Complexity and Algorithms for Two-Stage Flexible Flowshop Scheduling with Availability Constraints

Jinxing Xie*, Xijun Wang
Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China
jxie@math.tsinghua.edu.cn

Abstract

This paper considers the two-stage flexible flowshop scheduling problem with availability constraints. We discuss the complexity and the approximability of the problem, and provide some approximation algorithms with finite and tight worst case performance bounds for some special cases of the problem.

Keywords: Scheduling; Flexible flowshop; Availability constraints; Approximability; Approximation algorithms; Worst-case analysis

1 Introduction

A flexible flowshop, which is also known as hybrid flowshop or flowshop with parallel machines, consists of a set of machine centers with parallel machines. Scheduling problems for such kind of flowshops are firstly studied by Salvador in 1973 [1]. From then on, the flexible flowshop scheduling problems with makespan objective have attracted a great deal of attention.

For example, the two-stage problem $F_2(P) \mid \cdot \mid C_{max}$ is already proved to be $\mathcal{NP}$-hard in the strong sense, even in the case where there are only two machines in one stage and there is only a single machine in the other stage [2]. Various heuristics or approximation algorithms for these problems have been designed [3, 4, 5], for some of which the corresponding worst case performance bounds and the average case performance bounds have been proved. Hall [6], Schuurman and Woeginger [7] have discussed the existence of polynomial time approximation schemes (PTAS) for flexible flowshop scheduling problems. Recently, the scheduling problems for flexible flow shops with sequence-dependent setup times are considered, where some heuristics are proposed and their computational effectiveness are examined [8, 9].

Most studies on scheduling problems assume that the machines are available at all times. In real industry settings, however, a machine may not always be available in the scheduling period due to, for example, a breakdown (stochastic) or preventive maintenance (deterministic). This paper only considers the scheduling problem under the deterministic case, i.e., the unavailable time intervals called holes are deterministically known before the decision making for the schedules starts.

*Corresponding author. This research has been supported by NSFC Project No. 70471008.

In some models with availability constraints, a job started its processing but not finished before the unavailable period can be resumed without restarting when the machine becomes available again, which is usually the case in the food industry. The job is then called resumable. Otherwise, if a job started its processing but not finished before the unavailable period must be restarted later, which usually occurs in steel industry, it is called nonresumable. There is also a semiresumable case, in which a job started its processing but not finished before the unavailable period will have to partially restart when the machine becomes available again [10]. This paper only considers the scheduling problems in the resumable case.

Scheduling problems with limited machine availability have been studied to a less extent. Adiri et al. [11] showed that a single machine problem with machine breakdowns is $\mathcal{NP}$-hard if the breakdowns are known in advance. Schmidt [12] studied parallel machine scheduling problems with availability constraints. Hwang and Chang [13] proved that the makespan of the LPT (Longest Processing Time) schedule is bounded by twice the optimal makespan if no more than half of the machines are allowed to be shutdown simultaneously. Lee [14] proved that the two-machine flowshop problem with makespan objective and one unavailability period is $\mathcal{NP}$-hard in the ordinary sense, and presented a dynamic programming approach and some approximation algorithms. Kubiak et al. [15] proved that the two-machine problem with arbitrary number of unavailability periods on one machine is NP-hard in the strong sense, and proposed a branch and bound algorithm and some heuristic algorithms [16] . Lee [10] given the first discussion about the semiresumable two-machine flow shop model and provided worst case performance analysis for some algorithms. Espinouse et al. [17] studied both the resumable and the nonresumable two-machine flowshop models under the no-wait environment. Recently, the flexible flowshop scheduling problems problems with limited machine availability are firstly studied by Wang and Xie [18], in which a bround and bound algorithm is presented.

This paper discusses the complexity and approximability of two-stage flexible flowshop problems for the nonresumable case with deterministic availability constraints. In the next section, we give the notations of the problem. In Section 3, we point out the strongly $\mathcal{NP}$-hardness firstly; then we prove the $\mathcal{APX}$-hardness of the problems with holes on the whole second stage, or on the whole first stage but with more than one machine at this stage. In Section 4, we analyze the worst case performance of two algorithms for the case where there is only one machine at the first stage and only one hole on that machine. In section 5, we provide an algorithm with finite worst case performance bound for the special case where the unavailable periods apply only on a part of machines at some machine centers. In the last section, we conclude this paper with a short discussion.

2 Problem formulation

Let us suppose that the flexible flowshop consists of a set of $m \geq 2$ machine centers $[Z_1, Z_2, \ldots, Z_m]$ with center Z_j having $m_j \geq 1$ identical parallel machines $\{M_{j1}, M_{j2}, \ldots, M_{jm_j}\}$. There are n jobs $\{J_i \mid 1 \leq i \leq n\}$ to be processed in the flowshop. Function $p(J_i) = [p_{i1}, p_{i2}, \ldots, p_{im}]$ will denote the processing times required by J_i on

centers $[Z_1, Z_2, \ldots, Z_m]$ respectively. For convenience unavailable periods will be called holes. We assume that the holes do not overlap, i.e., on each machine the holes do not cross over with each other, since otherwise they should be considered as one hole only. Therefore we can denote by $s_{jk,g}$ and $l_{jk,g}$ the starting time and the length, respectively, of the hole g on the machine M_{jk}, numbered according to their starting times. Each machine can process at most one job at a time and each job can be processed by at most one machine at a time. The objective considered is to minimize the makespan of the schedule (the maximum completion time of all jobs).

We use in this paper the notation similar to the one described in [15]. Specifically, $F_2(P), h_{jk,g} \mid \cdot \mid C_{max}$ represents the two-stage flexible flowshop problem with an arbitrary number of holes on each machine. $F_2(P), h_{1k,g} \mid \cdot \mid C_{max}$ represents the problem with an arbitrary number of holes on each machine at the first stage but no holes on the second stage. $F_2(P), h_{11,1} \mid \cdot \mid C_{max}$ represents the problem with one hole on machine M_{11} only. In this notation $h_{jk,g}$ specifies the number of the hole(s) and the machine(s) on which they appear. If j or k is replaced by a positive integer, it means that holes are only on those machines. Otherwise, holes will be allowed to be on all machines. If g is replaced by a positive integer, it denotes the number of holes on the corresponding machine. Otherwise this number is arbitrary.

3 $\mathcal{NP}$-hardness and $\mathcal{APX}$-hardness

Due to the strongly $\mathcal{NP}$-hardness of both $F_2(P) \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P) \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ [2], Theorem 1 stands obviously.

Theorem 1 *$F_2(P), h_{11,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$, $F_2(P), h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$, $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P), h_{2k,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ are $\mathcal{NP}$-hard in the strong sense.*

During the approximability discussions in the following, several specific notations will be used. $C^*_{max}(I)$ and $C^H_{max}(I)$ denote, respectively, the optimal makespan and the makespan given by algorithm H for any instance I. They are often abbreviated to C^*_{max} and C_{max} respectively. An algorithm for problems minimizing makespan is called having *worst case performance bound* ρ if $C_{max} \leq \rho C^*_{max}$ for all instances. From this definition it is easily to see that $\rho \geq 1$ always holds. Furthermore, the bound is called *tight* if, for all $\varepsilon > 0$, there exists instance I satisfying $C_{max} > (\rho - \varepsilon)C^*_{max}$.

Theorem 2 *A polynomial time algorithm for the problem $F_2(P), h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ with a finite worst case performance bound cannot be found unless $\mathcal{P} = \mathcal{NP}$.*

Proof The theorem will be proved by a contradiction. Let us suppose now that there exists an algorithm H which gives a solution in polynomial time for this problem with worst case bound R, i.e., $C^H_{max}(I) \leq RC^*_{max}(I)$ for all instances of $F_2(P), h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$. For any instance I' of problem $F_2(P) \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and for any integer $y > 0$, we construct instance I'' of $F_2(P), h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ as follows: All are the same

to instance I' except that there is one hole with $s_{21,1} = y, l_{21,1} = Ry$ on machine M_{21}. We can see that there exists a schedule satisfying $C_{max} \leq y$ for instance I' if and only if the schedule generated by algorithm H satisfies $C_{max} \leq y$ for instance I''. This is inconsistent with the $\mathcal{NP}$-hardness of $F_2(P) \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ unless $\mathcal{P} = \mathcal{NP}$. ∎

Lemma 3 *A polynomial time algorithm for the problem $P, h_{k,1} \mid \cdot \mid C_{max}$ with a finite worst case bound cannot be found unless $\mathcal{P} = \mathcal{NP}$.*

Proof Similarly to the proof of Theorem 2, the existence of a polynomial time algorithm for $P, h_{k,1} \mid \cdot \mid C_{max}$ with a finite worst case bound will be inconsistent with the $\mathcal{NP}$-hardness of $P \mid \cdot \mid C_{max}$ unless $\mathcal{P} = \mathcal{NP}$ [19]. ∎

From Lemma 3, Theorem 4 holds obviously.

Theorem 4 *A polynomial time algorithm for the problem $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P), h_{2k,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ with a finite worst case bound cannot be found unless $\mathcal{P} = \mathcal{NP}$.*

Corollary 5 *A polynomial time algorithm for the problem $F_2(P), h_{1k,1}, h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P), h_{11,1}, h_{2k,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ with a finite worst case bound cannot be found unless $\mathcal{P} = \mathcal{NP}$.*

However, unlike $F_2(P), h_{21,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$, there exist polynomial time algorithms for $F_2(P), h_{11,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ with finite worst case bounds. This result can be seen from the discussions in the next section, and it differs very much from the symmetry between $F_2(P) \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P) \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$.

4 Worst case performance of algorithms for $F_2(P), h_{11,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$

4.1 List scheduling algorithm

In this algorithm, a list L (or permutation) of the job indices $1, 2, \ldots, n$ is provided. Jobs are fed to the first machine center Z_1 in the order they appear on the list L. Since we have only one machine at Z_1, the jobs processed at Z_1 form a queue at the buffer between the center Z_1 and Z_2 in the same order L. Then the jobs are processed in the order L, whenever the machine at Z_2 becomes available.

Theorem 6 *For any instance of $F_2(P), h_{11,1} \mid m_1 = 1, m_2 = \mu \geq 2 \mid C_{max}$, let f^* and f be the finish time of an optimal schedule and the schedule obtained by the list scheduling algorithm respectively. Then $\frac{f}{f^*} \leq 4 - \frac{1}{\mu}$, and this bound is tight.*

Proof In this problem, there is only one machine M_{11} in the first machine center, and there are μ identical machines in the second machine center. Besides, there is only one hole and it is on M_{11}, where the starting time and the length of the hole are denoted by $s_{11,1}$ and $l_{11,1}$ respectively. If the tasks assigned to M_{11} by the list scheduling algorithm are all before $s_{11,1}$, f^* and f will be the same respectively to those in the case the availability constraint is discarded. From [4], we have $\frac{f}{f^*} \leq 3 - \frac{1}{\mu}$.

Now suppose that the tasks assigned to M_{11} by the list scheduling algorithm are not all before $s_{11,1}$. Let f_1 be the finish time on M_{11} of the schedule obtained by the list scheduling algorithm. First we process all n tasks $\{p_{i1}\}$ on Z_1. And we start the processing of any tasks of $\{p_{i2}\}$ only after the completions of all of $\{p_{i1}\}$ on Z_1. Then the processing of $\{p_{i2}\}$ on Z_2 can be treated as a parallel-machine shop. The tasks p_{i1} and p_{i2}, $1 \leq i \leq n$, have to be carried out in the order specified in the list. Let f_2 be the finish time of all of $\{p_{i2}\}$ when they are processed in the parallel-machine shop Z_2 in the order of the list. It should be clear that

$$f_1 \leq s_{11,1} + l_{11,1} + \sum_{i=1}^{n} p_{i1}; \tag{1}$$

$$f \leq f_1 + f_2. \tag{2}$$

According to the assumptions of the problem, we have

$$f^* \geq s_{11,1} + l_{11,1}; \tag{3}$$

$$f^* \geq h_{11,1} + \sum_{i=1}^{n} p_{i1}. \tag{4}$$

Please note that there are μ identical machines in the second machine center. From [20], we have

$$\frac{f_2}{f_2^*} \leq 2 - \frac{1}{\mu}, \tag{5}$$

where f_2^* is the optimal finish time of the operations set $\{p_{i2}\}$ in the parallel-machine shop Z_2. Besides, it is obvious that

$$f^* \geq f_2^*. \tag{6}$$

Using inequalities (1)–(6), we obtain

$$\begin{aligned} \frac{f}{f^*} &\leq \frac{f_1 + f_2}{f^*} \\ &\leq \frac{s_{11,1} + l_{11,1} + \sum_{i=1}^{n} p_{i1}}{f^*} + \frac{f_2}{f_2^*} \\ &\leq 4 - \frac{1}{\mu}. \end{aligned}$$

A worst case instance is constructed using the following job set J:

$p[J_a] = [2\mu\varepsilon, \varepsilon]$;

$p[J_b] = [\mu L - (2\mu - 1)\varepsilon, \varepsilon]$;

$p[J_c(i)] = [\varepsilon, (\mu - 1)L], 1 \leq i \leq \mu - 1$;

$p[J_d(i)] = [\varepsilon, L], 1 \leq i \leq \mu - 1$;

$p[J_e] = [\varepsilon, \mu L]$;

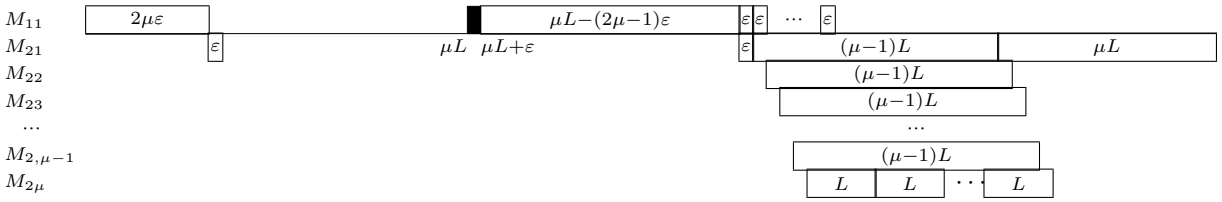
where $L \gg \varepsilon$ and the only hole on M_{11} is $s_{11,1} = \mu L, l_{11,1} = \varepsilon$. Using the processing sequence $[J_a, J_b, J_c, J_d, J_e]$, the schedule obtained by the list scheduling algorithm is shown in Figure 1(a). The optimal schedule is shown in Figure 1(b) in the processing sequence $[J_d, J_e, J_c, J_b, J_a]$. Therefore, we have

$$\begin{aligned} \frac{f}{f^*} &= \frac{(4\mu-1)L-(2\mu-3)\varepsilon}{\mu L+(2\mu+2)\varepsilon} \\ &= \frac{(4\mu-1)-(2\mu-3)\varepsilon/L}{\mu+(2\mu+2)\varepsilon/L}. \end{aligned}$$

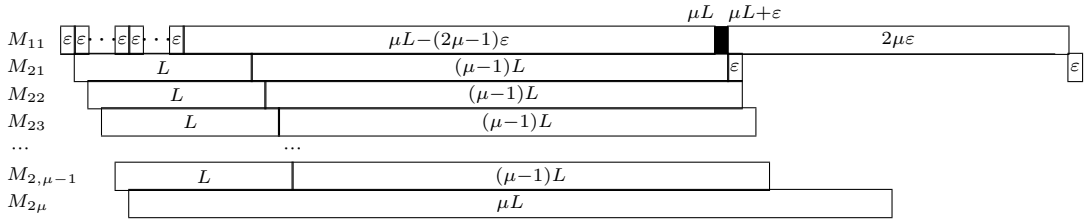
Thus,

$$\lim_{\varepsilon/L\to 0} \frac{f}{f^*} = 4 - \frac{1}{\mu}.$$

This completes the proof. ∎



(a) Schedule by list scheduling algorithm: $f = (4\mu-1)L-(2\mu-3)\varepsilon$



(b) Optimal schedule: $f^* = \mu L + (2\mu+2)\varepsilon$

Figure 1: A worst case instance for list scheduling algorithm

4.2 LPT algorithm

In the LPT (Longest Processing Time) algorithm, we sort the job set according to the processing time on M_{11} in non-increasing order, then we use the list scheduling algorithm.

Theorem 7 *For any instance of $F_2(P), h_{11,1} \mid m_1 = 1, m_2 = \mu \geq 2 \mid C_{max}$, let f^* and f be the finish time of an optimal schedule and the schedule obtained by the LPT algorithm respectively. Then $\frac{f}{f^*} \leq \frac{7}{2} - \frac{1}{\mu}$, and this bound is tight.*

Proof The problem is the same as that of Theorem 6: There is only one machine M_{11} in the first machine center, and there are μ identical machines in the second machine center. Besides, there is only one hole and it is on M_{11}, where the starting time and the length of the hole are denoted by $s_{11,1}$ and $l_{11,1}$ respectively. The only difference is that now we use LPT algorithm instead of a general list scheduling algorithm. Similarly to the proof of Theorem 6, if the tasks assigned to M_{11} by LPT algorithm are all before $s_{11,1}$, we have $\frac{f}{f^*} \leq 3 - \frac{1}{\mu}$ from [4].

Now suppose that $[J_1, J_2, \cdots, J_b]$ and $[J_{b+1}, \cdots, J_n]$ are assigned respectively to M_{11} before and after the hole in that order, where $0 \leq b < n$. Similarly to Theorem 6, we define f_1, f_2^* and f_2. Then inequalities (1)–(6) still hold. Furthermore, we have

$$f_1 = s_{11,1} + l_{11,1} + \sum_{i=b+1}^{n} p_{i1}. \tag{7}$$

Using equation (7), we obtain

$$\begin{aligned} \frac{f}{f^*} &\leq \frac{f_1 + f_2}{f^*} \\ &= \frac{s_{11,1} - \sum_{i=1}^{b} p_{i1}}{f^*} + \frac{l_{11,1} + \sum_{i=1}^{n} p_{i1}}{f^*} + \frac{f_2}{f^*} \\ &\leq \frac{s_{11,1} - \sum_{i=1}^{b} p_{i1}}{f^*} + 3 - \frac{1}{\mu}. \end{aligned} \tag{8}$$

If $b = 0$, we have

$$s_{11,1} \leq p_{11}, \tag{9}$$

$$f^* \geq s_{11,1} + l_{11,1} + p_{11}. \tag{10}$$

Then

$$\begin{aligned} \frac{s_{11,1} - \sum_{i=1}^{b} p_{i1}}{f^*} &= \frac{s_{11,1}}{f^*} \\ &\leq \frac{s_{11,1}}{s_{11,1} + l_{11,1} + p_{11}} \\ &\leq \frac{1}{2}. \end{aligned} \tag{11}$$

If $b \geq 1$, from our algorithm we have

$$\sum_{i=1}^{b} p_{i1} \leq s_{11,1} < \sum_{i=1}^{b+1} p_{i1}. \tag{12}$$

Then

$$\begin{aligned} \frac{s_{11,1} - \sum_{i=1}^{b} p_{i1}}{f^*} &\leq \frac{p_{b+1,1}}{l_{11,1} + \sum_{i=1}^{n} p_{i1}} \\ &\leq \frac{p_{b+1,1}}{p_{b,1} + p_{b+1,1}} \\ &\leq \frac{1}{2}. \end{aligned} \tag{13}$$

So we can see that $\frac{f}{f^*} \leq \frac{7}{2} - \frac{1}{\mu}$ holds for all instances.

The following instance shows that this bound is tight. The job set is given as:

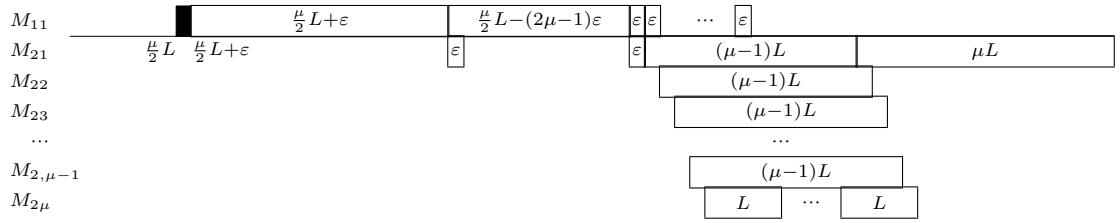
$p[J_a] = [\frac{\mu}{2}L + \varepsilon, \varepsilon]$;

$p[J_b] = [\frac{\mu}{2}L - (2\mu - 1)\varepsilon, \varepsilon]$;

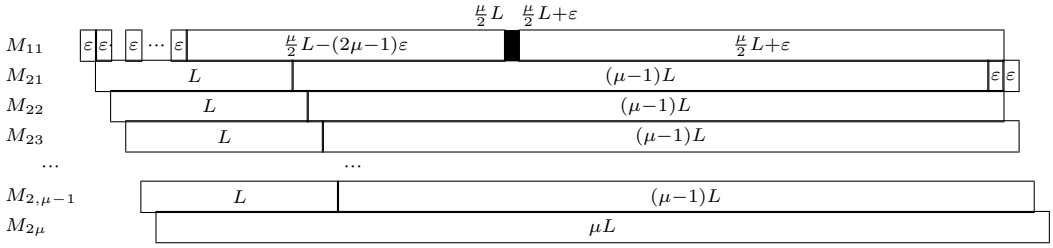
$p[J_c(i)] = [\varepsilon, (\mu - 1)L], 1 \leq i \leq \mu - 1$;

$p[J_d(i)] = [\varepsilon, L], 1 \leq i \leq \mu - 1$;

$p[J_e] = [\varepsilon, \mu L]$;



(a) Schedule by LPT algorithm: $f = (\frac{7}{2}\mu - 1)L - (2\mu - 4)\varepsilon$



(b) Optimal schedule: $f^* = \mu L + \mu\varepsilon$

Figure 2: A worst case instance for LPT algorithm

where $L \gg \varepsilon$ and the only hole is on M_{11} with $s_{11,1} = \frac{\mu}{2}L, l_{11,1} = \varepsilon$. The schedule obtained by LPT algorithm is shown in Figure 2(a) in the processing sequence $[J_a, J_b, J_c, J_d, J_e]$. The optimal schedule is shown in Figure 2(b) in the processing sequence $[J_d, J_e, J_c, J_b, J_a]$. Therefore, we have

$$\lim_{\varepsilon/L \to 0} \frac{f}{f^*} = \frac{7}{2} - \frac{1}{\mu}.$$

This completes the proof. ∎

5 Algorithm for a special case of $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$

From the previous discussions, we know that the flexible flowshop sheduling problems with availability constraints are difficult to be solved because of their $\mathcal{APX}$-hardness properties. According to Theorem 4, both the problems $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ and $F_2(P), h_{2k,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$ are $\mathcal{APX}$-hard. However, in the industrial reality, it's unusual that all the machines at one machine center are breakdown in the same time. By making use of some special limitations on the availability constraints on machines, the corresponding problem might be much easier to be solved. For example, Hwang and Chang [13], Cheng and Wang [21] make some efforts toward this direction for parallel machine scheduling and flowshop scheduling, respectively. According to their work, it will be possible to approximate the problems with unavailable periods only on a part of machines at some machine centers. However, they do not consider the flexible flowshop problems which are the focus of this paper. In fact, as we can see from the following discussion, if we relax the constraints that the holes can be presented on all the machines at all machine centers, then the flexible flowshop problems might not be $\mathcal{APX}$-hard anymore. For example, under the situation where at any time the number of unavailable machines does not exceed one half of the number of all the machines in each machine center, there exist polynomial time algorithms with finite worst case bounds for the

problem $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$. One algorithm for this problem can be described as follows:

Step 0: Sort all the jobs in the non-increasing order in terms of p_{i1} and denote the sequence of jobs as LJ1. That's to say, LJ1 is the LPT sequence of jobs in terms of p_{i1}.

Step 1: Assign the jobs to the first machine center Z_1 for processing as early as possible in according to the sequence LJ1. Let the completion time for job i at Z_1 be C_{i1}. Sort all the jobs in the non-decreasing order in terms of C_{i1} and denote the new sequence of jobs as LJ2.

Step 2: Assign the jobs to the machine in machine center Z_2 for processing as early as possible in according to the sequence LJ2.

Step 3: Calculate the makespan and stop.

Theorem 8 *For any instance of $F_2(P), h_{1k,1} \mid m_1 \geq 2, m_2 = 1 \mid C_{max}$ where at any time the number of unavailable machines does not exceed one half of the number of the machines (at the first stage), let f^* and f be the makespan of an optimal schedule and the schedule obtained by the above algorithm respectively. Then $\frac{f}{f^*} \leq 3$.*

Proof Consider the sub-problem with only the first machine center as a classical parallel scheduling problem $P \mid \cdot \mid C_{max}$. Let f_1^* and f_1 be the makespan of the optimal schedule and the schedule obtained by the LPT algorithm [13] respectively. Then we have

$$f^* \geq f_1^*, \tag{14}$$

$$f^* \geq \sum_{i=1}^{n} p_{i2}, \tag{15}$$

$$f \leq f_1 + \sum_{i=1}^{n} p_{i2}. \tag{16}$$

According to [13],

$$\frac{f_1}{f_1^*} \leq 2. \tag{17}$$

From (14)–(17),

$$\frac{f}{f^*} \leq \frac{f_1 + \sum_{i=1}^{n} p_{i2}}{f^*} \leq \frac{f_1}{f^*} + \frac{\sum_{i=1}^{n} p_{i2}}{f^*} \leq 3. \tag{18}$$

This completes the proof. ∎

6 Discussion

In this paper, we present firstly the formulation of two-stage flexible flowshop problems with available constraints. Then we discuss the complexity and approximability of these models. We show that they are much more difficult to approximate than the case without availability constraints, except $F_2(P), h_{11,1} \mid m_1 = 1, m_2 \geq 2 \mid C_{max}$. However,

if we relax the constraints that the holes can be presented on all the machines at all machine centers, then these problems might not be $\mathcal{APX}$-hard anymore.

References

[1] M. S. Salvador, A solution to a special case of flow shop scheduling problems, in: S. E. Elmaghraby, (Ed.), Symposium on the Theory of Scheduling and Its Applications, Springer-Verlag, Berlin, 83C91, (1973).

[2] J.A. Hoogeveen, J.K. Lenstra, B. Veltman, Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard, European Journal of Operational Research 89, 172–175, (1996).

[3] R. J. Wittrock, An adaptive scheduling algorithm for flexible flow lines, Operations Research 36, 445–453, (1988).

[4] C. Sriskandarajah, S.P. Sethi, Scheduling algorithms for flexible flowshops: Worst and average case performance, European Journal of Operational Research 43, 143–160, (1989).

[5] B. Chen, Analisis of classes of heuristics for scheduling a tow-stage flow shop with parallel machines at one stage, Journal of the Operational Research Society 46, 234–244, (1995).

[6] L.A. Hall, Approximability of flow shop scheduling, Mathematical Programming 82, 175–190, (1998).

[7] P. Schuurman, G. J. Woeginger, A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem, Theoretical Computer Science 237, 105–122, (2000).

[8] M. E. Kurz, A. G. Ronald, Comparing scheduling rules for flexible flow lines, International Journal of Production Economics 85, 371–388, (2003).

[9] M. E. Kurz, A. G. Ronald, Scheduling flexible flow lines with sequence-dependent setup times, European Journal of Operational Research 159, 66–82 (2004).

[10] C.Y. Lee, Two-machine flowshop scheduling with availability constraints, European Journal of Operational Research 114, 420–429, (1999).

[11] I. Adiri, J. Bruno, E. Frostig, A.H.G. Rinnooy Kan, Single machine flow-time scheduling with a single breakdown, Acta Informatica 26, 679–696, (1989).

[12] G. Schmidt, Scheduling on semi-identical processors, Zeitschrift für Operations Research 28, 153–162, (1984).

[13] H.C. Hwang, S.Y. Chang, Parallel machines scheduling with machine shutdowns, Computers and Mathematics with Applications 36, 21–31, (1998).

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

[15] W. Kubiak, J. Błazewicz, P. Formanowicz, J. Breit, G. Schmidt, Two machine flow shops with limited machine availability, European Journal of Operational Research 136, 528–540, (2002).

[16] J. Błazewicz, J. Breit, P. Formanowicz, W. Kubiak, G. Schmidt, Heuristic algorithms for the two-machine fowshop with limited machine availability, Omega 29, 599–608, (2001).

[17] M.L. Espinouse, P. Formanowicz, B. Penz, Complexity results and approximation algorithms for the two machine no-wait flow-shop with limited machine availability, Journal of the Operational Research Society 52, 116–121, (2001).

[18] X. Wang, J. Xie, Branch and bound algorithm for flexible flowshop with limited machine availability, Asian Information-Science-Life 1, 241-248, (2002).

[19] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theorey of NP-Completeness, Freeman, San Francisco, CA, (1979).

[20] R.L. Graham, Bounds for certain multiprocessing anomalies, The Bell System Technical Journal 45, 1563–1581, (1966).

[21] T.C.E. Cheng, G.Q. Wang, Two-machine flowshop scheduling with consecutive availability constraints, Information Processing Letters 71, 49–54, (1999).