Journal of Combinatorial Optimization (2021) 41:28–42
https://doi.org/10.1007/s10878-020-00657-2

Improved approximation algorithms for two-stage flexible flow shop scheduling

Anzhen Peng[1] · Longcheng Liu[1] · Weifeng Lin[1]

Accepted: 29 September 2020 / Published online: 23 October 2020

Abstract
A two-stage flexible flow shop scheduling is a manufacturing infrastructure designed to process a set of jobs, in which a single machine is available at the first stage and m parallel machines are available at the second stage. At the second stage, each task can be processed by multiple parallel machines. The objective is to minimize the maximum job completion time, i.e., the makespan. Sun et al. (J Softw 25:298–313, 2014) presented an $O(n \log n)$-time 3-approximation algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ under some special conditions. Zhang et al. (J Comb Optim 39:1–14, 2020) presented a 2.5-approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$ and a 2.67-approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$, which both run in linear time. In this paper, we achieved following improved results: for $F2(1, P2) \mid line_i \mid C_{\max}$, we present an $O(n \log n)$-time 2.25-approximation algorithm, for $F2(1, P3) \mid line_i \mid C_{\max}$, we present an $O(n \log n)$-time 7/3-approximation algorithm, for $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption $\min_{1 \le i \le n} \{p_{1i}\} \ge \max_{1 \le i \le n} \{p_{2i}\}$, we present a linear time optimal algorithm.

Keywords Scheduling · Two-stage flow shop · Approximation algorithm · Optimal algorithm

1 Introduction

We study the following two-stage flexible flow shop scheduling problem, denoted as $F2(1, Pm) \mid size_i \mid C_{\max}$ or $F2(1, Pm) \mid line_i \mid C_{\max}$ in the three field notation (Graham et al. 1979). Given a job set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a two-stage flow

This research is supported by the Fundamental Research Funds for the Central Universities (Grant No. 20720190068).

✉ Longcheng Liu
longchengliu@xmu.edu.cn

[1] School of Mathematical Sciences, Xiamen University, Xiamen 361005, People's Republic of China

shop where there are a single machine at the first stage and m parallel machines at the second stage. Each job has to be processed non-preemptively on the only one machine of the first stage. After it has been finished in the first stage, it has to be processed non-preemptively on one or multiple parallel machines of the second stage. Each job J_i can be charactered as $(p_{1i}, p_{2i}, size_i/line_i)$, where p_{1i} means the processing time at the first stage, p_{2i} means the processing time at the second stage and $size_i$ or $line_i$ means the number of required parallel machines at the second stage, particularly, $line_i$ represents the contiguous machine assignments at the second stage. The objective is to minimize the maximum job completion time, i.e., the makespan. The scheduling constraint is as usual, that is at every time point, a job can be processed by at most one machine and a machine can process at most one job.

Flexible flow shop scheduling problems have many real-life applications: many discrete manufacturing industries have a flow shop architecture where there is a multi-stage production process with the property that all tasks have to be proceed through the stages in the same order. Flexible flow shop scheduling problems have been studied since the 1970's (Arthanari and Ramamurthy 1971; Salvador 1973) and attract more and more attention from all over the world. For the two-stage flexible flow shop under the condition that each task requires only one machine, and multiple parallel machines are available at every stage, Lee and Vairaktarakis (1994) developed a $2 - \frac{1}{\max\{m_1, m_2\}}$ approximation algorithm, where m_1 and m_2 are the numbers of machines at the first and the second stage. Gupta et al. (2002) studied a generalization of the permutation flow shop problem that combines the scheduling function with planning stage. They presented iterative algorithms based on local search and constructive algorithms based on job insertion techniques. Alisantoso et al. (2003) studied the scheduling of a flexible flow shop for PCB (printed circuit board) manufacturing. They first presented an overview of the flexible flow shop problem and the basic notions of an immune algorithm. And then, they presented an immune algorithm in detail and implemented it to compare with the well know evolutionary algorithms-genetic algorithms, which showed that immune algorithm outperformed genetic algorithms. Lin and Liao (2003) considered a two-stage hybrid flow shop with characteristics of sequence dependent setup time at first stage, dedicated machines at the second stage, and two due dates. They presented a heuristic algorithm to find the near-optimal schedule for the problem. He et al. (2008) considered the two-stage flexible flow shop scheduling problem with m identical parallel machines in stage 1 and only one batch machine in stage 2. They showed the problem is NP-hard in general and proposed corresponding approximation algorithm. Moseley et al. (2011) presented a 12-approximation algorithm for two-stage parallel machine scheduling problem and used it to solve the popular Map-Reduce frame of big data circumstance. Almeder and Hartl (2013) considered a scheduling problem of a real-world production process in the metal-working industry which can be described as an offline stochastic flexible flow-shop problem with limited buffers. They first studied a simplified model and proposed a variable neighbourhood search based solution approach. And then, they applied the solution approach to a real-world case using a detailed discrete-event simulation to evaluate the production plans. Choi and Lee (2013) proposed approximation algorithms for the two-stage flexible flow shop problem with the condition that the processing times of each job are identical at both stages and with a single machine at one stage and m identical machines

at the other stage. For $m = 2$, they presented a 5/4-approximation algorithm, and for $m \geq 3$, they presented a $\frac{\sqrt{1+m^2}+1+m}{2m}$-approximation algorithm. Sun et al. (2014) considered the two-stage scheduling problem with multiple parallel machines at the executing stage, which can be used to solve the transporting and executing stage of massively parallel load tasks on GPU. Especially, they proposed an $O(n \log n)$-time 3-approximation algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumptions that $\sum_{J_i \in \mathcal{J}} p_{1i} > \sum_{J_i \in \mathcal{J}} \{size_i \times p_{2i}\}$ and $\min_{1 \leq i \leq n} \{p_{1i}\} \geq \max_{1 \leq i \leq n} \{p_{2i}\}$. Recently, Zhang et al. (2020) studied the two-stage flexible flow shop scheduling problem with m identical parallel machines at one stage and a single machine at the other stage. They first presented a $(2+\epsilon)$-approximation algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ and a $(2.5+\epsilon)$-approximation algorithm for $F2(1, Pm) \mid line_i \mid C_{\max}$. Then, they presented a 2.5-approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$ and a 2.67-approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$, which both run in linear time.

In this paper, we consider the two-stage flow shop where there are a single machine at the first stage and m parallel machines at the second stage. For $F2(1, P2) \mid line_i \mid C_{\max}$, we present an $O(n \log n)$-time 2.25-approximation algorithm, for $F2(1, P3) \mid line_i \mid C_{\max}$, we present an $O(n \log n)$-time 7/3-approximation algorithm, which improved the approximation results presented in Zhang et al. (2020). For $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption that $\min_{1 \leq i \leq n} \{p_{1i}\} \geq \max_{1 \leq i \leq n} \{p_{2i}\}$, we present a linear time optimal algorithm, which improved the result presented in Sun et al. (2014).

The rest of the paper is organized as follows. In Sect. 2, we present an $O(n \log n)$-time 2.25-approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$. We present in Sect. 3 an $O(n \log n)$-time 7/3-approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$. Section 4 contains a linear time optimal algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption that $\min_{1 \leq i \leq n} \{p_{1i}\} \geq \max_{1 \leq i \leq n} \{p_{2i}\}$. We conclude the paper with some remarks in Sect. 5.

2 A 2.25-approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$

In this section, we consider $F2(1, P2) \mid line_i \mid C_{\max}$: the two-stage flexible flow shop scheduling problem with a machine at the first stage and two machines at the second stage, the objective is to minimize the makespan. It is easy to see that the bounds of $line_i$ case are also hold for the $size_i$ case. Hence, the algorithm we present in this section are applicable for $F2(1, P2) \mid size_i \mid C_{\max}$.

Now, we are ready to present an approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$.

In the first stage, we arrange all the jobs on the machine one by one in random order non-preemptively.

In the second stage, we first group the jobs into two subsets: $\mathcal{J}_1$ denote the set of jobs that requiring only one machine at the second stage and $\mathcal{J}_2$ denote the set of jobs that requiring two machines at the second stage. Secondly, we sort the jobs in $\mathcal{J}_1$ in non-increasing order according to p_{2i}. At last, we first arrange the jobs in $\mathcal{J}_2$ on the two parallel machines one by one non-preemptively, and then arrange the jobs in $\mathcal{J}_1$ according to list scheduling rule where the order of the jobs is priority.

A high-level description of the approximation algorithm is showed in Algorithm 1.

Algorithm 1: Approximation algorithm for $F2\,(1,\,P2)\,|line_i|C_{\max}$

Input: The job list $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a two-stage flexible flow shop scheduling problem, the first stage has only one machine and the second stage has two machines.

Output: A schedule L.

Step 1. Let $\mathcal{J}_1 = \phi, \mathcal{J}_2 = \phi$.

Step 2. For $1 \le i \le n$,

if $line_i = 1$,then add J_i to $\mathcal{J}_1$;

if $line_i = 2$,then add J_i to $\mathcal{J}_2$;

Step 3. The jobs of $\mathcal{J}$ are processed on the machine of the first stage one by one in random order non-preemptively.

Step 4. The jobs of $\mathcal{J}_2$ are processed on the two parallel machines of the second stage one by one in random order non-preemptively.

Step 5. The jobs of $\mathcal{J}_1$ are processed on the two parallel machines of the second stage in non-increasing order according to p_{2i} by using the list scheduling rule.

Theorem 1 *The Algorithm 1 is an $O(n \log n)$-time 2.25-approximation algorithm for $F2\,(1,\,P2)\,|line_i|C_{\max}$.*

Proof Without loss of generality, we assume that the job set $\mathcal{J}_1 = \{J_1, J_2, \ldots, J_k\}$ satisfies $p_{21} \ge p_{22} \ge \cdots \ge p_{2k}$.

Let us consider the following two cases:

Case 1: $p_{21} \ge \sum_{i=2}^{k} p_{2i}$.

From the feature of $F2\,(1,\,P2)\,|line_i|C_{\max}$,we have

$$C^*_{\max} \ge \sum_{J_i \in \mathcal{J}} p_{1i}, \tag{1}$$

$$C^*_{\max} \ge \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{21}, \tag{2}$$

where $C^*_{\max}$ denote the optimal value for $F2\,(1,\,P2)\,|line_i|C_{\max}$.

According to the Algorithm 1 we have (an illustration is showed in Fig. 1)

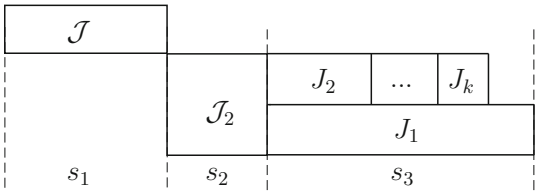

Fig. 1 An illustration of the schedule produced by Algorithm 1 under case 1, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_3 is the processing time of job J_1 on the second stage

$$C_{\max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{21}. \tag{3}$$

Put (1), (2) and (3) together, we have

$$C_{\max}(L) \leq 2C^*_{\max}.$$

Case 2: $p_{21} < \sum_{i=2}^{k} p_{2i}$.

In this case, (1) is also hold.

From the feature of $F2\,(1,\, P2)\,|line_i|C_{\max}$,we have

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}_2} p_{2i} + \frac{1}{2} \sum_{J_i \in \mathcal{J}_1} p_{2i}. \tag{4}$$

There are at least three jobs J_1, J_2, J_3 in the job set $\mathcal{J}_1$ since $p_{21} < \sum_{i=2}^{k} p_{2i}$. Hence we have

$$C^*_{\max} \geq p_{22} + p_{23}. \tag{5}$$

According to the Algorithm 1 we have (an illustration is showed in Fig. 2)

$$C_{\max}(L) \leq \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + \frac{1}{2} \sum_{J_i \in \mathcal{J}_1} p_{2i} + \frac{1}{2} p_{23}. \tag{6}$$

Put (1), (4), (5) and (6) together, we have

$$\begin{aligned} C_{\max}(L) &\leq 2C^*_{\max} + \tfrac{1}{2} \times \tfrac{1}{2} (p_{22} + p_{23}) \\ &\leq 2.25 C^*_{\max}. \end{aligned}$$

Combining cases 1 and 2, we have showed the approximation ratio of Algorithm 1 is 2.25.

At last, the time complexity of the Algorithm 1 is $O(n \log n)$ which is obviously from the algorithm. □

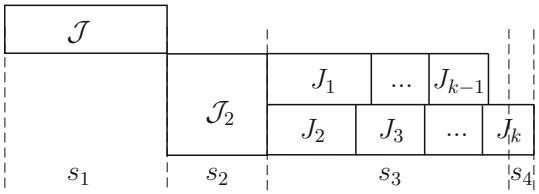

Fig. 2 An illustration of the schedule produced by Algorithm 1 under case 2, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_3 is the average processing time of jobs in $\mathcal{J}_1$ on the second stage, s_4 is $C_{\max}(L) - s_1 - s_2 - s_3$, and it's easy to know that $s_4 \leq \frac{1}{2} p_{2k} \leq \frac{1}{2} p_{23}$

3 A 7/3-approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$

In this section, we consider $F2(1, P3) \mid line_i \mid C_{\max}$: the two-stage flexible flow shop scheduling problem with a machine at the first stage and three machines (m_1, m_2 and m_3) at the second stage, the objective is to minimize the makespan. It is easy to see that the bounds of $line_i$ case are also hold for the $size_i$ case. Hence, the algorithm we present in this section are applicable for $F2(1, P3) \mid size_i \mid C_{\max}$.

Now, we are ready to present an approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$.

In the first stage, we arrange all the jobs on the machine one by one in random order non-preemptively.

In the second stage, we first group the jobs into three subsets: $\mathcal{J}_1$ denote the set of jobs that requiring only one machine at the second stage, $\mathcal{J}_2$ denote the set of jobs that requiring two machines at the second stage, and $\mathcal{J}_3$ denote the set of jobs that requiring three machines at the second stage. Secondly, we sort the jobs in $\mathcal{J}_1$ in non-increasing order according to p_{2i}. At last, we first arrange the jobs in $\mathcal{J}_3$ on the three parallel machines one by one non-preemptively, and then arrange the jobs in $\mathcal{J}_2$ on machine m_1 and machine m_2, arrange the jobs in $\mathcal{J}_1$ on machine m_3. Here we discuss the following two cases:

Case 1: $\sum_{J_i \in \mathcal{J}_2} p_{2i} \geq \sum_{J_i \in \mathcal{J}_1} p_{2i}$, i.e., machine m_3 is idle after processing the jobs in $\mathcal{J}_1$. In this case, we continue processing the jobs in $\mathcal{J}_2$ on machine m_1 and machine m_2.

Case 2: $\sum_{J_i \in \mathcal{J}_2} p_{2i} < \sum_{J_i \in \mathcal{J}_1} p_{2i}$. In this case, after processing all the jobs in $\mathcal{J}_2$ on machine m_1 and machine m_2, we process the remaining jobs of $\mathcal{J}_1$ on machine m_1, machine m_2 and machine m_3 according to list scheduling rule.

A high-level description of the approximation algorithm is showed in Algorithm 2.

Algorithm 2: Approximation algorithm for $F2\,(1,\, P3)\,|line_i|C_{\max}$

Input: The job set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a two-stage flexible flow shop scheduling problem,the first stage has only one machine and the second stage has three machines.

Output: A schedule L.

Step 1. Let $\mathcal{J}_1 = \phi, \mathcal{J}_2 = \phi, \mathcal{J}_3 = \phi$.

Step 2. For $1 \leq i \leq n$,

if $line_i = 1$,then add J_i to $\mathcal{J}_1$,
if $line_i = 2$,then add J_i to $\mathcal{J}_2$,
if $line_i = 3$,then add J_i to $\mathcal{J}_3$.

Step 3. The jobs of $\mathcal{J}$ are processed on the machine of the first stage one by one in random order non-preemptively.

Step 4. The jobs of $\mathcal{J}_3$ are processed on the three parallel machines of second stage one by one non-preemptively.

Step 5. The jobs of $\mathcal{J}_2$ are processed on machine m_1 and machine m_2 of the second stage one by one non-preemptively. The jobs of $\mathcal{J}_1$ are processed on machine m_3 of second stage in non-increasing order according to p_{2i}. Here we discuss the following two cases:

Case 1: $\sum_{J_i \in \mathcal{J}_2} p_{2i} \geq \sum_{J_i \in \mathcal{J}_1} p_{2i}$, i.e., machine m_3 is idle after processing the jobs in $\mathcal{J}_1$. In this case, we continue processing the jobs in $\mathcal{J}_2$ on machine m_1 and machine m_2.

Case 2: $\sum_{J_i \in \mathcal{J}_2} p_{2i} < \sum_{J_i \in \mathcal{J}_1} p_{2i}$. In this case, after processing all the jobs in $\mathcal{J}_2$ on machine m_1 and machine m_2, we process the remaining jobs of $\mathcal{J}_1$ on machine m_1, machine m_2 and machine m_3 in non-increasing order according to p_{2i} by using list scheduling rule.

Theorem 2 *The Algorithm 2 is an $O(n \log n)$-time $7/3$-approximation algorithm $F2\,(1, P3)\,|line_i|C_{\max}$.*

Proof Without loss of generality, we assume that the job set $\mathcal{J}_1 = \{J_1, J_2, \ldots, J_k\}$ satisfies $p_{21} \geq p_{22} \geq \cdots \geq p_{2k}$.

Let us consider the following three cases.

Case 1: $\sum_{J_i \in \mathcal{J}_2} p_{2i} \geq \sum_{J_i \in \mathcal{J}_1} p_{2i}$.

From the feature of $F2\,(1, P3)\,|line_i|C_{\max}$, we have

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}} p_{1i}, \tag{7}$$

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i}. \tag{8}$$

According to the Algorithm 2, we have (an illustration is showed in Fig. 3)

$$C_{\max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i}. \tag{9}$$

Put (7), (8) and (9) together, we have

$$C_{\max}(L) \leq 2C^*_{\max}.$$

Case 2: $\sum_{J_i \in \mathcal{J}_2} p_{2i} < \sum_{J_i \in \mathcal{J}_1} p_{2i}$ and $p_{21} \leq \sum_{J_i \in \mathcal{J}_2} p_{2i}$.

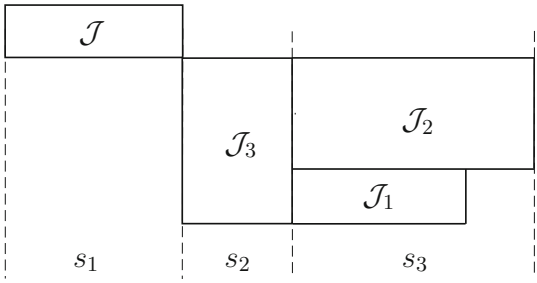

Fig. 3 An illustration of the schedule produced by Algorithm 2 under case 1, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of jobs in $\mathcal{J}_2$ on the second stage

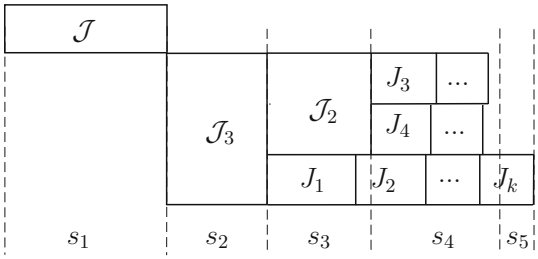

Fig. 4 An illustration of the schedule produced by Algorithm 2 under case 2, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_4 is the average processing time of the processing time of jobs in $\mathcal{J}_1$ subtract the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_5 is $C_{\max}(L) - s_1 - s_2 - s_3 - s_4$, and it's easy to know that $s_5 \leq \frac{2}{3} p_{2k} \leq \frac{2}{3} p_{22}$

In this case, (7) is also hold.

From the feature of $F2\,(1, P3)\,|line_i|C_{\max}$, we have

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}_3} p_{2i} + \frac{2}{3} \sum_{J_i \in \mathcal{J}_2} p_{2i} + \frac{1}{3} \sum_{J_i \in \mathcal{J}_1} p_{2i}. \tag{10}$$

There are at least two jobs in $\mathcal{J}_1$ since $\sum_{J_i \in \mathcal{J}_2} p_{2i} < \sum_{J_i \in \mathcal{J}_1} p_{2i}$ and $p_{21} \leq \sum_{J_i \in \mathcal{J}_2} p_{2i}$. Hence, we have

$$C^*_{\max} \geq p_{21} + p_{22}. \tag{11}$$

According to the Algorithm 2, we have (an illustration is showed in Fig. 4)

$$\begin{aligned} C_{\max}(L) \leq & \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} \\ & + \frac{1}{3} \left(\sum_{J_i \in \mathcal{J}_1} p_{2i} - \sum_{J_i \in \mathcal{J}_2} p_{2i} \right) + \frac{2}{3} p_{22}. \end{aligned} \tag{12}$$

Put (7), (10), (11) and (12) together, we have

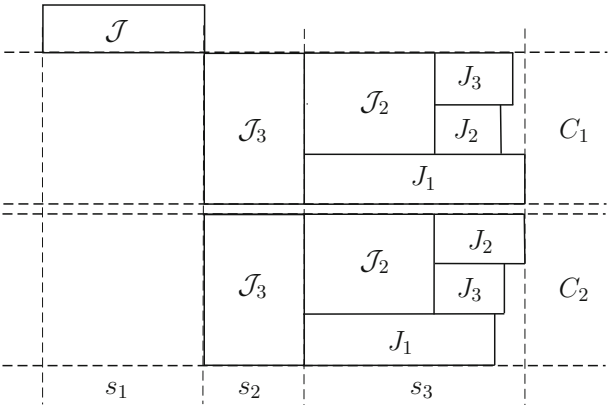

Fig. 5 An illustration of the schedule produced by Algorithm 2 under case 3 with $k < 3$, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is $C_{\max}(L) - s_1 - s_2$, C_1 is the case that $p_{21} \geq \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}$, C_2 is the case that $p_{21} < \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}$

$$\begin{aligned} C_{\max}(L) &\leq 2C^*_{\max} + \tfrac{2}{3} \times \tfrac{1}{2}(p_{21} + p_{22}) \\ &\leq \tfrac{7}{3} C^*_{\max}. \end{aligned}$$

Case 3: $p_{21} > \sum_{J_i \in \mathcal{J}_2} p_{2i}$.

In this case, (7) and (10) are also hold.

If the number of the jobs in $\mathcal{J}_1$ satisfies $k < 4$, then from the feature of $F2\,(1, P3)\,|line_i|C_{\max}$,we have

$$C^*_{\max} \geq \max\left\{ \sum_{J_i \in \mathcal{J}_3} p_{2i} + p_{21}, \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22} \right\}. \tag{13}$$

According to the Algorithm 2, we have (an illustration is showed in Fig. 5)

$$C_{\max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + \max\left\{ \sum_{J_i \in \mathcal{J}_3} p_{2i} + p_{21}, \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22} \right\}. \tag{14}$$

Put (7), (13) and (14) together, we have

$$C_{\max}(L) \leq 2C^*_{\max}.$$

Otherwise, the number of the jobs in $\mathcal{J}_1$ satisfies $k \geq 4$, i.e., there exists at least four jobs J_1, J_2, J_3 and J_4 in $\mathcal{J}_1$, which means one of the machines at the second stage must process at least two jobs of $\{J_1, J_2, J_3, J_4\}$ since there are only three machines. Hence, we have

$$C^*_{\max} \geq p_{23} + p_{24}. \tag{15}$$

From the feature of $F2\,(1, P3)\,|line_i|C_{\max}$, we have

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}_3} p_{2i} + p_{21}, \tag{16}$$

$$C^*_{\max} \geq \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}. \tag{17}$$

According to the Algorithm 2, we know that at the second stage, the job J_1 is processed on the machine m_3, the job J_2 is processed on the machine m_1, the job J_3 is processed on the machine m_2.

Let us consider the following two subcases.

Case 3.1: $p_{21} < \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}$.

According to the Algorithm 2, we first process the remaining jobs in $\mathcal{J}_1$ on machines m_2 and m_3.

If machines m_2 and m_3 are idle after finish processing the remaining jobs in $\mathcal{J}_1$ when the job J_2 are still processed on machine m_1, then, according to the Algorithm 2, we have (an illustration is showed in Fig. 6)

$$C_{\max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}. \tag{18}$$

Put (7), (17) and (18) together, we have

$$C_{\max}(L) \leq 2C^*_{\max}.$$

Otherwise, the job J_2 is finished first on machine m_1, then we process the remaining jobs in $\mathcal{J}_1$ on machines m_1, m_2 and m_3 according to list scheduling rule, so we have (an illustration is showed in Fig. 7)

$$\begin{aligned} C_{\max}(L) \leq & \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} \\ & + \frac{1}{3}\left(\sum_{J_i \in \mathcal{J}_1} p_{2i} - \sum_{J_i \in \mathcal{J}_2} p_{2i}\right) + \frac{2}{3} p_{24}. \end{aligned} \tag{19}$$

Put (7), (10), (15) and (19) together, we have

$$C_{\max}(L) \leq 2C^*_{\max} + \frac{2}{3} \times \frac{1}{2}(p_{23} + p_{24}) \leq \frac{7}{3} C^*_{\max}.$$

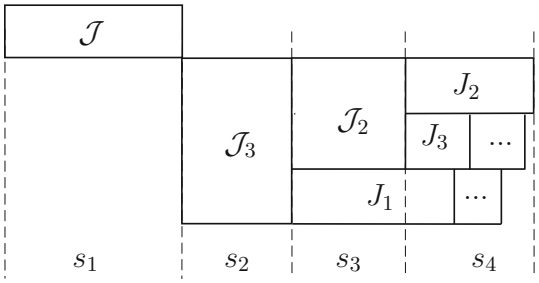

Fig. 6 An illustration of the schedule produced by Algorithm 2 under case 3.1 with machines m_2 and m_3 are idle after finish processing the remaining jobs in $\mathcal{J}_1$ when the job J_2 are still processed on machine m_1, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_4 is the processing time of job J_2 on the second stage

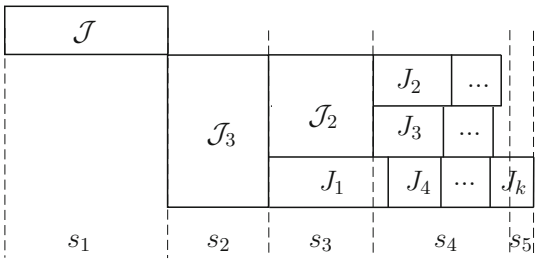

Fig. 7 An illustration of the schedule produced by Algorithm 2 under case 3.1 with the job J_2 is finished first on machine m_1, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_4 is the average processing time of the processing time of jobs in $\mathcal{J}_1$ subtract the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_5 is $C_{\max}(L) - s_1 - s_2 - s_3 - s_4$, and it's easy to know that $s_5 \leq \frac{2}{3}p_{2k} \leq \frac{2}{3}p_{24}$

Case 3.2: $p_{21} \geq \sum_{J_i \in \mathcal{J}_2} p_{2i} + p_{22}$.

According to the Algorithm 2, we first process the remaining jobs in $\mathcal{J}_1$ on machines m_1 and m_2.

If machines m_1 and m_2 are idle after finishing all the jobs in $\mathcal{J}_1$ when the job J_1 is still processed on machine m_3, then, according to the Algorithm 2, we have (an illustration is showed in Fig. 8)

$$C_{\max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + p_{21}. \tag{20}$$

Put (7), (16) and (20) together, we have

$$C_{\max}(L) \leq 2C^*_{\max}.$$

Otherwise, the job J_1 is finished first on machine m_3, then we process the remaining jobs in $\mathcal{J}_1$ on machines m_1, m_2 and m_3 according to list scheduling rule, so we have

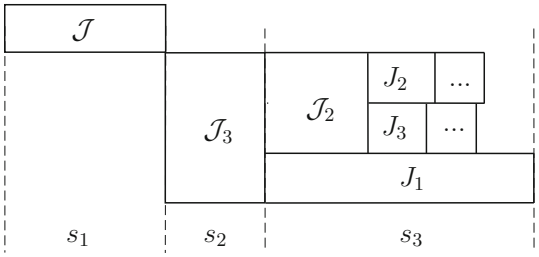

Fig. 8 An illustration of the schedule produced by Algorithm 2 under case 3.2 with machines m_1 and m_2 are idle after finishing all the jobs in $\mathcal{J}_1$ when the job J_1 is still processed on machine m_3, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of job J_1 on the second stage

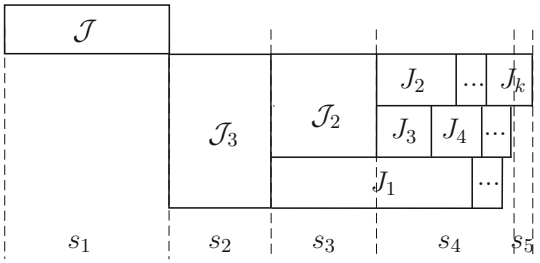

Fig. 9 An illustration of the schedule produced by Algorithm 2 under case 3.2 with the job J_1 is finished first on machine m_3, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the processing time of jobs in $\mathcal{J}_3$ on the second stage, s_3 is the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_4 is the average processing time of the processing time of jobs in $\mathcal{J}_1$ subtract the processing time of jobs in $\mathcal{J}_2$ on the second stage, s_5 is $C_{\max}(L) - s_1 - s_2 - s_3 - s_4$, and it's easy to know that $s_5 \leq \frac{2}{3}p_{2k} \leq \frac{2}{3}p_{24}$

(an illustration is showed in Fig. 9)

$$C_{\max}(L) \leq \sum_{J_i \in \mathcal{J}} p_{1i} + \sum_{J_i \in \mathcal{J}_3} p_{2i} + \sum_{J_i \in \mathcal{J}_2} p_{2i} + \frac{1}{3}\left(\sum_{J_i \in \mathcal{J}_1} p_{2i} - \sum_{J_i \in \mathcal{J}_2} p_{2i}\right) + \frac{2}{3}p_{24}. \tag{21}$$

Put (7), (10), (15) and (21) together, we have

$$\begin{aligned} C_{\max}(L) &\leq 2C^*_{\max} + \tfrac{2}{3} \times \tfrac{1}{2}(p_{23} + p_{24}) \\ &\leq \tfrac{7}{3}C^*_{\max}. \end{aligned}$$

Hence, the approximation ratio of the Algorithm 2 is 7/3.
At last, it is easy to know that the time complexity of the Algorithm 2 is $O(n \log n)$.
□

4 An optimal algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ under some special conditions

In this section, we consider $F2(1, Pm) \mid size_i \mid C_{\max}$ under some special conditions: the two-stage flexible flow shop scheduling problem with a machine at the first stage and m machines at the second stage, the processing time of the jobs satisfies: $\min_{1\le i\le n}\{p_{1i}\} \ge \max_{1\le i\le n}\{p_{2i}\}$, the objective is to minimize the makespan.

Now, we are ready to present an optimal algorithm for the problem.

Firstly, for the given job set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, find out the job with the smallest processing time on the second stage, if there are more than one job with the smallest processing time on the second stage, then arbitrarily choose one of them. Without loss of generality, we assume J_n is the job with the smallest processing time on the second stage, i.e., $p_{2n} = \min\limits_{1\le i\le n}\{p_{2i}\}$.

Secondly, process the jobs with the order $\{J_1, J_2, \ldots, J_n\}$ non-preemptively in the first stage.

Lastly, for each job J_i, process it in the second stage immediately when it has been finished in the first stage.

A high-level description of the algorithm is showed in Algorithm 3.

Algorithm 3

Input: The job set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a two-stage flexible flow shop scheduling problem, the first stage has only one machine and the second stage has m machines, and the processing time of the jobs satisfies: $\min_{1\le i\le n}\{p_{1i}\} \ge \max_{1\le i\le n}\{p_{2i}\}$.

Output: A schedule L.

Step 1. Find out the job with the smallest processing time on the second stage, without loss of generality, denoted it as J_n.

Step 2. Process the jobs with the order $\{J_1, J_2, \ldots, J_n\}$ non-preemptively in the first stage.

Step 3. For each job J_i, process it in the second stage immediately when it has been finished in the first stage.

Theorem 3 *The Algorithm 3 is a linear time optimal algorithm for* $F2(1, Pm) \mid size_i \mid C_{\max}$ *with the assumption that* $\min_{1\le i\le n}\{p_{1i}\} \ge \max_{1\le i\le n}\{p_{2i}\}$.

Proof From the feature of $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption that $\min_{1\le i\le n}\{p_{1i}\} \ge \max_{1\le i\le n}\{p_{2i}\}$, we have

$$C^*_{\max} \ge \sum_{J_i\in\mathcal{J}} p_{1i} + \min_{1\le i\le n}\{p_{2i}\}. \tag{22}$$

According to Algorithm 3 with the assumption that $\min_{1\le i\le n}\{p_{1i}\} \ge \max_{1\le i\le n}\{p_{2i}\}$, we have (an illustration is showed in Fig. 10)

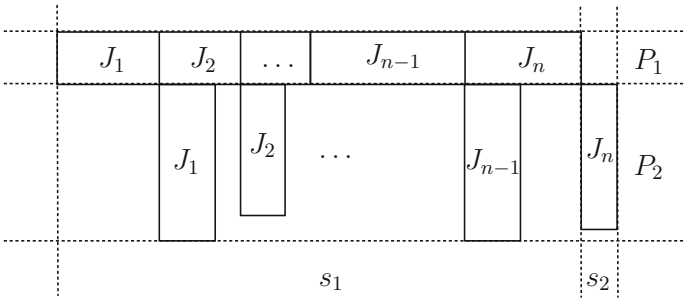

Fig. 10 An illustration of the schedule produced by Algorithm 3, where s_1 is the processing time of jobs in $\mathcal{J}$ on the first stage, s_2 is the minimal processing time of jobs in $\mathcal{J}$ on the second stage, P_1 is the first stage, P_2 is the second stage

$$C_{max}(L) = \sum_{J_i \in \mathcal{J}} p_{1i} + p_{2n} = \sum_{J_i \in \mathcal{J}} p_{1i} + \min_{1 \le i \le n} \{p_{2i}\}. \tag{23}$$

Put (22) and (23) together, we have

$$C_{max}(L) \le C^*_{\max}.$$

Combining with the fact that $C^*_{\max}$ is the optimal value, we have

$$C_{max}(L) = C^*_{\max}.$$

Hence, the Algorithm 3 is an optimal algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption that $\min_{1 \le i \le n} \{p_{1i}\} \ge \max_{1 \le i \le n} \{p_{2i}\}$.

At last, it is easy to know that the Algorithm 3 can be finished in linear time. □

5 Concluding remarks

In this paper, we studied the two-stage flexible flow shop scheduling problems. We first presented an $O(n \log n)$-time 2.25-approximation algorithm for $F2(1, P2) \mid line_i \mid C_{\max}$, and then proposed an $O(n \log n)$-time 7/3-approximation algorithm for $F2(1, P3) \mid line_i \mid C_{\max}$. Lastly we proposed a linear time optimal algorithm for $F2(1, Pm) \mid size_i \mid C_{\max}$ with the assumption that $\min_{1 \le i \le n} \{p_{1i}\} \ge \max_{1 \le i \le n} \{p_{2i}\}$.

As a future research topic, firstly, it will be meaningful to consider the more efficient approximation algorithms for the problem discussed in this paper. Secondly, design optimal algorithms or approximation algorithms for other type of two-stage flexible flow shop scheduling problems are also very interesting.

Data Availibility Statement Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

References

Alisantoso D, Khoo LP, Jiang PY (2003) An immune algorithm approach to the scheduling of a flexible PCB flow shop. Int J Adv Manuf Technol 22:819–827

Almeder C, Hartl RF (2013) A metaheuristic optimization approach for a real-world stochastic flexible flow shop problem with limited buffer. Int J Prod Econ 145:88–95

Arthanari TS, Ramamurthy KG (1971) An extension of two machines sequencing problem. Opsearch 8:10–22

Choi BC, Lee K (2013) Two-stage proportionate flexible flow shop to minimize the makespan. J Comb Optim 25:123–134

Graham RL, Lawler EL, Lenstra JK, Kan R (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5:287–326

Gupta J, Krüger K, Lauff V, Werner F, Sotskov Y (2002) Heuristics for hybrid flow shops with controllable processing times and assignable due dates. Comput Oper Res 29:1417–1439

He LM, Sun SJ, Luo RZ (2008) Two-stage flexible flow shop scheduling problems with a batch processor on second stage. Chin J Eng Math 25:829–842

Lee CY, Vairaktarakis GL (1994) Minimizing makespan in hybrid flowshops. Oper Res Lett 16:149–158

Lin HT, Liao CJ (2003) A case study in a two-stage hybrid flow shop with setup time and dedicated machines. Int J Prod Econ 86:133–143

Moseley B, Dasgupta A, Kumar R, Sarlós T (2011) On scheduling in map-reduce and flow-shops. In: Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures(SPAA '11), pp 289–298

Salvador MS (1973) A solution to a special class of flow shop scheduling problems. In: Symposium of theory of scheduling and its applications, pp 83–91

Sun JH, Deng QX, Meng YK (2014) Two-stage workload scheduling problem on GPU architectures: formulation and approximation algorithm. J Softw 25:298–313

Zhang MH, Lan Y, Han X (2020) Approximation algorithms for two-stage flexible flow shop scheduling. J Comb Optim 39:1–14