde{\varepsilon} T, \tag{10}$$

assuming that S^* is an optimal schedule that also belongs to class $\mathcal{S}_0$. Let C^0 denote an optimal value of C in the linear programming relaxation of the integer program $IP(0)$. Let J_r be a crossover job in schedule S^0_{LP}; recall that in this case J_r is either a small job or a pseudo-job. Recall that according to the semi-resumable scenario job J_r will be reprocessed on machine B for no more than αb_r extra time units.

If in schedule S^0_{LP} a big job J'_u is critical, then the value of $C_{\max}(S^0_{LP})$ exceeds that of C^0 by no more than αb_r. Thus, $C_{\max}(S^0_{LP}) \leq C^0 + \delta^2 T \leq C_{\max}(S^*) + \delta^2 T$.

Suppose that in schedule S^0_{LP} the critical job belongs to the set W_k of small jobs and pseudo-jobs positioned between the big jobs J'_{k-1} and J'_k for some $k, 1 \leq k \leq n_b + 1$. Without loss of generality, we may assume that the critical job us a non-fractional small job J_w; the case of a pseudo-job being critical is analogous. Consider the contribution to the value of $C_{\max}(S^0_{LP})$ that is delivered by the jobs in W_s. Job J_w contributes $a_w + b_w$, a job J_j (or pseudo-job J^k_j) that precedes J_w contributes $a_j x_{jk}$, while a job J_j (or pseudo-job J^k_j) that follows J_w contributes $b_j x_{jk}$. If $a_w \leq b_w$ then according to Johnson's rule we have that $a_j x_{jk} \leq b_j x_{jk}$ for all jobs of W_s that precede J_w; similarly, if $a_w > b_w$ then $a_j x_{jk} > b_j x_{jk}$ for all jobs of W_s that follow J_w. In any case, the contribution of the jobs that are contained in set W_s to the makespan $C_{\max}(S^0_{LP})$ does not exceed $a_w + b_w$ plus total processing time of these jobs on one of the machines. This implies that the length of the critical path with job J_w being critical does not exceed the length of the longer path is which either big job J'_{k-1} or big job J'_k is critical plus the value of $a_w + b_w$. As above, the length of the critical path with a big critical job is bounded by $C^0 + \delta^2 T$. Thus, if not a big job is critical in schedule S^0_{LP}, we derive that $C_{\max}(S^0_{LP}) \leq C^0 + 2\delta^2 T \leq C_{\max}(S^*) + 2\delta^2 T$.

When the pseudo-jobs are removed from schedule S^0_{LP} and the fractional jobs and the medium jobs are appended to that schedule, for the resulting schedule we have that

$$C_{\max}(S_\varepsilon) \leq \max\{C_{\max}(S^0_{LP}), t\} + 2n_b\delta^2 T + \tilde{\varepsilon} T \leq C_{\max}(S^*) + (2n_b + 2)\delta^2 T + \tilde{\varepsilon} T.$$

Case 2

We now look for an approximate solution such that a big job J'_v, where $1 \leq v \leq n_b$, is the first big job that completes on machine B after the hole. Due to Lee (1997, 1999), we only have to consider schedules in which the big jobs that precede job J'_v are sequenced by Johnson's rule, and so are the big jobs that follow job J'_v. Thus, to obtain a sequence of big jobs that is associated with a certain schedule we need to split the set of the remaining big jobs into two subsets, so that the jobs of one subset are positioned before job J'_v (we call this subset the *front part*) and the jobs of the other subset are positioned after job J'_v (we call this subset the *rear part*). Our approximation scheme will generate all possible partitions of the set of big jobs into the front and rear parts.

We give further description and analysis of the approximation scheme, provided that job J'_v is fixed and the partition of the remaining big jobs into the front and rear parts is

also fixed. Suppose that there are $h-1$ big jobs in the front part. Sequence the jobs in the front part and those in the rear part by Johnson's rule. For notational convenience, relabel the big jobs in such a way that the obtained sequence is given by $(I_0, I_1, \ldots, I_{h-1}, I_h = J'_v, I_{h+1}, \ldots, I_{n_b}, I_{n_{b+1}})$, where I_0 and $I_{n_{b+1}}$ are the dummy jobs with zero processing times on both machines (similar to J'_0 and J'_{n_b+1} used in Case 1). Similarly to Case 1, for a big job I_j denote its processing times on machines A and B by a'_j and b'_j, respectively.

For a small job J_j, we define the variables

$$x_{jk} = \begin{cases} 1, & \text{if } J_j \text{ is scheduled between jobs } I_{k-1} \text{ and } I_k, \\ 0, & \text{otherwise,} \end{cases}$$

for $k = 1, \ldots, n_b + 1$.

Case 2a

We first study the situation that an approximate solution to the original problem is sought for in class $\bar{\mathcal{S}}_v$. The following integer program is a relaxation of the problem of finding a schedule from class $\bar{\mathcal{S}}_v$ for processing the big jobs and the small jobs. The variable C provides a lower bound on the makespan of that partial schedule. The variable R_h corresponds to the starting time of job $I_h = J'_v$ on machine B. We call this integer program $\overline{IP}(v)$.

$$C \to \min$$

$$\text{s.t. } t \le R_h; \tag{11}$$

$$\sum_{k=1}^{h-1} \left(\sum_{j \in W_s} a_j x_{jk} + a'_k \right) + a'_h \le R_h; \tag{12}$$

$$\sum_{k=1}^{u} \left(\sum_{j \in W_s} a_j x_{jk} + a'_k \right) + b'_u + \sum_{k=u+1}^{h} \left(\sum_{j \in W_s} b_j x_{jk} + b'_k \right) - b'_h + \Delta \le R_h, \quad u = 1, \ldots, h-1; \tag{13}$$

$$R_h + b'_h + \sum_{k=h+1}^{n_b+1} \left(\sum_{j \in W_s} b_j x_{jk} + b'_k \right) \le C; \tag{14}$$

$$\sum_{k=1}^{u} \left(\sum_{j \in W_s} a_j x_{jk} + a'_k \right) + b'_u + \sum_{k=u+1}^{n_b+1} \left(\sum_{j \in W_s} b_j x_{jk} + b'_k \right) \le C; u = h+1, \ldots, n_b; \tag{15}$$

$$\sum_{k=1}^{u} \left(\sum_{j \in W_s} a_j x_{jk} + a'_k \right) + b'_u + \sum_{k=u+1}^{h-1} \left(\sum_{j \in W_s} b_j x_{jk} + b'_k \right) \le s; \quad u = 1, \ldots, h-1; \tag{16}$$

$$\sum_{k=1}^{n_b+1} x_{jk} = 1, \qquad j \in W_s; \tag{17}$$

$$x_{jk} \in \{0, 1\}, \qquad j \in W_s, \ k = 1, \ldots, n_b + 1. \tag{18}$$

Constraints (11) and (13) guarantee that job I_h starts on B after the hole and not earlier than a preceding small job completes on that machine, provided that big job I_u is critical, $1 \le u \le h-1$. Constraint (12) does not allow job I_h to start on B earlier than that job completes on machine A. Constraints (14) and (15) give lower bounds on the value of makespan, provided that job I_u is critical. Constraint (16) implies that all big jobs

$I_1, \ldots, I_{h-1}$ in the partial schedule must be completed on B before the hole. Constraints (17) have the same meaning as in $IP(0)$. Notice that in the formulation of this integer program the resumable scenario is assumed.

As in Case 1, we solve the linear programming relaxation of this problem in which the constraints $x_{ij} \in \{0, 1\}$ in (18) are replaced by the non-negativity constraints $x_{ij} \geq 0$. The relaxation problem may appear to be infeasible, but that only means that a wrong partition has been used for a given job J'_v; for further purposes we are only interested in situations that a linear programming relaxation can be solved to optimality. Similarly to Case 1, it can be verified that a basic optimal solution the relaxation problem contains at most $n_b + h - 1$ fractional jobs, which is again no more than $2n_b$.

The resulting schedule S_ε can be found as in Case 1, i.e., by introducing pseudo-jobs; ordering the jobs between the big jobs according to Johnson's rule; determining a schedule $\bar{S}^v_{LP}$ associated with the found permutation, provided that the crossover job, if exists, is processed in accordance with the chosen scenario; removing all pseudo-jobs and appending all fractional small jobs and all medium jobs in an arbitrary order starting at time $C_{\max}(\bar{S}^v_{LP})$.

We prove that (10) holds, provided that S^* is an optimal schedule associated with the same choice of job $J'_v = I_h$ and the same partition of the other big jobs into the front and rear parts. Let $\bar{C}^v$ denote an optimal value of C in the linear programming relaxation of the integer program $\overline{IP}(v)$.

Similarly to Case 1, it can be seen that

- $C_{\max}(\bar{S}^v_{LP}) = \bar{C}^v$ if in schedule $\bar{S}^v_{LP}$ a big job J'_u for $u \geq h$ is critical; or
- $C_{\max}(\bar{S}^v_{LP}) \leq \bar{C}^v + \delta^2 T$ if either the critical job is either a big job J'_u for $u \leq h-1$ or one of the small jobs (or, possibly, pseudo-jobs) positioned after job I_h; or
- $C_{\max}(\bar{S}^v_{LP}) \leq \bar{C}^v + 2\delta^2 T$ for any other critical job.

When the pseudo-jobs are removed from schedule $\bar{S}^v_{LP}$ and the fractional jobs and the medium jobs are appended to that schedule, for the resulting schedule we have that

$$C_{\max}(S_\varepsilon) \leq C_{\max}(\bar{S}^v_{LP}) + 2n_b\delta^2 T + \tilde{\varepsilon}T \leq C_{\max}(S^*) + (2n_b + 2)\delta^2 T + \tilde{\varepsilon}T.$$

Case 2b

We now consider the situation that an approximate solution to the original is sought for in class $\tilde{\mathcal{S}}_v$. The following integer program is a relaxation of the problem of finding a schedule from class $\tilde{\mathcal{S}}_v$ for processing the big jobs and the small jobs. The variable C provides a lower bound on the makespan of that partial schedule. The variable R_h corresponds to the starting time of job $I_h = J'_v$ on machine B. We call this integer program $\widetilde{IP}(v)$.

$$C \to \min$$

$$\text{s.t. } R_h \le s; \tag{19}$$

$$R_h + b'_h > s; \tag{20}$$

$$\sum_{k=1}^{h-1}\left(\sum_{j\in W_s} a_j x_{jk} + a'_k\right) + a'_h \le R_h; \tag{21}$$

$$\sum_{k=1}^{u}\left(\sum_{j\in W_s} a_j x_{jk} + a'_k\right) + b'_u + \sum_{k=u+1}^{h}\left(\sum_{j\in W_s} b_j x_{jk} + b'_k\right) - b'_h \le R_h, u = 1,\dots,h-1; \tag{22}$$

$$R_h + b'_h + \Delta + \alpha(s - R_h) + \sum_{k=h+1}^{n_b+1}\left(\sum_{j\in W_s} b_j x_{jk} + b'_k\right) \le C; \tag{23}$$

$$\sum_{k=1}^{u}\left(\sum_{j\in W_s} a_j x_{jk} + a'_k\right) + b'_u + \sum_{k=u+1}^{n_b+1}\left(\sum_{j\in W_s} b_j x_{jk} + b'_k\right) \le C, \qquad u = h+1,\dots,n_b; \tag{24}$$

$$\sum_{k=1}^{n_b+1} x_{jk} = 1, \qquad j \in W_s; \tag{25}$$

$$x_{jk} \in \{0,1\}, \qquad j \in W_s,\ k = 1,\dots,n_b+1. \tag{26}$$

Notice that in the formulation of this integer program the semi-resumable scenario is applied. Constraints (19) and (20) imply job I_h is the crossover job. Constraints (22) guarantee that job I_h starts on B not earlier than a preceding small job completes on that machine, provided that big job I_u is critical, $1 \le u \le h-1$. Constraint (21) does not allow job I_h to start on B earlier than that job completes on machine A. Constraint (23) gives a lower bound on the makespan, provided that job I_h is the crossover job, and one of the jobs I_u is critical, $1 \le u \le h$. Constraints (15) give lower bounds on the value of makespan, provided that job I_u is critical, $h+1 \le u \le n_b$.

We solve the linear programming relaxation of this problem. As in the Case 2a, ignoring the problems that appear to be infeasible, it can be verified that a basic optimal solution of the relaxation problem contains at most $n_b + 1$ fractional jobs, which is again no more than $2n_b$.

The resulting schedule S_ε can be found as in Case 2a. Let $\tilde{S}^v_{LP}$ be an analog of schedule $\bar{S}^v_{LP}$ defined in Case 2a. Notice that in schedule $\bar{S}^v_{LP}$ a crossover job is not a big job, while in schedule $\bar{S}^v_{LP}$ the crossover job is job I_h that starts on B at time R_h, as determined by the solution of the linear programming relaxation.

We prove that (10) holds, provided that S^* is an optimal schedule associated with the same choice of the crossover job $J'_v = I_h$ and the same partition of the other big jobs into the front and rear parts. Let $\tilde{C}^v$ denote an optimal value of C in the linear programming relaxation of the integer program $\widetilde{IP}(v)$.

Similarly to previous case, it can be seen that either $C_{\max}(\tilde{S}^v_{LP}) = \tilde{C}^v$ (if in schedule $\tilde{S}^v_{LP}$ a big job is critical); or $C_{\max}(\tilde{S}^v_{LP}) \le \bar{C}^v + \delta^2 T$ for any other critical job.

When the pseudo-jobs are removed from schedule $\tilde{S}^v_{LP}$ and the fractional jobs and the medium jobs are appended to that schedule, for the resulting schedule we have that

$$C_{\max}(S_\varepsilon) \le C_{\max}(\tilde{S}^v_{LP}) + (n_b+1)\delta^2 T + \tilde{\varepsilon} T \le C_{\max}(S^*) + (2n_b+2)\delta^2 T + \tilde{\varepsilon} T.$$

The value of δ is chosen in such a way that $n_b \leq 1/\delta$ and $\delta \leq \tilde{\varepsilon}^2 \leq \tilde{\varepsilon}$, so that for $\tilde{\varepsilon} < 1$ we have that

$$(2n_b + 2)\delta^2 \leq 2\delta + 2\delta^2 \leq 4\delta \leq 4\tilde{\varepsilon}^2 \leq 4\tilde{\varepsilon}.$$

This implies that

$$C_{\max}(S_\varepsilon) \leq C_{\max}(S^*) + 5\tilde{\varepsilon}T \leq (1+\varepsilon)C_{\max}(S^*), \tag{27}$$

where the final inequality is obtained from $T \leq 2C_{\max}(S^*)$ and our choice $\tilde{\varepsilon} = \varepsilon/10$.

We now provide the main result in this section.

Theorem 2 *For problem $F2|h(0,1), S\text{-}Re|C_{\max}$ the family of approximation algorithms for finding schedule S_ε is a polynomial-time approximation scheme.*

Proof. Inequality (27) establishes that some schedule S_ε that is generated by the algorithm provides a makespan that is no more than $1+\varepsilon$ times the optimal makespan. Thus, it remains to show that the algorithm requires polynomial time.

The algorithm constructs at most $n_b 2^{n_b} + 1$ schedules, one in class $\mathcal{S}_0$ and at most $2^{n_b - 1}$ schedules in each class $\bar{\mathcal{S}}_v$ and $\tilde{\mathcal{S}}_v$ due to partitioning the big jobs other than job J'_v. This number of schedules is fixed for a fixed ε. For each schedule, a linear programming problem is solved, the number of variables and the number of constraints being bounded from above by a polynomial of n_s and n_b. Such a linear program is solvable in polynomial time, using the algorithm of Vaidya (1989), for example. To obtain the final schedule from the solution of the linear program, the small jobs are sequenced using Johnson's algorithm, see Johnson (1954), in $O(n_s \log n_s)$ time. Thus, the running time of the algorithm is polynomial. ∎

5 Conclusions

In this paper we consider the two-machine flow shop scheduling problem with availability constraints under different scenarios. The contribution of this paper against the previously known results is summarized in Table 1 (for the resumable scenario) and Table 2 (for the semi-resumable scenario).

Structure of holes	Resumable	
	Previously known	In this paper
$(1,0)$	Dynamic programming, Lee (1997) $\rho = \frac{4}{3}$, Cheng and Wang (2000)	FPTAS, Section **??**
$(0,1)$	Dynamic programming, Lee (1997) $\rho = \frac{4}{3}$, Lee (1999)	FPTAS, Section **??**
$(q,0)$		Dynamic programming, Section **??** FPTAS, Section **??** $\rho = \frac{3}{2}$, Section **??**
$(0,q)$	Not approximable for $q \geq 2$, Kubiak et al (2002)	Dynamic programming, Section **??**
$(1,1)$	Not approximable, Kubiak et al (2002)	

Table 1: Results for the two-machine flow shop scheduling problem with availability constraints under the resumable scenario

Strucure of holes	Semi-resumable	
	Previously known	In this paper
$(1,0)$	Dynamic programming, Lee (1999) $\rho = 2$, Lee (1999)	PTAS, Section 4
$(0,1)$	Dynamic programming, Lee (1999) $\rho = \frac{3}{2}$, Lee (1999)	PTAS, Section 4
$(q,0)$	Not approximable for $q \geq 2$, Lee (1996), Breit et al (2003)	
$(0,q)$	Not approximable for $q \geq 2$, Lee (1996), Breit et al (2003)	
$(1,1)$	Not approximable, Lee (1996), Breit et al (2003)	

Table 2: Results for the two-machine flow shop sheduling problem with availability constraints under the semi-resumable scenario

It can be seen that the paper provides a fairly complete approximability classification of the relevant problems. An interesting topic for future research is whether the two-machine flow shop problem with a single hole under the semi-resumable scenario admits a FPTAS.

References

1. Breit, J., G. Schmidt and V.A. Strusevich. 2003. Non-preemptive two-machine open shop scheduling with non-availability constraints. Report B0101, Department of Economics, University of Saarland (to appear in *Mathematical Methods of Operations Research*).

2. Cheng, T.C.E., G. Wang. 1999. Two-machine flowshop scheduling with consecutive availability constraints. *Information Processing Letters* **71** 49-54.

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

4. Espinouse, M.-L., P. Formanowicz and B. Penz. 1999. Minimizing the makespan in the two machine no-wait flow-shop with limited machine availability. *Computers & Industrial Engineering* **37** 497-500.

5. Espinouse, M.-L., P. Formanowicz and B. Penz. 2001. Complexity results on and approximation algorithms for the two machine no-wait flow-shop with limited machine availability. *Journal of the Operational Research Society* **52** 116-121.

6. Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., New York.

7. Gilmore, P.C., R.E. Gomory. 1964. Sequencing a one-state variable machine: a solvable case of the traveling salesman problem. *Operations Research* **12** 655-679.

8. Johnson, S.M. 1954. Optimal two- and three-stage production schedules with Setup times included. *Naval Research Logistics Quarterly* **1** 61–68.

9. Kubiak, W., J. Błażevicz, P. Formanowicz, J. Breit, G. Shmidt. 2002. Two-machine flow shop with limited machine availability. *European Journal of Operational Research* **136** 528-540.

10. Kubzin, M.A., V.A. Strusevich. 2002. Two-machine flow shop no-wait scheduling with a non-availability interval. Paper 02/IM/99, CMS Press, University of Greenwich, London, U.K.

11. Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys. 1993. "Sequencing and scheduling: algorithms and complexity," in Handbooks in *Operations Research and Management Science*, vol. 4, Logistics of Production and Inventory, S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin (Editors), North–Holland, Amsterdam. 455-522.

12. Lee, C.-Y. 1996. Machine scheduling with an availability constraint. *Journal of Global Optimization* **9** 395–416.

13. Lee, C.-Y. 1997. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters* **20** 129-139.

14. Lee, C.-Y. 1999. Two-machine flowshop scheduling with availability constraints. *European Journal of Operations Research* **114** 420-429.

15. Potts, C.N. 1985. Analysis of a linear programming heuristic for scheduling unrelated parallel machines, *Discrete Applied Mathematics* **10** 155–164.

16. Sanlaville, E., G. Schmidt. 1998. Machine scheduling with availability constraints. *Acta Informatica* **35** 795-811.

17. Schmidt, G. 2000. Scheduling with limited machine availability. *European Journal of Operations Research* **121** 1-15.

18. Sevastianov, S.V., G.J. Woeginger. 1998. Minimizing makespan in open shops: A polynomial time approximation scheme, *Mathematical Programming* **82** 191–198.

19. Vaidya, P.M. 1989. Speeding up linear programming using fast matrix multiplication. In: *Proceedings of IEEE 30th Annual Symposium on Foundations of Computer Science*. 332–337.

20. Wang, G., T.C.E. Cheng. 2001. Heuristics for two-machine no-wait flowshop scheduling with an availability constraint. *Information Processing Letters* **80** 305-309.