软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(5):1073−1084 [doi: 10.3724/SP.J.1001.2012.00587]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一类两阶段杂交流水作业的近似算法*

魏 麒[1], 蒋义伟[2+]

[1](浙江大学 宁波理工学院, 浙江 宁波 315100)

[2](浙江理工大学 理学院, 浙江 杭州 310018)

Approximation Algorithms for a Two-Stage Hybrid Flow Shop

WEI Qi[1], JIANG Yi-Wei[2+]

[1](Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China)

[2](School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China)

+ Corresponding author: E-mail: mathjyw@yahoo.com.cn

Wei Q, Jiang YW. Approximation algorithms for a two-stage hybrid flow shop. *Journal of Software*, 2012, 23(5):1073−1084. http://www.jos.org.cn/1000-9825/587.htm

Abstract: This paper investigates a variant scheduling problem of minimizing makespan in a two-machine flow shop. In this variant, there will be two tasks for each job. The first task can be processed on either machine, and the second task can only be processed on the second machine after the first task has been finished. Furthermore, if the second task should start right after the first task is completed, it is called a no-waited case and is denoted by NSHFS. On the other hand, if the second task is allowed to be processed at any time after the first task is completed, the problem is then denoted as SHFS. In the case of SHFS, based on the result of Wei and He, an improved polynomial time approximation algorithm with worst-case ratio of 8/5 is presented. In the case of NSHFS, this paper shows that it is NP-hard, and presents a polynomial time approximation algorithm with worst-case ratio of 5/3.

Key words: flowshop scheduling; computational complexity; approximation algorithm; worst-case ratio; makespan

摘 要: 讨论了一类两台机流水作业要求最后完工工件完工时间最早的排序问题.问题中每个工件包含两个加工任务:第 1 个任务可以在任何一台机器上加工,第 2 个任务只能在第 1 个任务完成后在第 2 台机器上加工.如果要求在加工同一个工件的两个任务时,两个任务之间不能有停顿,则称其为不可等待的模型,记作 NSHFS.如果第 2 个任务可以在第 1 个任务完成后的任意时间加工,则称其为允许等待的模型,记作 SHFS.对于 SHFS 模型,在魏麒和何勇工作的基础上给出了一种改进的最坏情况界为 8/5 的多项式时间近似算法.对于 NSHFS 模型,首先证明它是 NP-难的,并且给出了一种最坏情况界为 5/3 的多项式时间近似算法.

关键词: 流水作业;计算复杂性;近似算法;最坏情况界;最后完工工件完工时间

中图法分类号: TP393, O223 **文献标识码**: A

* 基金项目: 国家自然科学基金(11001242, 11071220); 浙江省自然科学基金(Y6090554, Y6090175)
收稿时间: 2008-12-10; 定稿时间: 2010-06-29

1 Introduction

This paper considers the following two-machine flow shop scheduling problem. There is a set of independent jobs $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$ to be processed in two machines M_1 and M_2. Each J_i, for $i=1,2,\ldots,n$, consists of two tasks A_i and B_i, and only upon finishing task A_i can task B_i start. Task A_i can be processed on M_1 for a_i time units, or on M_2 for a_i time units; task B_i can only be processed on M_2 for b_i time units. We assume that jobs and machines are available at time zero, and preemption is not allowed. For a given schedule, we denote the completion time of task A_i by C_{A_i}, S_{B_i}, the start time of task B_i, and $C_{\max}=\max\{C_i\}$, the makespan. The goal is to minimize the makespan.

The above model, proposed first by Wei and He[1], applies to the graphic programs processing which comprises of data and graphics processing. Graphics processing cannot start until data processing is completed. Data processing can be done by either CPU (central processing unit), or GPU (graphic processing unit), while graphics processing must be done by GPU. The above model is derived from the fact that we refer to CPU and GPU as two machines and the processing of data and graphics as the jobs. We usually put the results of data processing into a cache memory in advance and take them out when we begin graphics processing.

The two-stage hybrid flow shop problems (denoted by HFS) proposed by Panagiotis and George[2] are similar to our problem. It is assumed that the two tasks of each job can be both processed on either machine, or processed by the traditional model, i.e., the first task on M_1 and the second task on M_2. Clearly, in HFS, there are three processing models for each job, while only two models in our problem. Hence, we call our problem the two-stage semi- hybrid flow shop problem. In some cases, the space of the cache memory is not enough to store some of the results that data processing needs. Sometimes, there is no cache memory. Under these circumstances, the task B_i should begin to be processed after the completion of task A_i, i.e. $C_{A_i}=S_{B_i}$. We call this problem as a two-stage no-waited semi-hybrid flowshop denoted by NSHFS. On the other hand, if task B_i is allowed to be processed any time after the completion time of task A_i, i.e. $S_{B_i}\geqslant C_{A_i}$, then this problem would be denoted by SHFS. In this paper, both problems NSHFS and SHFS will be considered.

The problem HFS is proved to be NP-hard, and Ref.[2] proposed an optimal algorithm based on dynamic programming and extends it to a pseudo-polynomial approximation algorithm for a generalized problem. Another closely related problem is called two-stage flow shop problem with multi-processor flexibility by Vairaktarakis and Lee[3], where the first task must be completed before the second task can start. Moreover, the first (second) task can be processed on M_1 (M_2), or on both processors simultaneously with smaller processing time. For this problem, Vairaktarakis and Lee presented a dynamic programming algorithm and a polynomial time approximation algorithm with a worst-case ratio of 1.618.

In Ref.[1], the problem SHFS was also considered. Simply, the processing times of task A_i on M_1 and M_2 is different. They showed the problem SHFS is ordinary NP-hard, and presented a pseudo-polynomial time optimal algorithm and a polynomial time approximation algorithm with a worst-case ratio 2. In this paper, we present an approximation algorithm with a worst-case ratio of 8/5 for SHFS. In addition, we consider the new problem NSHFS. We show it is NP-hard, and present a polynomial time approximation algorithm with a worst-case ratio of 5/3.

The rest of the paper is organized as follows. In Section 2, we present a better polynomial time approximation algorithm with a worst-case ratio of 8/5 for the SHFS problem. In Section 3, we show the problem NSHFS is NP-hard and present a polynomial time approximation algorithm with a worst-case ratio of 5/3.

In the remainder of this paper, let C^H and C^* be the makespan yielded by an algorithm H and an optimal schedule for a given instance of SHFS or NSHFS.

2 SHFS

To present the polynomial time approximation algorithm for problem SHFS, we first define two processing modes for each job. One is called *mode* 1 if both tasks A_i and B_i of job J_i are processed in turn on M_2 with a_i and b_i time units, respectively. Another one is called *mode* 2 if the task A_i is processed on M_1 with a_i time units and task B_i on M_2 with b_i time units. It is easy to see that every job should be processed by mode 1, or alternatively mode 2 due to the definition of problem SHFS. Consequently, in an arbitrary schedule, the set of jobs $\mathcal{J}$ can be partitioned into two non-intersecting subsets: V_1 and V_2. V_1 is made up of all jobs which are processed by mode 1; V_2 is made up of all jobs which are processed by mode 2.

Johnson's rule[4] is the optimal schedule of two-machine flowshop, but if every task A_i of job J_i is processed on M_1 and task B_i on M_2, and the jobs in $\mathcal{J}$ are processed in the Johnson's rule, Wei and He[1] show the worst-case ratio is 2. Thus, Johnson's rule is not very efficient for SHFS.

2.1 A greedy-like algorithm H_1 for SHFS

In this subsection, we present an algorithm, denoted by H_1, including two phases 1 and 2, which are used for partitioning all jobs into two sets and processing jobs, respectively.

Algorithm H_1:

Let V_i^k be a set of jobs processed by mode i, i=1,2 after assigning the first k jobs in phase 1.

Phase 1:

1. Re-index all jobs in set $\mathcal{J}$ with respect to the values of a_i such that $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_n$;
2. Let $V_1^1 = \varnothing$ and $V_2^1 = \{J_1\}$, k=2;
3. While $k \leqslant n$, if $\sum_{J_i \in V_2^{k-1}} a_i > \sum_{J_i \in V_1^{k-1}} (a_i + b_i)$, let $V_1^k = V_1^{k-1} \cup \{J_k\}$ and $V_2^k = V_2^{k-1}$, and let $V_1^k = V_1^{k-1}$ and $V_2^k = V_2^{k-1} \cup \{J_k\}$, $k \leftarrow k+1$;
4. Return V_1^n and V_2^n.

For convenience, we denote $V_1 = V_1^n$ and $V_2 = V_2^n$ in the remainder of this subsection.

Phase 2:

1. Process all tasks A_i in V_2 on M_1 in order of their indices at time zero;
2. Process all jobs in V_1 on M_2 in order of their indices at time zero, and all tasks B_i in V_2 as early as possible in order of their indices.

It is not hard to obtain that the time complexity of the algorithm H_1 is $O(n\log n)$.

Theorem 2.1. $C^* \geqslant \max\left\{\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i), \sum_{i=1}^{n} b_i, \max_{1\leqslant i\leqslant n}\{a_i+b_i\}\right\}$.

Proof: The optimal makespan must be at least the average load of the 2 machines. Therefore, it is clear that $C^* \geqslant \frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)$. For each task, B_i, i=1,2,…,n, is processed on M_2, which follows that the completion time of M_2 is at least $\sum_{i=1}^{n} b_i$. Then, $C^* \geqslant \sum_{i=1}^{n} b_i$, and $C^* \geqslant \max_{1\leqslant i\leqslant n}\{a_i+b_i\}$ holds trivially.

The proof is applicable for both jobs that are allowed to wait and those that are not. Thus, Theorem 2.1 is suitable for both SHFS and NSHFS.

Theorem 2.2. $C^{H_1}/C^* \leqslant \frac{5}{3}$ and the bound is tight.

Proof: To obtain the desired worst-case ratio, we distinguish two cases as follows.

Case 1. $\sum_{J_i \in V_2} a_i > \sum_{J_i \in V_1} (a_i + b_i)$ (see Fig.1).

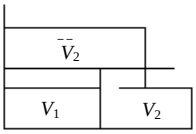


Fig.1 Case 1

Let J_t be the last job processed on M_1. It yields $J_i \in V_1$ for each $t+1 \leqslant i \leqslant n$. Furthermore, by the definition of J_t, we conclude that the completion time of all jobs in V_1^{t-1} is not less than the start time of task A_t, and the completion time of all jobs in V_1 is not greater than that of task A_t. Therefore, by step 2 of phase 2, all tasks B_i in $V_2 \backslash \{B_t\}$ can be processed on M_2 one by one right after the completion time of all jobs in V_1. Hence, the makespan is determined by task B_t. Two cases are considered as shown in Fig.2.

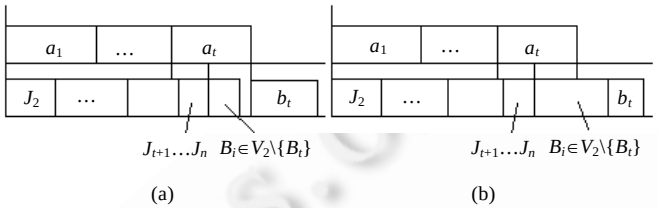


Fig.2 The makespan determined by B_t

For the first case (see Fig.2(a)), we have $C^{H_1} = \sum_{J_i \in V_2} a_i + b_t$. By the definition of J_t and step 3 in phase 1, we can conclude that $\sum_{J_i \in V_2 \backslash \{J_t\}} a_i \leqslant \sum_{J_i \in V_1^{t-1}} (a_i + b_i) \leqslant \sum_{J_i \in V_1} (a_i + b_i)$, together with Theorem 2.1, leads to

$$C^{H_1} = \sum_{J_i \in V_2 \backslash \{J_t\}} a_i + a_t + b_t = \frac{1}{2} \sum_{J_i \in V_2 \backslash \{J_t\}} a_i + \frac{1}{2} \sum_{J_i \in V_2 \backslash \{J_t\}} a_i + a_t + b_t \leqslant \frac{1}{2} \sum_{J_i \in V_2 \backslash \{J_t\}} a_i + \frac{1}{2} \sum_{J_i \in V_1} (a_i + b_i) + a_t + b_t$$
$$\leqslant \frac{1}{2} \sum_{J_i \in V_2} (a_i + b_i) + \frac{1}{2} \sum_{J_i \in V_1} (a_i + b_i) + \frac{1}{2}(a_t + b_t) = \frac{1}{2} \sum_{i=1}^{n} (a_i + b_i) + \frac{1}{2}(a_t + b_t) \leqslant C^* + \frac{1}{2} C^* = \frac{3}{2} C^*.$$

For the second case (see Fig.2(b)), we have

$$C^{H_1} = \sum_{J_i \in V_1} a_i + \sum_{n=1}^{n} b_i \tag{1}$$

and thus

$$C^{H_1} \leqslant \sum_{J_i \in V_2} (a_i + b_i) \tag{2}$$

due to $\sum_{J_i \in V_1} (a_i + b_i) < \sum_{J_i \in V_2} a_i$. Then, from Eq.(1), Eq.(2) and Theorem 2.1, we have

$$C^{H_1} \leqslant \frac{1}{2} \left(\sum_{J_i \in V_1} a_i + \sum_{i=1}^{n} b_i + \sum_{J_i \in V_2} (a_i + b_i) \right) \leqslant \frac{1}{2} \sum_{i=1}^{n} (a_i + b_i) + \frac{1}{2} \sum_{J_i \in V_2} b_i \leqslant C^* + \frac{1}{2} C^* = \frac{3}{2} C^*.$$

Case 2. $\sum_{J_i \in V_2} a_i \leqslant \sum_{J_i \in V_1} (a_i + b_i)$ (see Fig.3(a)). Then $C^{H_1} = \sum_{J_i \in V_1} a_i + \sum_{i=1}^{n} b_i$. We distinguish two cases according to the number of jobs in V_1.

Subcase 2.1. $|V_1|=1$ (see Fig.3(b)). We have $C^{H_1} = a_2 + \sum_{i=1}^{n} b_i$ and $a_2 \leqslant \frac{1}{2} \sum_{i=1}^{n} a_i$ from $a_2 \leqslant a_1$, then

$$C^{H_1} \leqslant \frac{1}{2} \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i = \frac{1}{2} \sum_{i=1}^{n} (a_i + b_i) + \frac{1}{2} \sum_{i=1}^{n} b_i \leqslant C^* + \frac{1}{2} C^* = \frac{3}{2} C^*.$$

Subcase 2.2. $|V_1|>1$. Let J_t be the last job to be put into V_1 (see Fig.3(c)). It yields that the start time of job J_t is

less than the completion time of all tasks A_i in V_2^{t-1}, which implies that

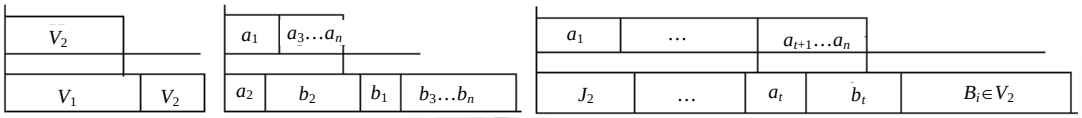
$$\sum_{J_i\in V_1}(a_i+b_i)-(a_t+b_t)\leqslant \sum_{J_i\in V_2^{t-1}}a_i\leqslant \sum_{J_i\in V_2}a_i=\sum_{i=1}^{n}a_i-\sum_{J_i\in V_1}a_i,$$

that is $\sum_{J_i\in V_1}a_i\leqslant\frac{1}{2}(a_t+b_t+\sum_{i=1}^{n}a_i-\sum_{J_i\in V_1}b_i)$.

Since $a_t\leqslant a_i$, i=1,2,…,t, we have $a_t\leqslant\frac{1}{t}\sum_{i=1}^{t}a_i\leqslant\frac{1}{t}\sum_{i=1}^{n}a_i$. Then, we obtain that

$$\begin{aligned}C^{H_1}&=\sum_{J_i\in V_1}a_i+\sum_{i=1}^{n}b_i\leqslant\frac{1}{2}(a_t+b_t+\sum_{i=1}^{n}a_i-\sum_{J_i\in V_1}b_i)+\sum_{i=1}^{n}b_i=\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)+\frac{1}{2}(a_t+b_t)+\frac{1}{2}\sum_{J_i\in V_2}b_i\\&\leqslant\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)+\frac{1}{2}a_t+\frac{1}{2}\sum_{i=1}^{n}b_i\leqslant\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)+\frac{1}{2t}\sum_{i=1}^{n}a_i+\frac{1}{2}\sum_{i=1}^{n}b_i\\&=\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)+\frac{1}{2t}\sum_{i=1}^{n}(a_i+b_i)+\frac{t-1}{2t}\sum_{i=1}^{n}b_i\leqslant C^*+\frac{1}{t}C^*+\frac{t-1}{2t}C^*=\left(\frac{3}{2}+\frac{1}{2t}\right)C^*\leqslant\frac{5}{3}C^*.\end{aligned}$$

The last inequality holds because $t\geqslant 3$, due to $|V_1|$ and the definition of t.



(a) (b) (c)

Fig.3 Case 2

The following instance shows that the worst-case ratio of 5/3 is tight. Consider the instance $\mathcal{I}=\{J_1,J_2,J_3,J_4\}$.

Where $a_1=L$, $b_1=\varepsilon$; $a_2=L-2\varepsilon$, $b_2=\varepsilon$; $a_3=L-3\varepsilon$, $b_3=L$; $a_4=3\varepsilon$, $b_4=2L(\varepsilon<<L)$. It is easy to obtain $C^*=3L+5\varepsilon$ by processing J_4 by mode 1 and J_1,J_2,J_3 by model 2 with the order of J_4,J_3,J_2,J_1, while $C^{H_1}=5L-3\varepsilon$. It follows that $C^{H_1}/C^*=\frac{(5L-3\varepsilon)}{(3L+5\varepsilon)}\to\frac{5}{3}$ when ε tends to 0.

2.2 An improvement of Algorithm H_1

In this section, we present an algorithm H_2 with a worst-case ratio of 8/5 by improving the algorithm H_1. Recall that, in the proof of Theorem 2.2, the worst-case ratio in case 1 and subcase 2.1 is 3/2, and 3/2+ $1/2t$ in subcase 2.2. Then, the desired result can be obtained trivially by algorithm H_1 if $t\geqslant 5$. In fact, the algorithm H_1 can also achieve the desired result when t=4 (see Lemma 2.3). Hence, in order to improve the worst-case ratio, we only need to improve the method for case t=3.

Before going to present the improved algorithm, we first show the following lemma.

Lemma 2.3. If t=4 in subcase 2.2 of the proof of Theorem 2.2, then $C^{H_1}/C^*\leqslant\frac{8}{5}$.

Proof: Recall that, $C^{H_1}=\sum_{J_i\in V_1}a_i+\sum_{i=1}^{n}b_i$ from case 2 in the proof of Theorem 2.2. Since $J_t\in V_1$, by the rule of the algorithm H_1, we can conclude that

$$\sum_{J_i\in V_2^{t-1}}a_i>\sum_{J_i\in V_1^{t-1}}(a_i+b_i)\tag{3}$$

Note that if $J_1\in V_2$ and $J_2\in V_1$, we then consider two cases according to the assignment of J_3.

Case 1. $J_3\in V_1$ (see Fig.4(a)). Then, Eq.(3) yields that $a_1\geqslant a_2+b_2+a_3+b_3\geqslant a_2+a_3$. Since $a_4\leqslant a_3\leqslant a_2$, we have $a_4\leqslant\frac{1}{2}(a_2+a_3)\leqslant\frac{1}{2}a_1$. Thus, $\frac{5}{2}a_4\leqslant\frac{1}{2}(a_2+a_3)+\frac{1}{2}a_1+\frac{1}{2}a_4=\frac{1}{2}(a_1+a_2+a_3+a_4)\leqslant\frac{1}{2}\sum_{i=1}^{n}a_i$.

That is, $\frac{1}{2}a_4 \leqslant \frac{1}{10}\sum_{i=1}^{n} a_i$. Therefore,

$$C^{H_1} = \sum_{J_i \in V_1} a_i + \sum_{i=1}^{n} b_i = a_2 + a_3 + a_4 + \sum_{i=1}^{n} b_i = \frac{1}{2}(a_2 + a_3) + \frac{1}{2}(a_2 + a_3 + a_4) + \frac{1}{2}a_4 + \sum_{i=1}^{n} b_i$$

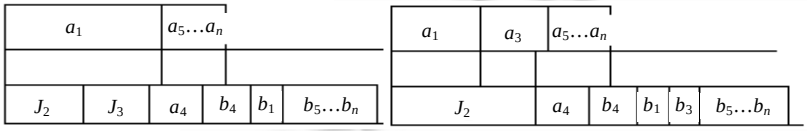
$$\leqslant \frac{1}{2}a_1 + \frac{1}{2}(a_2 + a_3 + a_4) + \frac{1}{10}\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i \leqslant \left(\frac{1}{2} + \frac{1}{10}\right)\sum_{i=1}^{n}(a_i + b_i) + \left(\frac{1}{2} - \frac{1}{10}\right)\sum_{i=1}^{n} b_i.$$

It follows that $C^{H_1} \leqslant \left(1 + \frac{1}{5}\right)C^* + \left(\frac{1}{2} - \frac{1}{10}\right)C^* = \frac{8}{5}C^*$ by Theorem 2.1.

Case 2. $J_3 \in V_2$ (see Fig.4(b)). Next, we have $C^{H_1} = \sum_{J_i \in V_1} a_i + \sum_{i=1}^{n} b_i = a_2 + a_4 + \sum_{i=1}^{n} b_i$, which, together with the fact that $a_4 \leqslant a_3 \leqslant a_2 \leqslant a_1$, leads to $C^{H_1} \leqslant \frac{1}{2}(a_1 + a_2) + \frac{1}{2}(a_3 + a_4) + \sum_{i=1}^{n} b_i \leqslant \frac{1}{2}(\sum_{i=1}^{n}(a_i + b_i)) + \frac{1}{2}\sum_{i=1}^{n} b_i$.

Hence, we have $C^{H_1} \leqslant C^* + \frac{1}{2}C^* = \frac{3}{2}C^*$ by Theorem 2.1.

The proof is complete.



(a) (b)

Fig.4 Assignment of J_3

Now, we focus on subcase 2.2 with t=3.

By the rule of algorithm H_1 and the definition of t, we can conclude that $J_1 \in V_2$ and $J_2, J_3 \in V_1$. Then, $\sum_{J_i \in V_2^{t-1}} a_i > \sum_{J_i \in V_1^{t-1}}(a_i + b_i)$ implies $a_1 > a_2 + b_2$. Recall that $\sum_{J_i \in V_2} a_i \leqslant \sum_{J_i \in V_1}(a_i + b_i)$ in case 2 in the proof of Theorem 2.2. It follows that $a_1 + \sum_{i=4}^{n} a_i \leqslant a_2 + b_2 + a_3 + b_3$. Thus, we can claim that this case occurs if and only if the job set $\mathcal{J}$ satisfies the following Condition 1:

$$\text{Condition 1:} \begin{cases} n \geqslant 3 \\ a_1 > a_2 + b_2 \\ a_1 + \sum_{i=4}^{n} a_i \leqslant a_2 + b_2 + a_3 + b_3 \end{cases}.$$

For any instance $\mathcal{J}=\{J_1, J_2, \ldots, J_n\}$, consider the job set $\mathcal{J}' = \{J_1', J_2', \ldots, J_n'\}$ where $J_i' = J_i$ for i=1,2,3 and J_i' satisfying $a_i' = 0$ and $b_i' = b_i$ for i=4,…,n. Note that every job J_i', i=4,…,n, is processed on M_2 in any solution, which implies that there exists an optimal solution to process these jobs one by one at time zero on M_2. It is easy to obtain an optimal schedule by using an enumerating method for the remainder jobs $J_1' = J_1, J_2' = J_2$ and $J_3' = J_3$. Let V_i' be the set of jobs in $\{J_1, J_2, J_3\}$ processed by mode i in this optimal schedule, i=1,2. Denote $V'=\{B_i | i=4,\ldots,n\}$, then the optimal schedule can be described as Fig.5.

Now we present a new algorithm *Ag* for subcase 2.2 with t=3 in the proof of Theorem 2.2 as follows:

Algorithm *Ag*:

1. Let $V_1'' = \varnothing$ and $V_2'' = \{J_4\}$, $k = 5$;
2. While $k \leqslant n$, if $\sum_{J_i \in V_2''} a_i > \sum_{J_i \in V_1''} a_i$, let $V_1'' = V_1'' \cup \{J_k\}$, and let $V_2'' = V_2'' \cup \{J_k\}, k \leftarrow k + 1$;
3. Process tasks $\{A_i \mid J_i \in V_2''\}$ and $\{A_i \mid J_i \in V_1''\}$ on M_1 and M_2 at time zero, respectively. Let T be the larger completion time on M_1 and M_2 for these tasks;

4. Since $V_1''\cup V_2''=\mathcal{J}\setminus\{J_1,J_2,J_3\}$, we have $\{B_i \mid J_i\in V_2''\}\cup\{B_i \mid J_i\in V_1''\}=V'$. Then, we can process the remainder tasks V', V_1', and V_2' by using the optimal schedule for $\mathcal{J}$ at time T.

The algorithm *Ag* can be described in Fig.6.

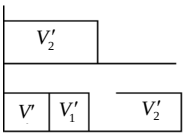


Fig.5 Optimal schedule for t=3

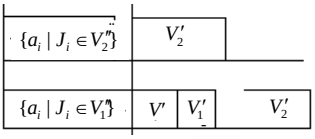


Fig.6 Illustrate of algorithm *Ag*

Lemma 2.4. If the algorithm *Ag* processes the job set $\mathcal{J}$ satisfying the Condition 1, then $C^{Ag}/C^{*}\leqslant\frac{3}{2}<\frac{8}{5}$.

Proof: Let C^{*} and $C^{*\prime}$ be the optimal makespan of $\mathcal{J}$ and $\mathcal{J}'$, respectively. It is clear that $C^{*}\geqslant C^{*\prime}$. Then, we have $C^{Ag}\leqslant T+C^{*\prime}\leqslant T+C^{*}$ from Fig.6. We will complete the proof after showing that $T\leqslant\frac{1}{2}C^{*}$.

In fact, the assignment of step 2 in algorithm *Ag* is identical to the *LS* algorithm. Let task A_l be the task used to determine the time T. Then, we have $T-a_l\leqslant\frac{1}{2}\left(\sum_{i=4}^{n}a_i-a_l\right)$ by the assignment of A_l, which yields that $T\leqslant\frac{1}{2}\sum_{i=4}^{n}a_i+\frac{1}{2}a_4$ with $a_l\leqslant a_4$. Combining $a_1>a_2+b_2$ and $a_1+\sum_{i=4}^{n}a_i\leqslant a_2+b_2+a_3+b_3$ in Condition 1, we obtain $\sum_{i=4}^{n}a_i<a_3+b_3$.

It follows that $T\leqslant\frac{1}{2}\sum_{i=4}^{n}a_i+\frac{1}{2}a_4\leqslant\frac{1}{4}\sum_{i=4}^{n}a_i+\frac{1}{4}(a_3+b_3)+\frac{1}{2}a_4$. It is clear that $a_4\leqslant\frac{1}{2}(a_1+a_2)$ due to $a_4\leqslant a_2\leqslant a_1$. Thus, $T\leqslant\frac{1}{4}\sum_{i=4}^{n}a_i+\frac{1}{4}(a_3+b_3)+\frac{1}{4}(a_1+a_2)=\frac{1}{4}\sum_{i=1}^{n}a_i+\frac{1}{4}b_3\leqslant\frac{1}{4}\sum_{i=1}^{n}(a_i+b_i)$, implying $T\leqslant\frac{1}{2}C^{*}$ by Theorem 2.1.

Now, we present the improved algorithm H_2:

Algorithm H_2:

1. Re-index all jobs in set $\mathcal{J}$ with respect to the values of a_i such that $a_1\geqslant a_2\geqslant\ldots\geqslant a_n$.

2. If $\mathcal{J}$ satisfies *Condition* 1, run the algorithm *Ag*. Otherwise, run the Algorithm H_1.

From the proof of Theorem 2.2, Lemmas 2.3 and 2.4, the main result of this subsection can be obtained directly.

Theorem 2.5. The worst-case ratio of algorithm H_2 is at most 8/5.

The worst-case ratio of 8/5 yielded by algorithm H_2 is tight. Consider the instance $\mathcal{J}=\{J_1,J_2,J_3,J_4,J_5\}$ where $a_1=2L$, $b_1=\varepsilon$; $a_2=L$, $b_2=\varepsilon$; $a_3=L-3\varepsilon$, $b_3=\varepsilon$; $a_4=L-4\varepsilon$, $b_4=\varepsilon$; $a_5=\varepsilon$, $b_5=5L$ (Where $\varepsilon<<L$).

It is optimal to process $J_1,\ldots,J_4$ by mode 2 and J_5 by mode 1 with the order of J_5,J_4,J_3,J_2,J_1. Then, we have $C^{*}=5L+5\varepsilon$, while $C^{H_2}=8L-3\varepsilon$. It follows that $C^{H_2}/C^{*}=\frac{8L-3\varepsilon}{5L+5\varepsilon}\to\frac{8}{5}$ when ε tends to 0.

From the Algorithm H_2, we believe that it is possible to design an algorithm with worst-case ratio 3/2 by presenting more detailed processing method for subcase 2.2 with t=4,5,6,…, though it may be very complicated and hard.

3 NSHFS

3.1 Computational complexity

Theorem 3.1. The NSHFS is NP-hard.

Proof: We show the result by reducing the 3-partitioning problem to this problem. Given an instance 3*P* of the 3-partitioning problem with a set of positive integers $A=\{c_1,c_2,\ldots,c_{3m}\}$ and $\sum_{i=1}^{3m}c_i=mB$, $B/4<c_i<B/2$, we construct an instance *NS* of the NSHFS as follows:

$$J_i: a_i=\varepsilon, b_i=c_i-\varepsilon, \quad i=1,2,...,3m \ (\varepsilon<B/(3m)),$$
$$J_i: a_i=B+1, b_i=1, \quad i=3m+1,3m+2,...,4m+1,$$
$$J_{4m+2}: a_{4m+2}=1, b_{4m+2}=B.$$

Let $K=(m+1)B+(m+2)$. We claim that the instance 3*P* has a solution if and only if the instance *NS* has a solution with the objective value of no more than *K*.

If 3*P* has a solution, that is, there exist *m* subsets $S_1,S_2,\ldots,S_m$ of *A* such that $S_1\cup S_2\cup\ldots\cup S_m=A$, $S_i\cap S_j=\varnothing$, $1\leqslant i,j\leqslant m$, $i\neq j$ and $\sum_{c_i\in S_j}c_i=B$, then we construct a solution for instance *NS*, which processes jobs $J_1,J_2,\ldots,J_{3m}$ by mode 1 and $J_{3m+1},J_{3m+2},\ldots,J_{4m+1}$ by mode 2 in order of their subscript and processes J_{4m+2} by mode 1 at time zero on M_2. Let $S'_j=\{J_i\mid c_i\in S_j\}$ and process the jobs in S'_j between J_{3m+j} and J_{3m+j+1} ($j=1,2,\ldots,m$) (see Fig.7). It is clear that $a_{3m+1}=a_{4m+2}+b_{4m+2}$ and for any $1\leqslant j\leqslant m$, $a_{3m+j+2}-b_{3m+j}=B=\sum_{c_i\in S_j}c_i=\sum_{J_i\in S'_j}(a_i+b_i)$ (i.e., the total size of the jobs in S'_j). It follows that the makespan is $\sum_{i=3m+1}^{4m+1}a_i+1=(m+1)B+m+2=K$, which yields a solution of *NS* as a result.

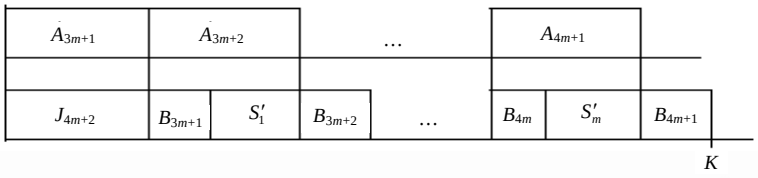


Fig.7 A solution for instance *NS*

Before going to construct a solution of 3*P* from that of *NS*, we first give a proposition as follows:

Proposition 3.2. Any solution of *NS* with makespan at most *K* must satisfy:

(i) Jobs $J_{3m+1},J_{3m+2},\ldots,J_{4m+1}$ must be processed by Mode 2;

(ii) Jobs $J_1,J_2,\ldots,J_{3m}$ and J_{4m+2} must be processed by Mode 1;

(iii) Job J_{4m+2} must be processed on M_2 before J_{3m+1}.

Proof: (i) If there is a job J_j, $3m+1\leqslant j\leqslant 4m+1$ processed by mode 1, then the completion time of M_2 is not less than $a_j+\sum_{i=1}^{4m+2}b_i=B+1+B+m+1+\sum_{i=1}^{3m}c_i-3m\varepsilon=(m+2)B+m+2-3m\varepsilon$, which is strictly greater than *K* due to the assumption of $\varepsilon<\frac{B}{3m}$, a contradiction. With loss of generality, we assume these jobs are processed in order of their subscript. Then, we can conclude that the completion time of these jobs is at least

$$\sum_{i=3m+1}^{4m+1}a_i+b_{4m+1}=K \tag{4}$$

(ii) Suppose there is a job J_j, $1\leqslant j\leqslant 3m$ or $j=4m+2$ processed by mode 2. From (i) and the Eq.(4), we obtain that the makespan is not less than $\begin{cases}K+\min\{a_{4m+2},b_{4m+2}\}=K+1>K, & \text{if } j=4m+2\\ K+\min\{a_i,b_i\}=K+\varepsilon>K, & \text{if } 1\leqslant j\leqslant 3m\end{cases}$ a contradiction too.

(iii) Suppose J_{4m+2} is processed between J_{3m+j} and J_{3m+j+1}, $1\leqslant j\leqslant m$, or after J_{4m+1} on M_2. If J_{4m+2} is processed after J_{4m+1}, combing it with Eq.(4), we conclude that the makespan is not less than $K+b_{4m+2}=K+B>K$, a contradiction. If it is processed between J_{3m+j} and J_{3m+j+1}, $1\leqslant j\leqslant m$, then the completion time of A_{3m+j+1}, i.e., the start time of B_{3m+j+1} (see Fig.8), is at least $\sum_{i=3m+1}^{3m+j}a_i+b_{3m+j}+a_{4m+2}+b_{4m+2}=\sum_{i=3m+1}^{3m+j}a_i+B+2$. Hence, the makespan is at least $\sum_{i=3m+1}^{3m+j}a_i+B+2+\sum_{i=3m+j+2}^{4m+1}a_i+b_{4m+1}=\sum_{i=3m+1}^{4m+1}a_i-a_{3m+j+1}+B+3=K+1>K$, which yields the last contradiction.

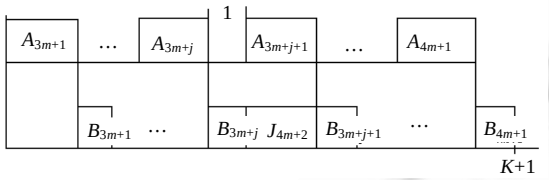


Fig.8 J_{4m+2} is processed between J_{3m+j} and J_{3m+j+1}

Now, we continue to prove our result. By Proposition 3.2, we can obtain one best possible schedule for J_i, $3m+1\leqslant i\leqslant 4m+2$, as described as Fig.9.

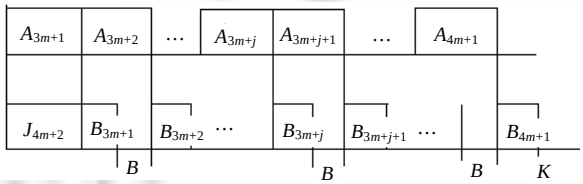


Fig.9 One optimal schedule for J_i, $3m+1\leqslant i\leqslant 4m+2$

From Eq.(4), in order to obtain the makespan no more than K, we must partition $\{J_1,J_2,\ldots,J_{3m}\}$ into m subsets S'_j with $\sum_{J_i\in S'_j}(a_i+b_i)\leqslant B$, $1\leqslant j\leqslant m$, and schedule all jobs in S'_j between J_{3m+j} and J_{3m+j+1}. Because $a_i+b_i=c_i$ and $\sum_{i=1}^{3m}c_i=mB$, we have $\sum_{J_i\in S'_j}(a_i+b_i)=\sum_{J_i\in S'_j}c_i=B$, $j=1,2,\ldots,m$. Let $S_j=\{c_i\mid J_i\in S'_j\}$. Then, we have $\sum_{c_i\in S_j}c_i=\sum_{J_i\in S'_j}c_i=B,\ j=1,2,...,m$, which implies that $S_1,S_2,\ldots,S_m$ is a solution of 3P.

If it takes time for constructing NS, then NSHFS is NP-hard.

Recall that, in Ref.[1], there exists an optimal algorithm for the problem SHFS if the processing mode of every job is known in advance (that is, the job set $\mathcal{J}$ has been partition into two sets V_1 and V_2 defined in section 2). For the pure two-stage two machines flowshop with no-waited problem, denoted by TTMF, if the job processing mode of every job is given in advance, there also exists an optimal algorithm presented by Gilmore and Gomory[5]. The difference between NSHFS and TTMF is that in NSHFS both two tasks of job can be processed on M_2, while in TTMF two tasks of job must be processed on different machines. However, the following theorem shows that the problem NSHFS is still NP-hard, even if the processing mode of every job is known in advance. Hence, we can conclude that, in some sense, the problem NSHFS is more difficult than the other two problems.

Theorem 3.3. NSHFS is NP-hard if the job processing mode of every job is given in advance.

Proof: We also show the result by reducing the 3-partitioning problem to this problem. Given the instance 3P, we construct an instance of the NSHFS with the processing mode of every job as follows:

$$J_i: a_i + b_i = c_i, \quad \text{processed by Mode 1}, i = 1,2,...,3m\ (\varepsilon < B/(3m)),$$
$$J_i: a_i = B+1, b_i = 1, \quad \text{processed by Mode 2}, i = 3m+1, 3m+2, ..., 4m+1,$$
$$J_{4m+2}: a_{4m+2} + b_{4m+2} = B+1, \quad \text{processed by Mode 1}.$$

Let $K=(m+1)B+(m+2)$. The following proof is just similar to that of Theorem 3.1.

3.2 A polynomial time approximation algorithm for NSHFS

Let C^H and C^* be the makespan yielded by algorithm H and an optimal algorithm for problem NSHFS, respectively. It is clear that the optimal makespan of NSHFS is not less than that of SHFS. Then, Theorem 2.1 is still true for problem NSHFS. In this section, we continue to use the notation V_1 and V_2 defined in section 2.

For this problem, the main idea of the algorithm also includes two parts: partitioning $\mathcal{J}$ into V_1 and V_2 and giving the processing order of jobs. The detailed can described as follows:

Algorithm H_3:

1. Re-index all jobs in set $\mathcal{J}$ with respect to the values of a_i such that $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_n$;
2. If $a_1 \geqslant \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$, put jobs $J_2,J_3,\ldots,J_n$ into set V_1, and job J_1 into V_2. Then, process jobs in order of $J_2,J_3,\ldots,J_n,J_1$ (see Fig.10). Stop.
3. If $a_1 < \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$, we do:
   3.1. If n is an even number, let $m = n/2$, put jobs J_{2i-1}, i=1,2,…,m into set V_2 and jobs J_{2i}, i=1,2,…,m into set V_1. Then, process the jobs in order of $J_2,J_1,J_4,J_3,\ldots,J_{2i+2},J_{2i+1},\ldots,J_{2m},J_{2m-1}$ (see Fig.11(a)). Stop.
   3.2. If n is an odd number, let $m = (n-1)/2$, put jobs J_{2i-1}, i=1,2,…,m+1 into set V_2 and jobs J_{2i}, i=1,2,…,m into set V_1. Then, process jobs in order of $J_2,J_1,J_4,J_3,\ldots,J_{2i+2},J_{2i+1},\ldots,J_{2m},J_{2m-1},J_{2m+1}$ (see Fig.11(b)). Stop.

Since it is clear that the time complexity of Step 1 is $O(n\log n)$, and the other steps run in $O(n)$, then the time complexity of algorithm H_3 is $O(n\log n)$.

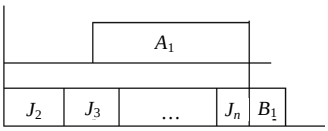


Fig.10 Case of $a_1 \geqslant \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$

Theorem 1. $C^{H_3}/C^* \leqslant 5/3$ and the bound is tight.

Proof: We distinguish two cases $a_1 \geqslant \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$ and $a_1 < \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$ to obtain the desired worst-case ratio.

Case 1. $a_1 \geqslant \frac{1}{6}\sum_{i=1}^{n}(a_i + b_i)$. Then from step 2 and Fig.10, it iseasy to obtain that

$$C^{H_3} = \max\left\{\sum_{i=1}^{n}(a_i + b_i) - a_1, a_1 + b_1\right\}.$$

If $C^{H_3} = a_1 + b_1$, then $C^* \geqslant a_1 + b_1 = C^{H_3}$ from Theorem 2.1, which yields $C^* = C^{H_3}$.

If $C^{H_3}=\sum_{i=1}^{n}(a_i+b_i)-a_1$, by the assumption of $a_1\geqslant\frac{1}{6}\sum_{i=1}^{n}(a_i+b_i)$, we obtain that

$$C^{H_3}=\sum_{i=1}^{n}(a_i+b_i)-a_1\leqslant\left(1-\frac{1}{6}\right)\sum_{i=1}^{n}(a_i+b_i)\leqslant\frac{5}{3}C^*.$$

Case 2. $a_1<\frac{1}{6}\sum_{i=1}^{n}(a_i+b_i)$. If *n* is an even number, then from step 3.1 and Fig.11(a), we have

$$C^{H_3}=\max\{a_2+b_2,a_1\}+\sum_{i=2}^{m}\max\{a_{2i}+b_{2i}+b_{2i-3},a_{2i-1}\}+b_{2m-1}.$$

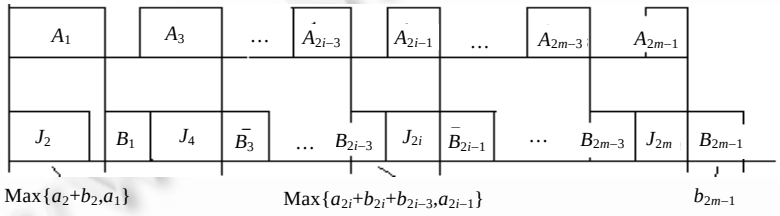
Since $a_1\geqslant a_2\geqslant\ldots\geqslant a_n$, then

$$\begin{aligned}C^{H_3}&\leqslant(a_1+b_1)+\sum_{i=2}^{m}(a_{2i-1}+b_{2i}+b_{2i-3})+b_{2m-1}=\sum_{i=2}^{m}a_{2i-1}+\sum_{i=1}^{n}b_i+a_1\\&\leqslant\sum_{i=2}^{m}\frac{1}{2}(a_{2i-2}+a_{2i-1})+\frac{1}{2}a_1+\sum_{i=1}^{n}b_i+\frac{1}{2}a_1<\frac{1}{2}\sum_{i=1}^{n}a_i+\sum_{i=1}^{n}b_i+\frac{1}{2}a_1\end{aligned}\tag{5}$$

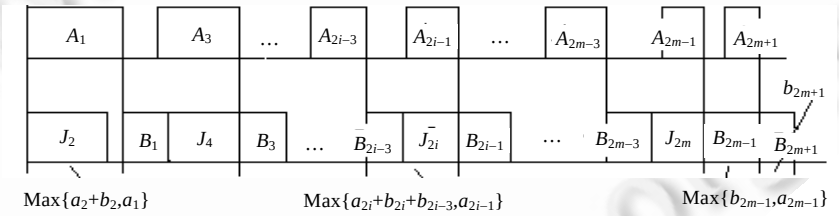
If *n* is an odd number, then from step 3.2 and Fig.11(b), we can obtain that

$$C^{H_3}=\max\{a_2+b_2,a_1\}+\sum_{i=2}^{m}\max\{a_{2i}+b_{2i}+b_{2i-3},a_{2i-1}\}+\max\{b_{2m-1},a_{2m+1}\}+b_{2m+1}.$$

By the similar argument for inequality (5), we can also obtain that $C^{H_3}\leqslant\frac{1}{2}\sum_{i=1}^{n}a_i+\sum_{i=1}^{n}b_i+\frac{1}{2}a_1$.



(a)



(b)

Fig.11 Case of $a_1<\frac{1}{6}\sum_{i=1}^{n}(a_i+b_i)$

According to $a_1<\frac{1}{6}\sum_{i=1}^{n}(a_i+b_i)$ and Theorem 2.1, we have $a_1<\frac{1}{3}C^*$. Hence, from the inequality (5) and Theorem 2.1, we obtain $C^{H_3}\leqslant\frac{1}{2}\sum_{i=1}^{n}(a_i+b_i)+\frac{1}{2}\sum_{i=1}^{n}b_i+\frac{1}{2}a_1\leqslant C^*+\frac{1}{2}C^*+\frac{1}{6}C^*=\frac{5}{3}C^*$.

To show that the worst-case ratio of 5/3 is tight, we consider the instance $\mathcal{J}=\{J_1,J_2,\ldots,J_6\}$, where $a_i=L-(i-1)\varepsilon$, $b_i=\varepsilon$ for i=1,2,3, $a_i=L-i\varepsilon$, $b_i=\varepsilon$ for i=4,5, and $a_i=L-(i+1)\varepsilon$, $b_i=\varepsilon$ for i=6 (where $\varepsilon\ll L$). It is not hard to obtain that it is optimal to process J_i, i=1,3,5 by Mode 2 and the remainder by Mode 1. Then, process them in order of J_2,J_1,J_4,J_3,J_6,J_5. We can obtain that $C^*=3L-6\varepsilon$, while $C^{H_3}=5L-13\varepsilon$. It follows that $C^{H_3}/C^*=\frac{5L-13\varepsilon}{3L-6\varepsilon}\to\frac{5}{3}$ when ε tends

to 0.

References:

[1] Wei Q, He Y. A two-stage semi-hybrid flowshop problem in graphics processing. Applied Mathematics A Journal of Chinese Universities, 2005,20(4):393−400. [doi: 10.1007/s11766-005-0016-6]

[2] Kouvelis P, Vairaktarakis G. Flowshops with processing flexibility across production stages. IIE Trans., 1998,30(8):735−746. [doi: 10.1023/A:1007552024802]

[3] Vairaktarakis GL, Lee CY. Analysis of algorithms for two-stage flowshops with multi-processor task flexibility. Naval Research Logistics, 2004,51(1):44−59. [doi: 10.1002/nav.10104]

[4] Johnson SM. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1954,1(1):61−68. [doi: 10.1002/nav.3800010110]

[5] Gilmore P, Gomory RE. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. Operations Research, 1964,12(5):655−679. [doi: 10.1287/opre.12.5.655]

WEI Qi was born in 1980. He is a lecturer at the Ningbo Institute of Technology, Zhejiang University. His research areas are scheduling theory and combinatorial optimization.

JIANG Yi-Wei was born in 1980. He is an associate professor at School of Science, Zhejiang Sci-Tech University. His research areas are scheduling theory, combinatorial optimization and design and analysis of algorithms.