European Journal of Operational Research 281 (2020) 16–24

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A polynomial-time approximation scheme for an arbitrary number of parallel two-stage flow-shops

Jianming Dong[a], Ruyan Jin[a], Taibo Luo[b], Weitian Tong[c,*]

[a] Department of Mathematics, Zhejiang Sci-Tech University, Hangzhou, Zhejiang 310018, China
[b] School of Economics and Management, Xidian University, Xi'an 710126, China
[c] Department of Computer Science, Eastern Michigan University, Ypsilanti, MI 48165, USA

ARTICLE INFO

Article history:
Received 15 November 2018
Accepted 8 August 2019
Available online 12 August 2019

Keywords:
Scheduling
Parallel two-stage flow-shops
Makespan
Mixed integer program
Polynomial-time approximation scheme

ABSTRACT

We investigate the approximability of the m parallel two-stage flow-shop (mP2FS) problem, where a set of jobs is scheduled on the multiple identical two-stage flow-shops to minimize the *makespan*, i.e., the finishing time of the last job. Each job needs to be processed non-preemptively on one flow-shop without switching to the other flow-shops. This problem is a hybrid of the classic parallel machine scheduling and two-stage flow-shop scheduling problems. Its strong NP-hardness follows from the parallel machine scheduling problem when the number of machines is part of the input. Our main contribution is a polynomial-time approximation scheme (PTAS) for the mP2FS problem when the number of shops is part of the input, which improves the previous best approximation algorithm of a ratio $(2+\epsilon)$. Owing to the strong NP-hardness, our PTAS achieves the best possible approximation ratio.

1. Introduction

In the *m parallel two-stage flow-shop* (mP2FS) problem, n jobs $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$ need to be assigned to m identical two-stage flow-shops $F_1,F_2,\ldots,F_m$, each of which consists of two sequential machines. Denote F_ℓ by $(M_{\ell,1}, M_{\ell,2})$, $\ell\in\{1,2,\ldots,m\}$. Each job $J_i, i\in\{1,2,\ldots,n\}$ consists of two operations, say the A-operation and B-operation, and the operations A_i and B_i take a_i and b_i units of time in the first and second stage, respectively. Every operation needs to be processed without preemption and no job can switch among flow-shops once it is assigned to a flow-shop. The objective of the mP2FS problem is to minimize the completion time of the last job, which is called the *makespan*. The mP2FS problem has wide applications in resource scheduling in manufacturing and cloud computing, etc. For example, in the cloud computing and transparent computing systems, a request from a client needs to go through two stages. That is, the requested code/data is first read from secondary memory into the main memory and then sent via the network to the requesting client. The goal is to minimize the maximum waiting time of the clients.

It is easy to observe that the mP2FS problem reduces to the classic *two-stage flow-shop scheduling* problem when $m=1$, and the *parallel machine scheduling* problem when each two-stage flow-shop degenerates to the one-stage flow-shop, i.e., a single machine. Though the two-stage flow-shop scheduling problem achieves an optimal schedule by the well-known Johnson's algorithm (Johnson, 1954), the parallel machine scheduling problem is NP-hard when $m\geq 2$ and becomes strongly NP-hard when m is part of the input (Garey & Johnson, 1979; Hochbaum & Shmoys, 1987), from which the strong NP-hardness of the mP2FS problem naturally follows.

Consider a minimization optimization problem Π with the instances $\mathcal{I}$. The approximation ratio of an approximation algorithm $\mathcal{A}$ is defined as $\rho=\max_{I\in\mathcal{I}}\{\mathcal{A}(I)/\mathrm{OPT}(I)\}$. We, of course, prefer the approximation algorithms with a ratio as close to one as possible. A *polynomial-time approximation scheme* (PTAS) is a type of approximation algorithm $\mathcal{A}_\epsilon$, which admits an approximation ratio $(1+\epsilon)$ for any $\epsilon>0$ and whose time complexity is a polynomial in the instance size. We say a PTAS is a *fully polynomial-time approximation scheme* (FPTAS) if its running time is a polynomial in $\frac{1}{\epsilon}$ and the instance size.

Most previous work on the mP2FS problem has focused on the approximability, either considering a fixed m or allowing m to be part of the input. The most important property leveraged in existing algorithms is that the optimal schedule for a two-stage flow-shop instance can be a *permutation schedule*, in which the jobs are processed on the two machines in the same order. Based on this observation, the general mP2FS problem can be broken down into two subproblems, a job partition problem and a classic two-stage

* Corresponding author.
E-mail address: wtong.research@gmail.com (W. Tong).

https://doi.org/10.1016/j.ejor.2019.08.019
0377-2217/

Table 1
Summary of the best-known results for the mP2FS problem. Our result in this article improves the ratio $(2+\epsilon)$ to $(1+\epsilon)$ for any $\epsilon \in (0, 1)$.

$m = 1$	$m \geq 2$ is fixed: NP-hard (Garey & Johnson, 1979)	m is part of the input: strongly NP-hard (Hochbaum & Shmoys, 1987)
Polynomial (Johnson, 1954)	FPTAS (Dong et al., 2017b)	$(2+\epsilon)$-approx. (Wu & Chen, 2018) **PTAS (our article)**

flow-shop problem (Zhang & Velde, 2012). More precisely, jobs are first sorted into Johnson's order (Johnson, 1954) and then assigned to each flow-shop in some way. Different assignment strategies result in different algorithms.

The mP2FS problem was first investigated by Kovalyov (1985) in 1985 and an FPTAS was presented when m is fixed, which is unfortunately not available online. The original paper (an abstract published in Russian) is only archived in the Library CWI at the Netherlands. This result was not noted by the other researchers in the last three decades until recently when Dong et al. (2017b) proposed another FPTAS independently and a corrigendum (Dong et al., 2017a) was published to credit Dr. Kovalyov for his work (Kovalyov, 1985).

Before that, He, Kusiak, and Artiba (1996)'s work in 1996 was believed to be pioneering on the mP2FS problem. They modeled an application from the glass industry as the mP2FS problem and proposed an efficient heuristic algorithm after formulating the mP2FS problem as a mixed integer linear program. Later in 2000, Vairaktarakis and Elhafsi (2000) investigated the special case when $m = 2$, and presented an optimal algorithm that runs in a pseudo-polynomial time. In 2012, Zhang and Velde (2012) designed a 3/2-approximation algorithm and a 12/7-approximation algorithm for $m = 2$ and 3, respectively. The main idea behind these two algorithms is that the job sequence in Johnson's order is somehow cut into two (three, respectively) parts for the two (three, respectively) two-stage flow-shops to minimize the makespan. Their results were published in the *European Journal of Operational Research*. Recently, in 2017, Dong et al. (2017b) proposed an FPTAS for the mP2FS problem when $m \geq 2$ is a fixed constant. Their idea is to combine a pseudo-polynomial time algorithm and a classic scaling technique for job sizes. An FPTAS is the best possible algorithm owing to the NP-hardness of the mP2FS problem when $m \geq 2$ is fixed.

Wu, Chen, and Wang (2017), Wu and Wang (2017) and Wu and Chen (2018) presented several approximation algorithms for the mP2FS problem considering m as part of the input. Recall that the mP2FS problem is strongly NP-hard when m is part of the input. Wu et al. (2017) studied two special cases, assuming that either the A-operation consumes no less time than the B-operation for each job, or the B-operation takes more time than the A-operation for each job. Both cases admit approximation algorithms with the same ratio 11/6. Later, Wu and Wang (2017) proposed a 17/6-approximation algorithm for the general mP2FS problem. Recently, Wu and Chen (2018) improved the approximation ratios from 17/6 and 11/6 to $(2+\epsilon)$ and $(1.5+\epsilon)$, respectively, for the general case and the previously introduced special cases, where $\epsilon \in (0, 1)$ is an arbitrary small number.

In 2018, Tong, Miyano, Goebel, and Lin (2018) designed a PTAS for the m parallel d-stage flow-shop (mPdFS) problem when both $m \geq 2$ and $d \geq 3$ are fixed integers. This closes the field of studying the approximability of the mPdFS problem when m is fixed, as a PTAS is the best possible algorithm due to its strong NP-hardness when $d \geq 3$. An open question was proposed in Tong et al. (2018) whether a PTAS exists for the mPdFS problem when m is part of the input. Our main contribution of this article is a partial firm answer to this open question, i.e., a PTAS for the mP2FS problem when m is part of the input. A summary of the best known results for the mP2FS problem is shown in Table 1.

Different from the previous approximation algorithms for the mP2FS problem, we do not employ Johnson's algorithm to sort jobs in advance. We first consider a type of specially restricted instances for the mP2FS problem, named as *restricted-type* instances. We show any *feasible* schedule for a restricted-type instance can be transformed into a *feasible strongly structured* (FSS) schedule at a cost of a small increase of the makespan. Then we introduce two novel concepts: configuration and a distribution of configurations over the m input two-stage flow-shops, the latter of which corresponds to an "almost" FSS schedule[1]. In other words, we are able to recover an "almost" FSS schedule from a distribution of configurations over the m input two-stage flow-shops. To retrieve the complete FSS schedule, a *mixed integer linear program* (MILP) based on the above distribution of configurations is constructed and the properties of its basic solutions are explored. As this MILP contains a constant number of integer variables, it can be solved by Lenstra's algorithm (Lenstra Jr, 1983) in polynomial time. In addition, this MILP describes FSS schedules for the given restricted-type instance, and therefore, we can obtain a near-optimal FSS schedule from the solution to this MILP. In the end, we show any mP2FS instance can be reduced to a restricted-type instance using a classic scaling technique by Schuurman and Woeginger (2000). Our PTAS is the best possible algorithm for the mP2FS problem considering its strong NP-hardness. Our result also improves the current best approximation ratio $2+\epsilon$ presented by Wu and Chen (2018).

This article is organized as follows: Section 2 introduces more related work; Section 3 presents the PTAS and the analysis of its time complexity and approximation ratio; finally, Section 4 concludes the article and proposes open questions.

2. Related work

In this section, we introduce related work for the mP2FS problem. The mP2FS problem is closely related to the classic *two-stage flow-shop scheduling* problem (Garey & Johnson, 1979) and the *parallel machine scheduling* problem (Garey & Johnson, 1979).

The two-stage flow-shop scheduling problem can be solved optimally within $O(n\log n)$ time by Johnson's algorithm (Johnson, 1954). However, the general d-stage flow-shop scheduling problem $(d \geq 3)$ is *strongly* NP-hard (Garey, Johnson, & Sethi, 1976). For fixed $d \geq 3$, Hall (1998) designed a PTAS for the d-stage flow-shop scheduling problem. When $d = 2$ or 3, the d-stage flow-shop scheduling problem admits an optimal schedule that is a *permutation schedule*, in which the jobs are processed on all d machines in the same order; but when $d \geq 4$, it has been shown (Conway, Maxwell, & Miller, 1967) that there may exist no optimal schedule that is a permutation schedule. The inapproximability was shown by Williamson et al. (1997) that no approximation algorithm achieves a ratio within 1.25 when d is part of the input. In fact, the question of whether this case admits an approximation algorithm with a constant ratio remains open.

[1] The formal definition of the almost FSS schedule is given in Section 3.2.

Table 2
Known results for the hybrid d-stage flow-shop problem.

	m_j machines in the jth stage, $j \in \{1, 2, \ldots, d\}$.		
	$m_j = 1$	m_j is fixed	m_j is part of the input
$d = 1$	Polynomial time	FPTAS (Sahni, 1976)	PTAS (Hochbaum & Shmoys, 1987)
$d = 2$	Polynomial time (Johnson, 1954)	PTAS (Hall, 1998)	PTAS (Schuurman & Woeginger, 2000)
$d \geq 3$ is fixed	PTAS (Hall, 1998)	PTAS (Hall, 1998)	PTAS (Jansen & Sviridenko, 2000)
d is part of the input	Cannot be approximated within 1.25 (Williamson et al., 1997)		

The m-parallel machine scheduling problem is a classic NP-hard problem when $m \geq 2$ (Garey & Johnson, 1979), and becomes strongly NP-hard if m is part of the input (Hochbaum & Shmoys, 1987). On the positive side, Sahni (1976) presented an FPTAS when m is fixed and Hochbaum and Shmoys (1987) proposed a PTAS when m is part of the input. It is also worth mentioning that Graham (1966) designed a neat and efficient List-Scheduling algorithm with an approximation ratio $(2 - 1/m)$ when m is part of the input.

The mP2FS problem is also closely related to the *hybrid d-stage flow-shop* problem (Lee & Vairaktarakis, 1994; Ruiz & Vázquez-Rodríguez, 2010), another generalization of the d-stage flow-shop scheduling problem and the parallel machine scheduling problem. In a hybrid d-stage flow-shop, the jth stage $j = 1, 2, \ldots, d$ contains $m_j \geq 1$ parallel machines, and the jth operation of J_i needs to be processed non-preemptively on any one of the m_j machines in the jth stage. The objective is also to minimize the makespan. We abbreviate this problem as $(m_1, m_2, \ldots, m_d)$-HFS for simplicity. It is easy to observe that when $m_1 = m_2 = \cdots = m_d = 1$, the problem reduces to the classic d-stage flow-shop problem; when $d = 1$, the problem reduces to the m-parallel machine scheduling problem. The $(m_1, m_2, \ldots, m_d)$-HFS has been studied extensively in the literature (Lee & Vairaktarakis, 1994; Ruiz & Vázquez-Rodríguez, 2010), in particular the special case (m_1, m_2)-HFS with $m_1 = 1$ or $m_2 = 1$ (Chen, 1995; Gupta, 1988; Gupta, Hariri, & Potts, 1997; Gupta & Tunc, 1991). Wang (2005) provided a detailed survey on the hybrid two-stage flow-shop problem with a single machine in one stage. When $\max\{m_1, m_2\} \geq 2$, Hoogeveen, Lenstra, and Veltman (1996) showed that the (m_1, m_2)-HFS problem is strongly NP-hard. Fortunately, Schuurman and Woeginger (2000) presented a PTAS for the (m_1, m_2)-HFS, when m_1 and m_2 are part of the input. For the general $(m_1, m_2, \ldots, m_d)$-HFS with fixed parameters m_1, m_2, …, m_d, Hall (1998) claimed that the PTAS for the classic d-stage flow-shop problem can be extended to the $(m_1, m_2, \ldots, m_d)$-HFS problem. When $m_1, m_2, \ldots, m_d$ are part of the input, Jansen and Sviridenko (2000) generalized the PTAS by Schuurman and Woeginger (2000) to the general $(m_1, m_2, \ldots, m_d)$-HFS as long as d is a fixed integer. Table 2 summarizes the results for the $(m_1, m_2, \ldots, m_d)$-HFS problem we reviewed thus far. In addition, Ruiz and Vázquez-Rodríguez (2010) surveyed plenty of interesting heuristic algorithms for the general $(m_1, m_2, \ldots, m_d)$-HFS problem.

3. A PTAS for the mP2FS problem

In this section, we present a PTAS for the mP2FS problem. In Section 3.1, we introduce the concept of a *restricted-type* instance, a type of specially restricted instances for the mP2FS problem. We also define a *feasible strongly structured* (FSS) schedule such that any *feasible* schedule for a restricted-type instance can be transformed into an FSS schedule without increasing the makespan too much. In Section 3.2, we design a *mixed integer linear program* (MILP) to describe the FSS schedules for the given restricted-type instance and then show how to convert the MILP solution to obtain an FSS schedule. In Section 3.3, we leverage the standard scaling technique by Schuurman and Woeginger (2000) to show that any mP2FS instance can be reduced to a restricted-type instance at a fractional cost. The final PTAS for the mP2FS problem follows from the above discussion.

Recall that in an mP2FS instance there are n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ with $J_i = (A_i, B_i)$, $i \in \{1, 2, \ldots, n\}$. We name the two operations A_i and B_i as the A-operation and B-operation, respectively, for J_i. Let a_i denote the size or processing time for A_i. We define b_i for B_i analogously. A schedule for an mP2FS instance is *feasible* if the schedule satisfies the following restrictions: (1) each job can have at most one operation undergoing processing at any moment; (2) each B-operation cannot be processed until the corresponding A-operation is completed; (3) a job cannot be rearranged to another flow-shop once it is assigned to one flow-shop; (4) each machine in the flow-shops can process at most one operation at any time. Let π and π^* denote a feasible schedule and optimal schedule, respectively. Define the makespan of a schedule π as $C_{\max}(\pi)$.

3.1. A restricted-type instance with an FSS schedule

Let $\alpha = \lceil \frac{43}{\epsilon} \rceil \geq 1$ be a constant integer for any $\epsilon \in (0, 1)$. Suppose δ is a constant less than 1 and its value will be determined later in Eq. (15). We call an operation a *big* operation if its processing time is at least δ; otherwise, we name it as a *small* operation. An mP2FS instance is called a *restricted-type* instance if it satisfies the following conditions:

1. $a_i, b_i \in [0, \alpha^4]$, $\forall\, i \in \{1, 2, \ldots, n\}$;
2. any big operation has an integral size.

Given a restricted-type instance, we partition the job set $\mathcal{J}$ into subsets $\mathcal{J}(x, y)$, $x, y \in \{0, 1, 2, \ldots, \alpha^4\}$ according to the sizes of job operations. These subsets can be further grouped into four categories: jobs with two small operations, jobs with one small operation in the first stage and one big operation in the second stage, jobs with one big operation in the first stage and one small operation in the second stage, and jobs with two big operations. More formally, we define $\mathcal{J}(x, y), x, y \in \{0, 1, 2, \ldots, \alpha^4\}$ as follows:

$$\mathcal{J}(x,y) = \begin{cases} \{J_i \in \mathcal{J} \mid a_i < \delta,\ b_i < \delta\}, & x = 0,\ y = 0; \\ \{J_i \in \mathcal{J} \mid a_i < \delta,\ b_i = y\}, & x = 0,\ y > 0; \\ \{J_i \in \mathcal{J} \mid a_i = x,\ b_i < \delta\}, & x > 0,\ y = 0; \\ \{J_i \in \mathcal{J} \mid a_i = x,\ b_i = y\}, & x > 0,\ y > 0. \end{cases} \tag{1}$$

An interval is named as an η-*interval* if the length of the interval is η. We cut along the time dimension with $(\alpha^2 + 2)$-intervals $I_t = \big[t \cdot (\alpha^2 + 2),\ (t + 1) \cdot (\alpha^2 + 2)\big)$, $t \in \{0, 1, 2, \ldots\}$. For any feasible schedule π, let $s^\pi(O)$ and $f^\pi(O)$ denote the starting processing time and finishing processing time of the operation O, respectively. Define $\mathcal{O}^\pi_{\ell,j,t}$ to be the sequence of operations that start processing on the machine $M_{\ell,j}$ during the time interval I_t, i.e.,

$$\mathcal{O}^\pi_{\ell,j,t} = \begin{cases} \{A_i \mid \pi \text{ assigns } J_i \text{ to } F_\ell \text{ and } s^\pi(A_i) \in I_t\}, & j = 1; \\ \{B_i \mid \pi \text{ assigns } J_i \text{ to } F_\ell \text{ and } s^\pi(B_i) \in I_t\}, & j = 2. \end{cases} \tag{2}$$

If there is no ambiguity, the superscript π in $s^\pi(O)$, $f^\pi(O)$, and $\mathcal{O}^\pi_{\ell,j,t}$ will be omitted in the following context. In addition, if only

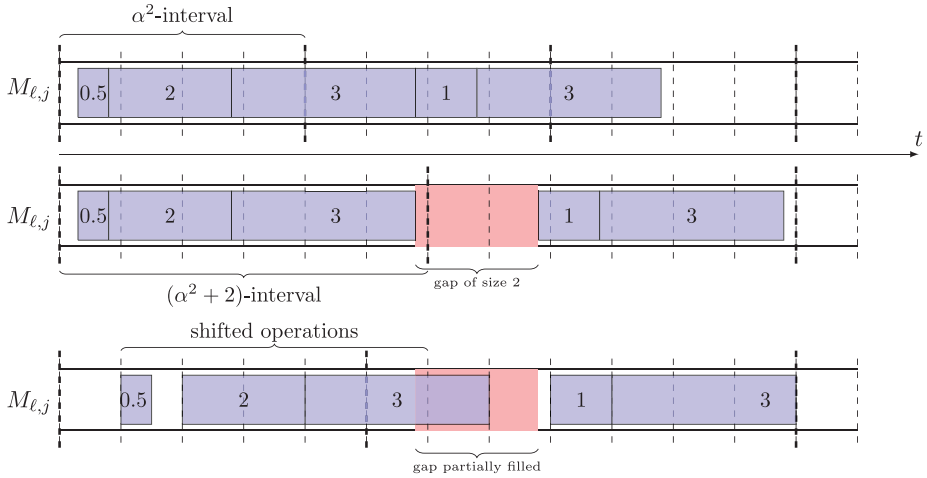

Fig. 1. Illustration of the transformation from π to $\tilde{\pi}$ in Theorem 1. Here we consider some flow-shop F_ℓ and the machine $M_{\ell j}$ on stage j, and assume $\alpha = 2$, $\delta = 1$. Suppose the operations are assigned on $M_{\ell j}$ according to the schedule π, which is shown in the top diagram. The second diagram shows the mapping of the starting time for each operation. The bottom diagram shows the schedule $\tilde{\pi}$ after we delay the processing time of the first small and big operations at some integer time points.

one flow-shop is considered, we may also omit the subscript ℓ. Then we define a *feasible strongly structured* (FSS) schedule for a restricted-type instance.

Definition 1. A feasible schedule for a restricted-type instance is a *feasible strongly structured* (FSS) schedule if it satisfies the following conditions.

1. For each machine $M_{\ell j}$ and each time interval I_t, all operations in $\mathcal{O}_{\ell,j,t}$ are processed in non-decreasing order with respect to the size.
   - If the first operation is small, it starts processing at some integer time point.
   - Every small operation in $\mathcal{O}_{\ell,j,t}$ has to be completed in I_t.
   - Every big operation in $\mathcal{O}_{\ell,j,t}$ has to start processing at some integer time point.
2. For each job J_i, the A-operation and B-operation cannot be processed in the same interval. That is, if $f(A_i) \in I_t$, $s(B_i)$ must be in I_k with $k > t$.

Theorem 1. *For a restricted-type mP2FS instance, if there exists a feasible schedule π with makespan at most α^4, then we can construct from π an FSS schedule π' with makespan at most $(\alpha^2+2)^2$.*

Proof. Recall that $\mathcal{O}^{\pi}_{\ell,j,t}$ is the set of operations assigned by π to start processing on the machine $M_{\ell j}$ during the time interval I_t.

For any feasible schedule π with makespan at most α^4, we cut the interval $[0,\ \alpha^4)$ into uniform α^2-intervals $I'_t = \left[t\cdot\alpha^2,\ (t+1)\cdot\alpha^2\right),\ t \in \{0,1,2,\ldots,\alpha^2-1\}$. Considering some operation O, assume its starting processing time under π is $s^{\pi}(O) = t\cdot\alpha^2 + p \in I'_t$. We construct a loose intermediate feasible schedule $\tilde{\pi}$ by mapping the starting time of each operation O to $s^{\tilde{\pi}}(O) = t\cdot(\alpha^2+2) + p \in I_t$. Refer to the top two diagrams in Fig. 1. Such a shifting strategy will produce an extra gap of size two between the finishing time of the last operation in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ and the starting time of the first operation in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,k}$, where $\mathcal{O}^{\tilde{\pi}}_{\ell,j,k}$ with $k > t$ is the first non-empty set after $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$. We can observe that the schedule $\tilde{\pi}$ is still feasible with a makespan at most $\alpha^2\cdot(\alpha^2+2)$.

Let P be the total processing time, including the waiting time between operations, of operations in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$. We sort operations in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ non-decreasingly with respect to the size. Then we modify the starting times for operations in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ such that the operations' starting times satisfy the first condition required by an FSS schedule.

- If $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ contains only small operations, P is less than $\alpha^2+\delta < \alpha^2+2$ as the interval I'_t has a length of α^2 and there might exist one unfinishing small operation (of size less than $\delta < 1$) in the end of the interval. If we delay the sorted operations to start at $\lceil (t+1)\cdot(\alpha^2+2) - P\rceil \in \left[t\cdot(\alpha^2+2),\ (t+1)\cdot(\alpha^2+2)\right) = I_t$, the first small operation starts processing at some integer time point and the maximum finishing time of all operations is at most $(t+1)\cdot(\alpha^2+2)$. This shifting strategy consumes at most one time unit.
- If $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ contains only big operations, we delay the starting time of the first big operation to the nearest integer time point. As each big operation has an integer size, all the following big operations also start at the integer time points. This delaying strategy consumes at most one time unit.
- If $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ contains both small and big operations, the total size of the small operations is at most α^2. We combine the previous two shifting strategies together, which consumes at most two time units. Refer to the bottom two diagrams in Fig. 1.

As we rearrange the operations in $\mathcal{O}^{\tilde{\pi}}_{\ell,j,t}$ and also delay some operations, the starting time for some A-operation may be delayed by at most α^2+2 time units whereas some B-operation may start earlier by at most α^2+2 time units. If these A- and B-operations belong to the same job, the schedule's feasibility will be destroyed. To restore the feasibility, we delay the machines in the second stage by $2\cdot(\alpha^2+2)$. Refer to Fig. 2. Denote the resultant schedule as π'. We can observe that π' is a feasible schedule and actually an FSS schedule. □

3.2. Obtaining an FSS schedule by MILP

According to Theorem 1, with some cost we switch to consider FSS schedules with makespan at most $(\alpha^2+2)^2$ if there is a feasible schedule with makespan at most α^4. In other words, we only need to consider the intervals $I_t = \left[t\cdot(\alpha^2+2),\ (t+1)\cdot(\alpha^2+2)\right),\ t \in \{0,1,\ldots,\alpha^2+1\}$.

3.2.1. Configuration of a flow-shop

We define a *configuration* for each flow-shop such that all flow-shops with the same configuration form a group. Therefore, each FSS schedule corresponds to a distribution of the distinct configurations for the m input flow-shops. More precisely, a distribution of the distinct configurations for the m input flow-shops is able to roughly depict an FSS schedule. To find an optimal FSS

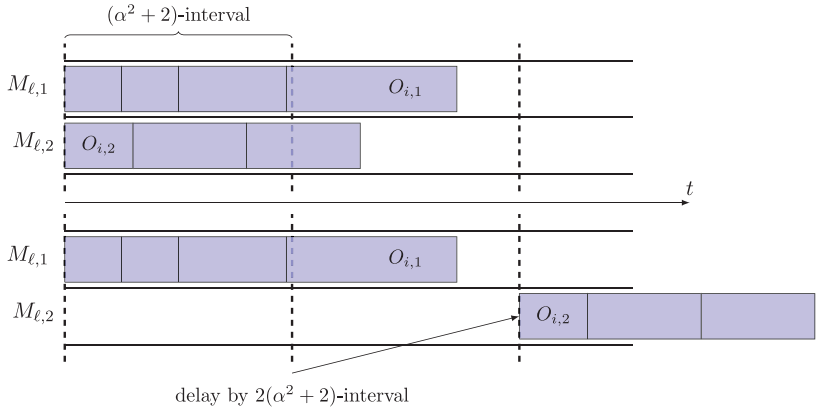

Fig. 2. Illustration of the infeasibility caused by the rearrangement of operations in each interval. In the top diagram, $O_{i,2}$ could be started earlier by at most α^2+2 time whereas $O_{i,1}$ could be delayed by at most α^2+2 time. As shown the bottom diagram, delaying the machine $M_{\ell,2}$ by $2\cdot(\alpha^2+2)$ time relative to the machine $M_{\ell,1}$ will address the conflict.

schedule, instead of exploring all possible FSS schedules, we only need to consider all possible distributions of different configurations of the m flow-shops by constructing a mixed integer linear program (MILP). By solving this MILP, we are able to identify a near-optimal FSS schedule, i.e., an FSS schedule with an almost smallest makespan.

Given an FSS schedule π, we fix a flow-shop, say F_ℓ. Let $L_{j,t},\ j\in\{1,2\},\ t\in\{0,1,2,\ldots,\alpha^2+1\}$, be the ceiling of the total processing time of small operations in $\mathcal{O}_{\ell,j,t}$. Note that the value of each $L_{j,t}$ is at most α^2+2, the length of I_t, according to Theorem 1. More specifically, $L_{j,t}\in\{0,1,2,\ldots,\alpha^2+2\}$. Define $n_{t,k,x,y}$ to be the number of jobs in $\mathcal{J}(x,y)$ with the A-operation starting in I_t and B-operation starting in I_k. The *configuration* for the flow-shop is defined as a tuple

$$(L_{j,t}, n_{t,k,x,y})_{j,t,k,x,y}, \tag{3}$$

where $j\in\{1,2\}, t<k\in\{0,1,2,\ldots,\alpha^2+1\}, x,y\in\{0,1,2,\ldots,\alpha^4\}$, $x+y>0$.

Lemma 1. *A flow-shop has at most* $(\alpha^2+2)^{2(\alpha^2+2)+(\alpha^2+1)^2\cdot(\alpha^4+1)^2}$ *distinct configurations.*

Proof. Recall that we always round up the total processing time of small operations in $\mathcal{O}_{j,t}$ to the nearest integer. Here, $L_{j,t}$ has at most α^2+2 values. As each interval I_t has the uniform size α^2+2 and each big operation has size at least one, the number of big operations in the interval I_t is upper bounded by α^2+2 and, thus, $n_{t,k,x,y}$ is also upper bounded by α^2+2. To summarize, the number of distinct configurations for one flow-shop is at most

$$(\alpha^2+2)^{2(\alpha^2+2)+(\alpha^2+1)^2\cdot(\alpha^4+1)^2},$$

where the exponent is due to the fact that a configuration (refer to Eq. (3)) contains at most $2(\alpha^2+2)$ components in the form of $L_{j,t}$ and $(\alpha^2+1)^2\cdot(\alpha^4+1)^2$ components in form of $n_{t,k,x,y}$. □

We can observe that every FSS schedule restricted on one flow-shop is associated with a configuration for this flow-shop, and we say such a configuration is *feasible*. From a configuration $c=(L_{j,t}, n_{t,k,x,y})_{j,t,k,x,y}$, we know during which time interval each big operation of a job (if it exists) starts processing and we also know the total processing time of small operations that start processing in each time interval on each machine. However, a configuration does not provide the information about the size of each small operation. Therefore, the feasibility of a configuration requires that there is enough space for all operations and that the processing time interval for two big operations of the same job cannot overlap. Indeed, we show how to transform a feasible configuration to an FSS schedule except for the small operations, which is named as an *almost* FSS schedule.

Lemma 2. *It takes* $O\big((\alpha^2+2)^{2(\alpha^2+2)}\big)$ *time to check the feasibility of a configuration and meanwhile convert the feasible configuration to an almost FSS schedule.*

Proof. Given a configuration of a flow-shop, we can obtain the set of big operations assigned to start processing on each machine M_j during each time interval I_t. Recall that $\mathcal{O}_{j,t}$ denotes the set of all operations that starts processing on machine M_j during I_t. For each time interval I_t, we first treat all small operations as an entirety (as we only know the rounded total processing time of small operations, not the size of each small operation), then sort all big operations in $\mathcal{O}_{j,t}$ non-decreasingly with respect to size, and finally concatenate the big operations following the block of all small operations. This results in consecutive sub-schedules of all operations in $\mathcal{O}_{j,t}$. Shifting the obtained sub-schedule in the time interval I_t such that the sub-schedule starts at some integer time point will satisfy the first condition required by an FSS schedule. As the length of I_t is α^2+2, there are at most α^2+2 shifting strategies for each time interval on one machine. Shifting each sub-schedule independently may result in an infeasible schedule as two consecutive sub-schedules may overlap.

By exhausting all possible shifted sub-schedules and concatenating the non-overlapping sub-schedules into one schedule on the flow-shop, we can check whether there is enough space for all operations. As the information $\{n_{t,k,x,y}\}_{t,k,x,y}$, $x,y>0$ is included in the configuration, we know all jobs with two big operations on the flow-shop, and, thus, we are able to check the feasibility for such jobs easily. To summarize, if the configuration is feasible, the resultant schedule is an almost FSS schedule, where only small operations may cause infeasibility.

As each interval I_t has the uniform size α^2+2 and each big operation has size at least one, the number of big operations in the interval I_t is upper bounded by α^2+2. Sorting big operations in each interval takes $O(\alpha^2)$ time by some linear sorting algorithm, say RadixSort. As there are α^2+2 (α^2+2)-intervals and the flow-shop consists of two machines, the total time of sorting big operations takes $O(\alpha^4)$ time. In addition, exhausting all possible shifted sub-schedules takes $O\big((\alpha^2+2)^{2(\alpha^2+2)}\big)$. Therefore, the time to check the feasibility of the given configuration is $O\big((\alpha^2+2)^{2(\alpha^2+2)}\big)$, which is a constant depending on α. □

Lemmas 1 and 2 guarantee that checking the feasibility of all possible configurations for one flow-shop takes a constant time that only depends on α. Without loss of generality, we only consider the feasible configurations in the following context.

3.2.2. Mixed integer linear program

By Lemma 2, a feasible configuration can be converted into an almost FSS schedule where only the small operations may cause the infeasibility. The objective of the mixed integer linear program (MILP) is to find a distribution of feasible configurations for the m flow-shops and figure out how to assign the remaining small operations appropriately to achieve an FSS schedule.

Let $\mathcal{C}$ be the set of all feasible configurations of a flow-shop. Define $Z^{(c)}$ to be the integer variable to indicate the number flow-shops with the configuration $c \in \mathcal{C}$. For any FSS schedule, there is a distribution of configurations $\{Z^{(c)} \mid c \in \mathcal{C}\}$ for the m input flow-shops. We have

$$\sum_{c\in\mathcal{C}} Z^{(c)} = m, \tag{4}$$

$$Z^{(c)} \in \mathbb{Z}^{\geq 0}, \quad \forall\ c \in \mathcal{C}.$$

From the almost FSS schedule obtained from a configuration c by Lemma 2, we can calculate the number of jobs in $\mathcal{J}(x,y)$, $x, y \in \{1,2,\ldots,\alpha^4\}$, such that under the configuration c, the big A-operation finishes in I_t and the big B-operation starts in I_k, $0 \leq t < k \leq \alpha^2+1$. Let $bb^{(c)}_{x,y,t,k}$ denote this number. In any FSS schedule, the jobs with two big operations have to satisfy Eq. (5).

$$\sum_{c\in\mathcal{C}} \sum_{0\leq t<k\leq\alpha^2+1} bb^{(c)}_{x,y,t,k} \cdot Z^{(c)} = |\mathcal{J}(x,y)|, \quad \forall\ x,y \in \{1,2,\ldots,\alpha^4\}. \tag{5}$$

Similarly, from the almost FSS schedule obtained from a configuration c, we can calculate $bs^{(c)}_{x,0,t}$ and $sb^{(c)}_{0,y,k}$, where $bs^{(c)}_{x,0,t}$ denotes the number of jobs in $\mathcal{J}(x,0)$, $x \in \{1,2,\ldots,\alpha^4\}$, such that under the configuration c, the big A-operation finishes in I_t, $0 \leq t \leq \alpha^2+1$; $sb^{(c)}_{0,y,k}$ denotes the number of jobs in $\mathcal{J}(0,y)$, $y \in \{1,2,\ldots,\alpha^4\}$, such that under the configuration c the big B-operation starts in I_k, $0 \leq k \leq \alpha^2+1$. Then in any FSS schedule, the jobs with one big operation (either in the first stage or the second stage) have to satisfy (Eqs. (6) and (7)).

$$\sum_{c\in\mathcal{C}} \sum_{0\leq t\leq\alpha^2+1} bs^{(c)}_{x,0,t} \cdot Z^{(c)} = |\mathcal{J}(x,0)|, \quad \forall\ x \in \{1,2,\ldots,\alpha^4\}. \tag{6}$$

$$\sum_{c\in\mathcal{C}} \sum_{0\leq k\leq\alpha^2+1} sb^{(c)}_{0,y,k} \cdot Z^{(c)} = |\mathcal{J}(0,y)|, \quad \forall\ y \in \{1,2,\ldots,\alpha^4\}. \tag{7}$$

Suppose L_j, $j \in \{1,2\}$ is the total processing of small operations on the jth stage for jobs in $\mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$. Recall that in the definition of the configuration for a flow-shop, $L_{j,t}$, $j \in \{1,2\}$, $t \in \{0,1,2,\ldots,\alpha^2+1\}$ is the ceiling of the total processing time of small operations in $\mathcal{O}_{j,t}$. Therefore, in any FSS schedule, the total processing time of small operations needs to be bounded by Eq. (8) such that there is enough space for small operations.

$$\sum_{c\in\mathcal{C}} \sum_{0\leq t\leq\alpha^2+1} L^{(c)}_{j,t} \cdot Z^{(c)} \geq L_j, \quad j \in \{1,2\}. \tag{8}$$

Define $u^{(c)}_{i,t,k}$, $t < k$ be a binary variable, $t, k \in \{0,1,\ldots,\alpha^2+1\}$. Here $u^{(c)}_{i,t,k} = 1$ indicates the job $J_i \in \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ is assigned to finish its A-operation in I_t and start its B-operation in I_k with $k > t$. As each job $J_i \in \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ is assigned exactly once in any FSS schedule, then Eq. (9) needs to be satisfied. Eq. (10) is used to maintain the consistency between the variables $u^{(c)}_{i,t,k}$ and $Z^{(c)}$ for the same configuration c. In particular, no $u^{(c)}_{i,t,k}$ has a positive value when $Z^{(c)} = 0$.

$$\sum_{c\in\mathcal{C}} \sum_{0\leq t<k\leq\alpha^2+1} u^{(c)}_{i,t,k} = 1,$$
$$\forall\ J_i \in \mathcal{J}(x,y), x \cdot y = 0,\ x,y \in \{0,1,2,\ldots,\alpha^4\}. \tag{9}$$

$$\sum_{0\leq t<k\leq\alpha^2+1} u^{(c)}_{i,t,k} \leq Z^{(c)}, \quad \forall c \in \mathcal{C}. \tag{10}$$

$$u^{(c)}_{i,t,k} \in \{0,1\}.$$

The assignments for jobs in $\mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ affect the total processing time of small operations on each stage during the interval I_t, $0 \leq t \leq \alpha^2+1$. Recall that the variable $L^{(c)}_{j,t}$ is always rounded up to the nearest integer. Therefore, in any FSS schedule, the total processing time of small operations needs to be estimated by (Eqs. (11) and (12)) such that there is enough space for small operations in each time interval. Meanwhile, the constraints (11) and (12) guarantee that if all operations in the interval are sorted non-decreasingly with respect to the size, the first small operation in each interval can be shifted to start at some integer time point.

$$L^{(c)}_{1,t} - 1 < \sum_{y\in\{1,2,\ldots,\alpha^4\}} \sum_{J_i\in\mathcal{J}(0,0)\cup\mathcal{J}(0,y)} \sum_{t<k\leq\alpha^2+1} u^{(c)}_{i,t,k} \cdot a_i \leq L^{(c)}_{1,t}, \tag{11}$$

$$L^{(c)}_{2,k} - 1 < \sum_{x\in\{1,2,\ldots,\alpha^4\}} \sum_{J_i\in\mathcal{J}(0,0)\cup\mathcal{J}(x,0)} \sum_{0\leq t<k} u^{(c)}_{i,t,k} \cdot b_i \leq L^{(c)}_{2,k}, \tag{12}$$

$$t, k \in \{0,1,\ldots,\alpha^2+1\}, c \in \mathcal{C}.$$

The assignments for jobs in $\mathcal{J}(x,0) \cup \mathcal{J}(0,y)$ also affect the distribution of big operations during the interval I_t, $0 \leq t \leq \alpha^2+1$. To keep the consistency with configurations, we have (Eqs. (13) and (14))

$$\sum_{J_i\in\mathcal{J}(x,0)} \sum_{t<k\leq\alpha^2+1} u^{(c)}_{i,t,k} = bs^{(c)}_{x,0,t} \cdot Z^{(c)}, \tag{13}$$

$$\sum_{J_i\in\mathcal{J}(0,y)} \sum_{0\leq t<k} u^{(c)}_{i,t,k} = sb^{(c)}_{0,y,k} \cdot Z^{(c)}, \tag{14}$$

$$t, k \in \{0,1,\ldots,\alpha^2+1\}, x, y \in \{1,2,\ldots,\alpha^4\}, c \in \mathcal{C}.$$

The constraints (4)–(14) indeed depict the FSS schedules. In particular, for every job, we require that its B-operation cannot start until the A-operation finishes in the previous interval. In addition, the constraints (6)–(8) requires the FSS schedule to remain consistent with the feasible configurations on the m flow-shops. The constraints (9)–(14) find the assignments $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ for jobs each of which has at least one small operations. The constraints (11) and (12) in the MILP imply every small operation starts and finishes in the same interval I_t for some t.

The MILP (4)–(14) contains variables

$$\left\{Z^{(c)}, u^{(c)}_{i,t,k}\right\}_{i,t,k,c},$$

where $J_i \in \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$. In fact, they are all integer variables. We relax $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ to non-negative fractional variables for each job J_i with at least one small operation. Then the integer variables are $\{Z^{(c)}\}_c$, whose cardinality is upper bounded by $|\mathcal{C}|$, a constant only depending on α according to Lemma 1. Therefore, the MILP has a constant number of integer variables, a polynomial number of continuous variables, and a polynomial number of constraints. Lenstra's algorithm (Lenstra Jr, 1983) can be employed to solve such MILP in polynomial time, as its time complexity is exponential in the number of integer variables but polynomial in the number of continuous variables, polynomial in the number of constraints, and polynomial in the logarithms of the coefficients.

Suppose $\left\{Z^{(c)}, u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ is a basic feasible solution to the relaxed MILP. Let $\mathcal{C}^+ = \left\{c \in \mathcal{C} \mid Z^{(c)} > 0\right\}$. We partition the m flow-shops into $|\mathcal{C}^+|$ groups such that flow-shops in the same group have the same configuration. Consider one flow-shop F in the group with the configuration $c \in \mathcal{C}^+$. Recall that for a job J_i with at least one small operation, $\left\{u^{(c)}_{i,t,k}\right\}_{t,k,c}$ determines the indexes of time intervals, where the A-operation of J_i finishes processing and the B-operation of J_i starts processing under the configuration c.

Lemma 3. *There are at most a constant number of jobs in $\mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ such that the corresponding variable $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ have fractional positive values. In other words, most $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ have integer values.*

Proof. Suppose $\Gamma = \left|\bigcup_{x,y} \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)\right|$. Our relaxation of $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ replaces the constraints $u^{(c)}_{i,t,k} \in \{0,1\}$ with $u^{(c)}_{i,t,k} \geq 0$. Then the total number of variables is at most

$$|\mathcal{C}| + (\alpha^2+1)^2 \cdot |\mathcal{C}| \cdot \Gamma.$$

Let γ be the total number of constraints in our MILP. There are 1, α^8, α^4, α^4, 2, Γ, $|\mathcal{C}|$, $2(\alpha^2+2)|\mathcal{C}|$, $2(\alpha^2+2)|\mathcal{C}|$, $(\alpha^2+2)\alpha^4|\mathcal{C}|$, $(\alpha^2+2)\alpha^4|\mathcal{C}|$ constraints in Eqs. (4)–(14), respectively. It is easy to check that γ is upper bounded by

$$3 + \alpha^8 + 2\alpha^4 + 4(\alpha^2+2)|\mathcal{C}| + 2\alpha^4(\alpha^2+2)|\mathcal{C}| + |\mathcal{C}| + \Gamma,$$

which is much less than the total number of variables when Γ is a function of n and n is large enough, as $|\mathcal{C}|$ and α are constants.

It follows that the basic feasible solution to this MILP has at most γ positive values. As $|\mathcal{C}| \leq (\alpha^2+2)^{2(\alpha^2+2)+(\alpha^2+1)^2\cdot(\alpha^4+1)^2}$ by Lemma 1, we can further upper bound γ by $\Gamma + 3^{26\alpha^{12}}\alpha^{50\alpha^{12}}$, which is a loose bound holding for any $\alpha \geq 1$. Let $\phi = 3^{26\alpha^{12}}\alpha^{50\alpha^{12}}$.

Note that for every job $J_i \in \bigcup_{x,y} \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$, if $u^{(c)}_{i,t,k}$ has a positive fractional value, then there must be another distinct combination (t', k', c') such that $u^{(c')}_{i,t',k'}$ also has a positive fractional value. Suppose the total number of positive fractional values in the basic feasible solution is γ'. Let Γ_1 denote the number of jobs in $\mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ for each of which there is an associated variable having value 1, that is, the assignment is determined in the basic solution; and let $\Gamma_2 = \Gamma - \Gamma_1$. It follows that $\Gamma_2 \leq \gamma'/2$, and thus $\Gamma_1 \geq \Gamma - \gamma'/2$. Therefore, the total number of positive values in the basic solution is at least $\Gamma - \gamma'/2 + \gamma' = \Gamma + \gamma'/2$. From $\Gamma + \gamma'/2 \leq \gamma \leq \Gamma + \phi$, we have $\gamma' \leq 2\phi$ and, thus, we conclude that

$$\Gamma_2 \leq \gamma'/2 \leq \phi,$$

which is a constant only depending on α. □

Lemma 4. *Any basic feasible solution to the relaxed MILP (4)–(14) can be used to construct an FSS schedule for the corresponding restricted-type mP2FS instance with makespan at most $(\alpha^2+2)^2+2$.*

Proof. Lemma 2 shows how to convert the feasible configuration c into an FSS schedule except for the small operations. The feasible configuration occupies at most (α^2+2) time intervals each having length α^2+2. Lemma 3 demonstrates for the jobs with at least one small operations, we are able to assign most of them explicitly to the m flow-shops.

Consider a job $J_i \in \bigcup_{x,y} \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ such that $u^{(c)}_{i,t,k}$ has a fractional value. If $J_i \in \bigcup_{x,y>0} \mathcal{J}(x,0) \cup \mathcal{J}(0,y)$, its big operation has been scheduled appropriately by Lemma 2. We can insert the remaining small A-operation (B-operation, respectively) to the front (end, respectively) of the flow-shop to which the corresponding big B-operation (A-operation, respectively) was assigned according to the configuration c. Such assignment strategies maintain the second property of the FSS schedule. If $J_i \in \mathcal{J}(0,0)$, we execute the two small operations sequentially and append jobs to the end of the current schedule arbitrarily on any flow-shop. It is easy to check that the resultant schedule is an FSS schedule.

Each small operation has size at most δ. By Lemma 3, there are at most ϕ jobs in $\bigcup_{x,y} \mathcal{J}(x,0) \cup \mathcal{J}(0,y) \cup \mathcal{J}(0,0)$ such that the corresponding variable $\left\{u^{(c)}_{i,t,k}\right\}_{i,t,k,c}$ have fractional positive values. For each job from $\mathcal{J}(x,0) \cup \mathcal{J}(0,y)$, the above assignment strategy increases the makespan by at most δ. The amplitude becomes 2δ when the job is from $\mathcal{J}(0,0)$. Therefore, the makespan increases by at most $2 \cdot \delta \cdot \phi$. Let

$$\delta = \frac{1}{\phi}. \tag{15}$$

The total makespan of the resultant feasible schedule is at most

$$(\alpha^2+2)^2 + 2.$$

□

Combining Theorem 1 and Lemma 4, we have the following theorem.

Theorem 2. *For the restricted-type mP2FS instance, if there exists a feasible schedule with makespan at most α^4, the corresponding MILP (4)–(14) admits at least one feasible solution. In addition, we can find an FSS schedule with makespan at most $(\alpha^2+2)^2+2$.*

3.3. The PTAS

In this section, we first show how to transform an arbitrary mP2FS instance into a restricted instance which is introduced in Section 3.1. Then we present a PTAS for the general mP2FS problem when m is part of the input.

For any $\epsilon \in (0, 1)$, set $\alpha = \lceil \frac{43}{\epsilon} \rceil$. Let C_g be a guess of the $C_{\max}(\pi^*)$. For each operation of a job $J_i \in \mathcal{J}$, say A_i, we scale a_i as follows (b_i can be scaled similarly):

$$a'_i = \begin{cases} a_i \cdot \frac{\alpha^4}{C_g}, & \text{if } a_i \cdot \frac{\alpha^4}{C_g} < \delta; \\ \lceil a_i \cdot \frac{\alpha^4}{C_g} \rceil, & \text{otherwise.} \end{cases} \tag{16}$$

By such a scaling method, an arbitrary mP2FS instance is transformed into a restricted-type instance introduced in Section 3.1. Denote the scaled job set as $\mathcal{J}' = \{J'_1, J'_2, \ldots, J'_n\}$ with $J'_i = (A'_i, B'_i)$.

Then construct the MILP for the scaled restricted-type instance $\mathcal{J}'$. If the MILP does not admit a feasible solution, we say that the guess C_g is *incorrect*. Otherwise, we say the guess C_g is *correct*. Use Lemma 4 to transform the feasible solution into a feasible schedule. Then enlarge the sizes of operations by a factor $\frac{(\alpha+1)C_g}{\alpha^5}$. Suppose the new instance is $\mathcal{J}'' = \{J''_1, J''_2, \ldots, J''_n\}$ with $J''_i = (A''_i, B''_i)$. Then

$$a''_i = a'_i \cdot \frac{(\alpha+1)C_g}{\alpha^5} \geq a_i \cdot \frac{\alpha^4}{C_g} \cdot \frac{(\alpha+1)C_g}{\alpha^5} = a_i \cdot \frac{\alpha+1}{\alpha} \geq a_i.$$

Similarly, we have $b''_i \geq b_i$. This indicates the feasible schedule for $\mathcal{J}''$ is also a feasible schedule for the original instance $\mathcal{J}$.

Lemma 5. *If $C_g \geq C_{\max}(\pi^*)$, then our guess is correct and a feasible schedule can be produced with makespan at most $\frac{\alpha+21}{\alpha} \cdot C_g$.*

Proof. If $C_g \geq C_{\max}(\pi^*)$, there exists a feasible schedule with makespan C_g for $\mathcal{J}$, which implies the scaled job set $\mathcal{J}'$ admits a feasible schedule with makespan at most α^4. By Theorem 2, we are able to find an FSS schedule with makespan at most $(\alpha^2+2)^2+2$.

Then we can obtain a feasible schedule for the enlarged job set $\mathcal{J}''$ with makespan at most

$$\begin{aligned}&\left((\alpha^2+2)^2+2\right)\cdot\frac{(\alpha+1)C_g}{\alpha^5}\\ =&\left(\alpha^5+\alpha^4+4\alpha^3+4\alpha^2+6\alpha+6\right)\cdot\frac{C_g}{\alpha^5}\\ \le&\frac{\alpha+21}{\alpha}\cdot C_g,\end{aligned}$$

where the last inequality is because $\alpha\ge 1$. □

Let $P_i=a_i+b_i$ denote the total processing time of the job J_i over both machines, and assume without loss of generality that $P_1\ge P_2\ge\cdots\ge P_n$. Define $P=\sum_{i=1}^{n}P_i$ as the summed processing time for all jobs. The optimal makespan $C_{\max}(\pi^*)$ can be bounded by Lemma 6, which is shown in Tong et al. (2018). Our PTAS just guesses the makespan of the optimal schedule by bisecting the interval $\left[\max\left\{\frac{P}{2m},P_1\right\},\frac{P}{m}+P_1\right]$ recursively until the a near-optimal makespan is found. Then Lemma 5 can be used to find a feasible schedule with this near-optimal makespan. Owing to the searching manner of the guess, we name our algorithm as BinarySearch. The details are given in Algorithm 1.

Algorithm 1 Binary Search

Input: m parallel 2-stage flow-shops, $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$;
Output: a feasible schedule π with makespan at most $(1+\epsilon)C_{\max}(\pi^*)$.

```
Upper = P/m + P_1;
Lower = max{P/(2m), P_1};
while ((α+1)/α)·Lower < Upper do
    Mid = (1/2)(Low + Upper);
    if Mid is incorrect then
        Lower = Mid;
    else
        Upper = Mid;
    end if
end while
Set the guess C_g = Upper;
Produce a feasible schedule by Lemma 5;
return the final schedule denoted as π.
```

Lemma 6. *Tong et al. (2018) We have the following upper and lower bounds on* $C_{\max}(\pi^*)$*:*

$$\max\left\{\frac{P}{2m},P_1\right\}\le C_{\max}(\pi^*)\le\frac{P}{m}+P_1.$$

Theorem 3. *Our algorithm* BinarySearch *is a PTAS. That is, it outputs a schedule with makespan at most* $(1+\epsilon)C_{\max}(\pi^*)$ *in polynomial time.*

Proof. First, we show $C_{\max}(\pi^*)\in$ [Lower, Upper] is a loop invariant. Initially, Lemma 6 (Tong et al., 2018) bounds the $C_{\max}(\pi^*)$ within [Lower, Upper]. During the execution of the loop, if Mid is an incorrect guess of the makespan, Lemma 5 guarantees Mid $< C_{\max}(\pi^*)$; if Mid is a correct guess of the makespan, Lemma 5 implies Mid $\ge C_{\max}(\pi^*)$. Therefore, the update of Low and Upper in the if–else statement maintains $C_{\max}(\pi^*)\in$ [Lower, Upper].

As the initial Upper $=\frac{P}{m}+P_1$ and Lower $=\max\left\{\frac{P}{2m},P_1\right\}$, we have Upper $\le 4\cdot$ Lower or Upper/Lower ≤ 4. The difference (Upper − Lower) decreases by half each iteration. Thus, the number of iterations of the while loop in BinarySearch is $O(\log\alpha)$.

When the loop terminates, we have Upper $\le\frac{\alpha+1}{\alpha}$Lower and the loop invariant $C_{\max}(\pi^*)\in$ [Lower, Upper] still holds. Of course, Upper is a valid guess of the makespan. According to Lemma 5, we are able to find a feasible schedule π with makespan at most

$$\begin{aligned}C_{\max}(\pi)&\le\frac{\alpha+21}{\alpha}\cdot C_g=\frac{\alpha+21}{\alpha}\cdot\text{Upper}\\ &\le\frac{\alpha+21}{\alpha}\cdot\frac{\alpha+1}{\alpha}\cdot\text{Lower}\\ &\le\left(1+\frac{43}{\alpha}\right)\cdot C_{\max}(\pi^*)\\ &\le(1+\epsilon)\cdot C_{\max}(\pi^*),\end{aligned}$$

where the last inequality comes from the fact that $\alpha=\lceil\frac{43}{\epsilon}\rceil$. □

4. Conclusions

We have considered the m parallel two-stage flow-shops (mP2FS) problem when m is part of the input. Our main contribution is a polynomial-time approximation scheme (PTAS), which partially addresses the open question proposed in Tong et al. (2018). As the mP2FS problem is strongly NP-hard when m is part of the input, our PTAS is the best possible approximation unless P = NP. Consider the m parallel d-stage flow-shops (mPdFS) problem. It is interesting to investigate whether the mPdFS problem also admits a PTAS if d is a fixed constant. When d is part of the input, the mPkFS problem is APX-hard following the APX-hardness of the classic d-stage flow-shop problem. Another open problem is whether the mPdFS problem admits an approximation algorithm with a constant ratio when d is part of the input.

Acknowledgments

Dong was supported by the National Natural Science Foundation of China Grant no. 11971435 and the Zhejiang Provincial Natural Science Foundation Grant no. LY18A010029. Luo was supported by the MOE (Ministry of Education in China) Project of Humanities and Social Sciences under Grant no. 18YJC630114 and the Shaanxi Province Natural Science Foundation Grants no. 2019JQ-154 and no. 2019JQ-079. Luo was also supported by the Youth Innovation Team of Shaanxi Universities. Tong was partially supported by the National Natural Science Foundation of China under Grant no. 71701162 when Tong was visiting the Northwest University, China in the Summer 2018.

References

Chen, B. (1995). Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *The Journal of the Operational Research Society, 46*, 234–244.
Conway, R. W., Maxwell, W. L., & Miller, L. W. (1967). *Theory of scheduling*. Reading, Mass: Addison-Wesley.
Dong, J., Hu, J., Kovalyov, M. Y., Lin, G., Luo, T., Tong, W., … Xu, Y. (2017a). Corrigendum to an FPTAS for the parallel two-stage flowshop problem. *Theoretical Computer Science, 687*, 93–94. [Theoretical Computer Science 657 (2017) 64–72]
Dong, J., Tong, W., Luo, T., Wang, X., Hu, J., Xu, Y., & Lin, G. (2017b). An FPTAS for the parallel two-stage flowshop problem. *Theoretical Computer Science, 657*, 64–72.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY, USA: W. H. Freeman & Co.
Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117–129.
Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal, 45*, 1563–1581.
Gupta, J. N. D. (1988). Two-stage, hybrid flowshop scheduling problem. *The Journal of the Operational Research Society, 39*, 359–364.
Gupta, J. N. D., Hariri, A. M. A., & Potts, C. N. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research, 69*, 171–191.
Gupta, J. N. D., & Tunc, E. A. (1991). Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research, 29*, 1489–1502.
Hall, L. A. (1998). Approximability of flow shop scheduling. *Mathematical Programming, 82*, 175–190.
He, D. W., Kusiak, A., & Artiba, A. (1996). A scheduling problem in glass manufacturing. *IIE Transactions, 28*, 129–139.
Hochbaum, D. S., & Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM, 34*, 144–162.

Hoogeveen, J. A., Lenstra, J. K., & Veltman, B. (1996). Preemptive scheduling in a two-stage multiprocessor flow shop is np-hard. *European Journal of Operational Research, 89*, 172–175.
Jansen, K., & Sviridenko, M. I. (2000). Polynomial time approximation schemes for the multiprocessor open and flow shop scheduling problem. In *Stacs*. In *LNCS: 1770* (pp. 455–465).
Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly, 1*, 61–68.
Kovalyov, M. Y. (1985). Efficient epsilon-approximation algorithm for minimizing the makespan in a parallel two-stage system. *Vesti Academii Navuk Belaruskai SSR. Seria Phizika-Matematichnikh Navuk*.
Lee, C.-Y., & Vairaktarakis, G. L. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters, 16*, 149–158.
Lenstra Jr, H. W. (1983). Integer programming with a fixed number of variables. *Mathematics of operations research, 8*(4), 538–548.
Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*, 1–18.
Sahni, S. K. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM, 23*, 116–127.
Schuurman, P., & Woeginger, G. J. (2000). A polynomial time approximation scheme for the two-stage multiprocessor flow shop problem. *Theoretical Computer Science, 237*, 105–122.
Tong, W., Miyano, E., Goebel, R., & Lin, G. (2018). An approximation scheme for minimizing the makespan of the parallel identical multi-stage flow-shops. *Theoretical Computer Science, 734*, 24–31.
Vairaktarakis, G., & Elhafsi, M. (2000). The use of flowlines to simplify routing complexity in two-stage flowshops. *IIE Transactions, 32*, 687–699.
Wang, H. (2005). Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems, 22*, 78–85.
Williamson, D. P., Hall, L. A., Hoogeveen, J. A., Hurkens, C. A. J., Lenstra, J. K., Sevastjánov, S. V., & Shmoys, D. B. (1997). Short shop schedules. *Operations Research, 45*, 288–294.
Wu, G., & Chen, J. (2018). Approximation algorithms on multiple two-stage flow-shops. In *Cocoon* (pp. 713–725).
Wu, G., Chen, J., & Wang, J. (2017). On approximation algorithms for two-stage scheduling problems. In *Faw* (pp. 241–253).
Wu, G., & Wang, J. (2017). Approximation algorithms for scheduling multiple two-stage flowshops. In *Cocoon* (pp. 516–528).
Zhang, X., & Velde, S. (2012). Approximation algorithms for the parallel flow shop problem. *European Journal of Operational Research, 216*, 544–552.