Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On scheduling inclined jobs on multiple two-stage flowshops ☆

Guangwei Wu [a,b], Jianer Chen [c,d], Jianxin Wang [a,*]

[a] School of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China
[b] College of Computer and Information Engineering, Central South University of Forestry and Technology, Changsha, Hunan 410004, China
[c] School of Computer Science & Education Software, Guangzhou University, Guangzhou, Guangdong 510006, China
[d] Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA

ARTICLE INFO

Article history:
Received 18 January 2018
Received in revised form 28 March 2018
Accepted 5 April 2018
Available online xxxx

Keywords:
Scheduling
Multiple two-stage flowshops
Online algorithms
Approximation algorithms
Cloud computing

ABSTRACT

We study scheduling on multiple two-stage flowshops in which each job has to pass through an R-operation and a T-operation. Motivated by the current research in data centers, we consider two restricted versions of the problem in which the jobs are inclined: one restricts that for each job, the R-operation consumes no less time than the T-operation, while the other assumes that the T-operation consumes no less time than the R-operation for each job. For the first case, we present an online 2-competitive algorithm and an offline 11/6-approximation algorithm. For the second case, we give an online 5/2-competitive algorithm, and prove, for the offline setting, that the problem can be reduced to the problem in the first case.

1. Introduction

A job is *two-stage* if it consists of an R-operation and a T-operation. Correspondingly, a flowshop is *two-stage* if it contains an R-processor and a T-processor. If a job is assigned to a flowshop, then the T-operation of the job cannot start on the T-processor of the flowshop until the R-operation of the job is finished on the R-processor of the same flowshop. Scheduling a set of two-stage jobs on multiple two-stage flowshops is to assign the jobs to the flowshops, and for each flowshop, to determine the execution order of the R-operations and the T-operations of the jobs assigned to the flowshop.

This paper will focus on schedulings whose objective is to minimize the makespan. Especially, two restricted versions of the problem, in which jobs are assumed to be inclined, are considered. The first version assumes that for each job the R-operation consumes no less time than the T-operation, while the second version restricts that the T-operation of each job consumes no less time than the R-operation of the same job.

These scheduling models are motivated by the current research in data centers. A modern data center contains hundreds of thousands of servers, and each server contains processors, network interface, and local I/O, etc. [1]. Software and data are stored as resources in the servers. Clients dynamically request the resources, and servers send the resources to the clients over networks. When a request arrives at a server, the server needs to read the resource from the secondary memory to the main memory first, and then send the resource to the client over networks. Therefore, each request needs to pass through two-stage operations, i.e., disk-read (the R-operation) and network-transmission (the T-operation), in such a way that the

☆ A preliminary version of this work was reported in *Proc. 11th International Workshop on Frontiers in Algorithmics*, *Lecture Notes in Computer Science 10336*, pp. 241–253, 2017.

* Corresponding author.
E-mail addresses: will99031827@hotmail.com (G. Wu), chen@cs.tamu.edu (J. Chen), jxwang@csu.edu.cn (J. Wang).

https://doi.org/10.1016/j.tcs.2018.04.005
0304-3975/

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

network-transmission cannot start until the disk-read is completed. Thus, each server can be regarded as a two-stage flowshop, which can execute the R-operations and the T-operations of different jobs in parallel. As a result, scheduling such requests on servers in data centers to minimize the makespan is exactly the scheduling model studied in the current paper.

The setting of servers in data centers is chosen according to the provided services [3]. For certain services that require high reliability of data, disk I/O transfer-rate may be much lower than that of network, i.e., the R-operation will cost more time than the T-operation. On the other hand, services such as many web services have high I/O rate requirement to tackle with the huge requests from clients. In this circumstance, the R-operation is less expensive than the T-operation. As a result, if the servers in a cloud are divided into clusters based on the services they provide, then the requests to the servers in each cluster will be likely to be inclined towards one side: either all with more expensive R-operations or all with more expensive T-operations.

Our scheduling models are generalizations of the classical Makespan problem, which schedules n (one-stage) jobs on m (one-stage) machines to minimize the makespan. The Makespan problem is NP-hard even when m is a constant, and becomes strongly NP-hard when m is part of the input [10]. As a consequence, both versions of the scheduling on multiple two-stage flowshops studied in the current paper are strongly NP-hard. When the number of two-stage flowshops is 1, the problem becomes the classical two-stage flowshop problem, which can be solved in time $O(n \log n)$ [15].

We will focus on approximation algorithms and competitive algorithms for the problems described above. An algorithm is an *α-approximation algorithm* for a scheduling problem, with an *approximation ratio* α, if for each instance of the problem, the algorithm constructs a schedule whose makespan is bounded by α times the minimum makespan of the instance [20]. Scheduling problems where the input is received online and the output must be given online are called *online scheduling problems*, and algorithms solving the problems are called *online algorithms*. An online algorithm is *α-competitive*, with a *competitive ratio* α, if for each finite input sequence I of a problem, the algorithm constructs an online schedule whose makespan is within α times the minimum makespan of the online schedules for the sequence I [5].

For the classical Makespan problem, *ListRanking* is a well-known algorithm for both online and offline settings, which will also impact our algorithms for the new scheduling models. The *ListRanking* algorithm was proposed by Graham [11], who proved that the algorithm achieves a $(2-\frac{1}{m})$-approximation ratio, where m is the number of machines. Later Graham improved the approximation ratio to $(\frac{4}{3}-\frac{1}{3m})$ by sorting the jobs before *ListRanking* [12]. *ListRanking* is also the first online algorithm for the problem. The competitive ratio $(2-\frac{1}{m})$ stood for a long time, until Galambos and Woeginger provided a $(2-\frac{1}{m}-\epsilon_m)$-competitive algorithm, where ϵ_m is a positive constant for $m \geq 4$, but tends to 0 with the growth of m [9]. Bartal et al. presented a 1.986-competitive algorithm for $m > 70$, that first achieved a competitive ratio $2-\epsilon$ for a positive constant ϵ [4]. Albers presented a 1.923-competitive algorithm based on a new strategy for $m \geq 2$ [2].

The multiple two-stage flowshop problem, that schedules n two-stage jobs on m two-stage flowshops, had not been studied throughly until very recently, and most of the studies were focused on the problem when the number m of flowshops is a fixed constant. He et al. [14] first studied the problem, motivated by applications in glass manufacturing, and gave a mixed-integer programming formulation and a heuristic algorithm for the problem. Vairaktarakis et al. [19] studied the problem in their work on hybrid flowshops, and proposed a formulation that leads to a pseudo-polynomial time algorithm for $m = 2$. Zhang et al. [23] presented constant ratio approximation algorithms for scheduling on two and three two-stage flowshops. The approximation ratio of these algorithms can reach 3/2 for $m = 2$ and 12/7 for $m = 3$. Using the formulation similar to that of [19], Dong et al. [8] (see also [7]) presented a pseudo-polynomial time algorithm and developed a fully polynomial-time approximation scheme for the problem. Wu et al. [21] proposed a new formulation for the problem that leads to a new fully polynomial-time approximation scheme for the problem with improved running time. Wu et al. [21] also studied algorithms for the cases where the costs of the two stages differ significantly. Approximation algorithms for k-stage jobs on multiple k-stage flowshops for general k have also been studied [18].

Very recently, Wu et al. [22] dealt with the problem where the number m of the flowshops is not a fixed constant, in this case the problem becomes strongly NP-hard. A 2.6-approximation algorithm is proposed [22]. To the best of our knowledge, this is the first approximation result for multiple-flowshop scheduling when the number m of flowshops is part of the input.

In the current paper, we will also consider the case where the number m of flowshops is part of the input, but will be focused on two restricted versions of the problem: one restricts that the R-operation consumes no less time than the T-operation for each job, while the other assumes that the T-operation consumes no less time than the R-operation for each job. For the first case, we present an online 2-competitive algorithm and an offline 11/6-approximation algorithm, for arbitrary m. For the second case, we give an online 5/2-competitive algorithm for arbitrary m, and prove, for the offline setting, that the problem can be reduced to the problem in the first case.

2. Scheduling on a single two-stage flowshop

Let $G = \{J_1, \ldots, J_n\}$ be a set of two-stage jobs to be scheduled in a system $\{M_1, \ldots, M_m\}$ of m identical two-stage flowshops. We assume

- each job $J_i = (r_i, t_i)$ consists of an R-operation of cost r_i (i.e. the *R-time*) and a T-operation of cost t_i (i.e. the *T-time*);
- each flowshop has an R-processor and a T-processor that can run in parallel and can process the R-operations and the T-operations, respectively, of the assigned jobs;

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Algorithm 1 Assign a job $J_i = (r_i, t_i)$ to a flowshop M_q.

1: $\rho_q = \rho_q + r_i$;
2: $\tau_q = \max(\rho_q, \tau_q) + t_i$.

- the R-operation and T-operation of a job must be executed in the R-processor and T-processor, respectively, of the same flowshop, in a way that the T-operation cannot start unless the R-operation is completed;
- there are no precedence constraints among the jobs; and
- preemption is not allowed.

A *schedule* $\mathcal{S}$ assigns all jobs in the given input $G = \{J_1, \ldots, J_n\}$ to flowshops, and for each flowshop, determines the execution order of the R- and T-operations of the jobs assigned to that flowshop. The *completion time* of a flowshop M under the schedule $\mathcal{S}$ is the time when M finishes the last operation of the jobs assigned to M. The *makespan* $C_{\max}$ of $\mathcal{S}$ is the largest flowshop completion time over all flowshops. In this paper, the number m of flowshops is part of the input. Following the three-field notation $\alpha|\beta|\gamma$ suggested by Graham et al. [13], the general multiple two-stage flowshop scheduling problem can be written as $P|2\text{FL}|C_{\max}$, as $P|2\text{FL}_{R\geq T}|C_{\max}$ if $r_i \geq t_i$ for all jobs (r_i, t_i) in the input (i.e., our first version), and as $P|2\text{FL}_{R\leq T}|C_{\max}$ if $r_i \leq t_i$ for all jobs (r_i, t_i) in the input (i.e., our second version).

Like most work in the literature on flowshop scheduling, our algorithms are based on permutation scheduling [17]. Thus, a schedule on a flowshop can be given as an ordered sequence $\langle J_1, \ldots, J_t\rangle$ of two-stage jobs where the R- and T-operations of the jobs are executed by the R- and T-processors of the flowshop, respectively, in the way that strictly follows the given order of the sequence. Let $\bar{\rho}_i$ and $\bar{\tau}_i$, respectively, be the times when the R-operation and the T-operation of job J_i are started. The following lemma holds if the objective of scheduling is to minimize the makespan (where $\bar{\tau}_0 = t_0 = 0$).

Lemma 2.1. ([21]) *Let $\mathcal{S} = \langle J_1, J_2, \ldots, J_t\rangle$ be a two-stage job schedule on a two-stage flowshop, where $J_i = (r_i, t_i)$, for $1 \leq i \leq t$. Then for all i, $1 \leq i \leq t$, we can assume: $\bar{\rho}_i = \sum_{j=1}^{i-1} r_j$; and $\bar{\tau}_i = \max\{\bar{\rho}_i + r_i, \bar{\tau}_{i-1} + t_{i-1}\}$.*

According to Lemma 2.1, the execution of the R-processor of a flowshop is continuous. The R-operation of a job starts as soon as the R-operations of previous jobs on the flowshop are finished. On the other hand, the way in which the T-processor executes is different. The T-operation of a job J_i cannot start unless not only the previous T-operations are finished but also the R-operation of J_i is finished. Thus, the execution of the T-processor may not be continuous, and each "gap" in the execution means that there is a job whose T-operation is waiting for its own R-operation to complete.

Let ρ_q, τ_q be the finishing times of the R-processor and the T-processor of the flowshop M_q, respectively, when the jobs currently assigned to M_q are finished. By Lemma 2.1, we can easily update ρ_q and τ_q of M_q when assigning a new job to it. The procedure is described in Algorithm 1, which will serve as a subroutine for our scheduling algorithms.

For a two-stage job $J_i = (r_i, t_i)$, the *dual job* $J_i^d = (t_i, r_i)$ of J_i is obtained from J_i by swapping its R-time and T-time. Let $\mathcal{S} = \langle J_1, J_2, \ldots, J_n\rangle$ be a schedule of a two-stage job set $G = \{J_1, J_2, \ldots, J_n\}$ on a two-stage flowshop. Define by $\mathcal{S}^d = \langle J_n^d, \ldots, J_2^d, J_1^d\rangle$ the *dual schedule* of $\mathcal{S}$ on the *dual job set* $G^d = \{J_1^d, J_2^d, \ldots, J_n^d\}$, where J_i^d is the dual job of the job J_i, for $1 \leq i \leq n$.

Theorem 2.2. ([21]) *On a single two-stage flowshop, the optimal schedule of the job set $G = \{J_1, J_2, \ldots, J_n\}$ and the optimal schedule of the dual job set $G^d = \{J_1^d, J_2^d, \ldots, J_n^d\}$, where J_i^d is the dual job of J_i for $1 \leq i \leq n$, have the same completion time. Moreover, if a schedule $\mathcal{S}$ is optimal for the job set G then its dual schedule $\mathcal{S}^d$ is optimal for the dual job set G^d.*

Suppose that $\mathcal{S}$ is a schedule of the job set $G = \{J_1, J_2, \ldots, J_n\}$ on m two-stage flowshops, where for each i, $\mathcal{S}$ assigns a subset G_i of the job set G to the i-th flowshop. By replacing the schedule of G_i on the i-th flowshop by its dual schedule of the dual job set G_i^d for each i, we get a schedule $\mathcal{S}^d$ on m flowshops for the dual job set G^d of G. By Theorem 2.2 and the above discussion, it is easy to see that the completion times of each flowshop of the two schedules are the same. Therefore, we have the following result.

Theorem 2.3. ([21]) *On multiple two-stage flowshops, the optimal schedule of a job set G and the optimal schedule of the dual job set G^d have the same makespan. Moreover, an optimal schedule for the job set G can be easily obtained from an optimal schedule for the dual job set G^d.*

3. Scheduling on model $P|2\text{FL}_{R\geq T}|C_{\max}$

In the rest of this paper, we assume that we are given n two-stage jobs $G = \{\bar{J}_1, \bar{J}_2, \ldots, \bar{J}_n\}$, and m two-stage flowshops $M_1, \ldots, M_m$. In this section, we consider the problem $P|2\text{FL}_{R\geq T}|C_{\max}$, assuming the R-operation consumes no less time than the T-operation for each job. This case corresponds to the situation where the major time consumed by each job comes from data reading from server disks, rather than from data transmission over networks.

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Algorithm 2 The online algorithm for $P|2\text{FL}_{R\geq T}|C_{\max}$.

1: When a job $\bar{J}_i$ arrives, assign $\bar{J}_i$ to the flowshop M_q with the minimum ρ-value;
2: update the ρ and τ values of M_q using Algorithm 1;

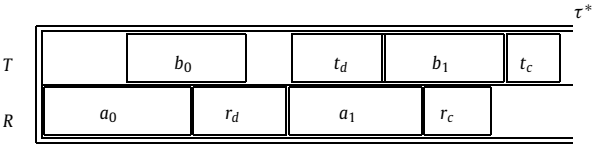

Fig. 1. The configuration of the flowshop M_{h^*} in Algorithm 2 for $P|2\text{FL}_{R\geq T}|C_{\max}$.

3.1. An online algorithm

First consider the online version of the model, where jobs come strictly following the ordered sequence $G = \langle \bar{J}_1, \bar{J}_2, \ldots, \bar{J}_n \rangle$, and a scheduling algorithm has to decide which flowshop to assign a job to when the job arrives without any information about the latter jobs.

We will always use symbols ρ and τ, respectively, for the finishing time of the R-processor and for the finishing time of the T-processor of a flowshop. In particular, for the flowshop M_q, we will use ρ_q and τ_q for the current finishing times of the R-processor and the T-processor of M_q, respectively, based on the jobs currently assigned to M_q. The online algorithm is given in Algorithm 2. The main idea is that for each new coming job $\bar{J}_i$, the algorithm assigns $\bar{J}_i$ to the flowshop with the smallest ρ-value. For this, we need a data structure that supports efficient search for the flowshop with the smallest ρ-value. This can be achieved using a min-heap structure and organizing the flowshops in the heap based on their ρ-values. It is well-known that such a data structure supports finding the flowshop with the smallest ρ-value in time $O(1)$ and re-organizing the flowshops in the heap structure in time $O(\log m)$ after the ρ-value of a flowshop is changed [6]. This means, in terms of [5], that the algorithm is "effective".

We introduce some notations. Let $\text{Opt}(G)$ be the minimum makespan of scheduling the two-stage job sequence G in a given system of m two-stage flowshops. Assume that the schedule constructed by Algorithm 2 achieves its makespan on the flowshop M_{h^*}, in which the corresponding schedule is a job sequence $\langle J_1, \ldots, J_c \rangle$. Let τ^* be the completion time of M_{h^*}. Let d be the minimum job index such that the T-operations of the jobs J_d, J_{d+1}, …, J_c are executed continuously on M_{h^*} without an execution gap. Note that by the definition of d, the starting time of the T-operation of J_d must be equal to the completion time of the R-operation of J_d.

Let $a_0 = \sum_{i=1}^{d-1} r_i$ be the sum of the R-times of the jobs on M_{h^*} from J_1 to J_{d-1}, $b_0 = \sum_{i=1}^{d-1} t_i$ be the sum of the T-times of these jobs correspondingly. Similarly, $a_1 = \sum_{i=d+1}^{c-1} r_i$ and $b_1 = \sum_{i=d+1}^{c-1} t_i$. Fig. 1 illustrates the configuration of the flowshop M_{h^*} under the schedule. Note that we denote by b_0 the sum of the T-times of the jobs before J_d, and these T-operations may not be executed continuously. But this will not affect our analysis.

Lemma 3.1. *Let $J_i = (r_i, t_i)$ be any job in the job set G. Then the makespan $\text{Opt}(G)$ of an optimal schedule for G is at least $r_i + t_i$, and, under the model $P|2\text{FL}_{R\geq T}|C_{\max}$, is not smaller than $2t_i$.*

Proof. In the model of scheduling 2-stage flowshops, the two operations of a job must be executed in the same flowshop, and the T-operation cannot start until the R-operation is finished. Thus the fact that the optimal makespan $\text{Opt}(G)$ must be not smaller than the entire execution time for a single job in G implies $\text{Opt}(G) \geq r_i + t_i$. We also have $r_i \geq t_i$ under the model $P|2\text{FL}_{R\geq T}|C_{\max}$, thus $\text{Opt}(G) \geq 2t_i$ follows immediately. □

Lemma 3.2. *b_1 is not larger than a_1.*

Proof. By definition, a_1 and b_1 represent the sum of the R-times and the T-times, respectively, of the same set of jobs $\{J_{d+1}, \ldots, J_{c-1}\}$. The model $P|2\text{FL}_{R\geq T}|C_{\max}$ assumes that the R-time is not smaller than the T-time for each job. Therefore, b_1 is not larger than a_1. □

Lemma 3.3. *$\text{Opt}(G)$ is not smaller than $a_0 + r_d + a_1$.*

Proof. J_c is the last job scheduled on the flowshop M_{h^*}. According to Algorithm 2, M_{h^*} must have the minimum ρ-value over all flowshops before the job J_c was assigned. Therefore, the completion time of the R-processor of each flowshop must be not smaller than the ρ-value of M_{h^*} before J_c was assigned to M_{h^*}. The ρ-value of M_{h^*} at that time was $a_0 + r_d + a_1$ (see Fig. 1). Combining with the fact that by Lemma 2.1, the R-operations are executed continuously in each flowshop, we conclude that the sum of the R-times of the jobs in the input job set is at least $m(a_0 + r_d + a_1)$, which implies that on any

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Algorithm 3 An offline algorithm for $P|2\text{FL}_{R\geq T}|C_{\max}$.

1: sort all jobs into a job sequence $G = \langle \bar{J}_1, \bar{J}_2, \ldots, \bar{J}_n \rangle$ in non-increasing order by their R-times;
2: **for** $i = 1$ to n **do**
3: assign the job $\bar{J}_i$ to the flowshop M_q with the minimum ρ-value;
4: update the ρ and τ values of M_q using Algorithm 1;

schedule of the input job set, the completion time of the R-processor of at least one flowshop, thus the completion time of that flowshop, is at least $m(a_0 + r_d + a_1)/m = a_0 + r_d + a_1$. Thus, $\text{Opt}(G) \geq a_0 + r_d + a_1$. □

Based on the above lemmas, the competitive analysis of the online algorithm Algorithm 2 is given as follows:

Theorem 3.4. *Algorithm 2 is a 2-competitive algorithm for the problem $P|2\text{FL}_{R\geq T}|C_{\max}$ of scheduling two-stage jobs on multiple flowshops, where the number m of flowshops is part of the input and the jobs are given online.*

Proof. By our assumption, the makespan of the schedule constructed by Algorithm 2 is the completion time of the flowshop M_{h^*}, which is expressed as follows (see Fig. 1):

$$\begin{aligned}
\tau^* &= a_0 + r_d + t_d + b_1 + t_c && (1)\\
&\leq a_0 + r_d + t_d + a_1 + t_c && (2)\\
&= (a_0 + r_d + a_1) + t_d + t_c \\
&\leq \text{Opt}(G) + \frac{1}{2} \cdot \text{Opt}(G) + \frac{1}{2} \cdot \text{Opt}(G) && (3)\\
&= 2 \cdot \text{Opt}(G).
\end{aligned}$$

We explain the above derivations. As discussed above, the execution of the R-processor of any flowshop is continuous, and by the definition of the job index d, the T-operations of the jobs on M_{h^*} from J_d to J_c are executed continuously. Moreover, the T-operation of J_d starts, without a gap, right after the R-operation of J_d is finished: otherwise by Lemma 2.1, the T-operation of J_d would be waiting for the T-operation of the previous job J_{d-1} on M_{h^*} to complete at that time, and would start immediately after the T-operation of J_{d-1} is completed. Thus, the flowshop M_{h^*} would execute the T-operations of the jobs J_{d-1}, J_d, …, J_c continuously, which contradicts the minimality of the job index d. As a result, the equality (1) holds. Lemma 3.2 claims that $b_1 \leq a_1$, thus explains the inequality (2). By Lemma 3.3, $a_0 + r_d + a_1 \leq \text{Opt}(G)$. By Lemma 3.1, we also have $t_i \leq \text{Opt}(G)/2$ for each job J_i in this case. These explain the inequality (3). □

This competitive ratio almost matches that of the best online algorithm for the classical MAKESPAN problem $P||C_{\max}$ [16], which can be regarded as a simpler version of the problem $P|2\text{FL}_{R\geq T}|C_{\max}$ in which all jobs are one-stage jobs (i.e., $t_i = 0$ for all i) and all machines are one-stage flowshops.

3.2. An offline algorithm

In this section, we study the same problem $P|2\text{FL}_{R\geq T}|C_{\max}$ in the situation where the jobs are given offline, i.e., all jobs are known at the beginning. The offline version is also important for scheduling problems [5]. Unlike online algorithms that make a decision when a job arrives, an offline algorithm outputs a schedule for all jobs at once. An offline algorithm for scheduling on multiple two-stage flowshops for the model $P|2\text{FL}_{R\geq T}|C_{\max}$ is described in Algorithm 3.

The time complexity of the algorithm is given in the following theorem.

Theorem 3.5. *Algorithm 3 runs in time $O(n \log n)$.*

Proof. It takes time $O(n \log n)$ for the first step of the algorithm to sort the jobs into the sequence $G = \langle \bar{J}_1, \bar{J}_2, \ldots, \bar{J}_n \rangle$. When the number m of flowshops is not smaller than the number n of the jobs to be scheduled, Algorithm 3 gives a schedule in time $O(n)$, which places each job in a different flowshop and is obviously an optimal schedule. Thus, we only need to consider the case when $n > m$.

We again use a min-heap for the flowshops in terms of their ρ-values. For each job $\bar{J}_i$ in G, the min-heap supports finding in time $O(1)$ the flowshop M_q with the minimum ρ-value in step 3, and reconstructing the min-heap in time $O(\log m)$ after step 4 of updating the ρ and τ values of M_q in time $O(1)$ [6]. There are n jobs to be scheduled, thus the loop of steps 2-4 takes time $O(n \log m)$. Since $O(n \log m)$ is not larger than $O(n \log n)$ due to the fact that $n > m$, the total time complexity of Algorithm 3 is $O(n \log n)$. □

The main idea of Algorithm 3 is similar to that of Algorithm 2. When assigning a job $\bar{J}_i$, Algorithm 3 always picks the flowshop M_q with the minimum ρ-value over all flowshops. The difference between them is that in the offline algorithm,

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

the jobs are sorted into non-increasing order by their R-times in the first step, while such sorting cannot be done in online setting where jobs are coming online. As a consequence, after scheduling by Algorithm 3, the schedule on each flowshop must be a subsequence of the sorted job sequence G, thus also follows the non-increasing order in terms of the R-times.

For a two-stage job $J_i = (r_i, t_i)$, call $J_i^r = (r_i, 0)$ the *R-partial job* of J_i, i.e., the job J_i^r is constructed from the job J_i by setting its T-time to 0. Given a two-stage job set $G = \{J_1, \ldots, J_t\}$, its *R-partial job set* is $G^r = \{J_1^r, \ldots, J_t^r\}$, where J_i^r is the R-partial job of J_i, for $1 \le i \le t$. Similarly, for a two-stage job sequence (i.e., a schedule on a flowshop), its *R-partial job sequence* is obtained from the original sequence with each job in the original sequence replaced by its R-partial job. Denote by $\mathrm{Opt}(G)$ and $\mathrm{Opt}(G^r)$ the minimum makespans of scheduling G and G^r, respectively, on m flowshops. It is straightforward to see that $\mathrm{Opt}(G) \ge \mathrm{Opt}(G^r)$. We also have the following observation.

Lemma 3.6. *Let G be a two-stage job set and G^r be its R-partial job set. Let $\mathcal{S}$ and $\mathcal{S}^r$ be the schedules constructed by Algorithm 3 for G and G^r, respectively. Then for each flowshop M_j, $\mathcal{S}$ assigns a job sequence G_j in G to M_j if and only if $\mathcal{S}^r$ assigns G_j's R-partial job sequence G_j^r in G^r to M_j.*

Proof. Intuitively, this lemma is rather straightforward: (1) for each job, Algorithm 3 assigns the job to the flowshop with the minimum ρ-value; (2) the ρ-value of each flowshop is determined by the R-times of the jobs assigned to that flowshop; and (3) the R-times of the corresponding jobs in the job sets G and G^r are the same. However, there are certain details that are subtle for our later discussion and should be clarified. For this we give a formal proof for this lemma as follows.

First of all, assume that Algorithm 3 uses two rules to break ties (for all inputs). In step 1 of sorting in the algorithm, for two jobs with the same R-time, the job with a smaller job index will always be placed before the one with a larger job index. For step 3 of selecting a flowshop for a new job in the algorithm, we use a more complicated strategy: let M_i and M_j be two flowshops that both have the minimum ρ-value, (1) if the minimum ρ-value is 0 (i.e., if both M_i and M_j are empty), then the flowshop with a smaller flowshop index will be selected for the new job; and (2) if the minimum ρ-value is larger than 0 (i.e., if both M_i and M_j are not empty), then the flowshop with a larger flowshop index will be selected for the new job. Note that the above breaking-tie rules are not necessary but make our analysis (notationally) much simpler. Other ways of breaking ties used by the algorithm will give the same approximation ratio, but would make the analysis more complicated when referring to the desired job index and flowshop index.

Without loss of generality, let $\langle \bar{J}_1, \ldots, \bar{J}_n \rangle$ be the job sequence sorted in non-increasing order in R-times by Algorithm 3 on the input job set G. It is easy to see that its R-partial job sequence $\langle \bar{J}_1^r, \ldots, \bar{J}_n^r \rangle$ must be the sorted job sequence by Algorithm 3 on the job set G^r, because $\bar{J}_i$ and $\bar{J}_i^r$ have the same R-time for $1 \le i \le n$, and step 1 of Algorithm 3 uses the same rules to break ties. Now we will prove the lemma using induction on the number i of jobs, i.e., after assigning i jobs by Algorithm 3, for each flowshop M_j, a job sequence in G is assigned to M_j by $\mathcal{S}$ if and only if its R-partial job sequence in G^r is assigned to M_j by $\mathcal{S}^r$.

For the case $i = 1$, the claim holds true, since before assigning the first job, all flowshops have no assigned jobs, and the algorithm, in both $\mathcal{S}$ and $\mathcal{S}^r$, will always pick M_1 for $\bar{J}_1$ and $\bar{J}_1^r$, respectively. Suppose that after $i-1$ jobs have been assigned, for each flowshop M_j, a job sequence in G is assigned to M_j by $\mathcal{S}$ if and only if its R-partial job sequence in G^r is assigned to M_j by $\mathcal{S}^r$. Thus, for each flowshop M_j, the ρ-value of M_j in the schedule $\mathcal{S}$ is the same as that of M_j in the schedule $\mathcal{S}^r$, due to the fact that a job and its R-partial job have the same R-time. Now let M_q be the flowshop picked by the schedule $\mathcal{S}$ for the job $\bar{J}_i$, then (1) if the ρ-value of M_q is 0, then q must be the smallest flowshop index such that M_q has the ρ-value 0, and (2) if the ρ-value of M_q is larger than 0, then q must be the largest flowshop index such that M_q has the minimum ρ-value over all flowshops. Thus, in schedule $\mathcal{S}^r$, M_q is also the picked flowshop for the job $\bar{J}_i^r$. Note that the job sequences scheduled on other flowshops remain the same. Therefore after assigning i jobs in G and G^r by Algorithm 3 in the schedules $\mathcal{S}$ and $\mathcal{S}^r$, respectively, the claim still holds true. This completes the proof of the lemma. □

Let $\mathcal{S}$ be the schedule constructed by Algorithm 3 on the given job set G. In the following, we study the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$ and compare it with the makespan $\mathrm{Opt}(G)$ of an optimal schedule for G. For this, we first introduce some notations that will be used in the discussion.

Notations.

- Let M_{h^*} be the flowshop that achieves the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$ constructed by Algorithm 3 on the job set G, i.e., under the schedule $\mathcal{S}$, the completion time of the flowshop M_{h^*} is equal to the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$;
- Let $\langle J_1, J_2, \ldots, J_c \rangle$ be the job sequence assigned to the flowshop M_{h^*} by the schedule $\mathcal{S}$;
- Let d be the minimum job index such that the T-operations of the jobs J_d, J_{d+1}, …, J_c on the flowshop M_{h^*} are executed continuously;
- Let G_c be the job set obtained from the job set G by removing the jobs after J_c in the sorted sequence constructed in step 1 of Algorithm 3.

Lemma 3.7. *$Opt(G) \ge Opt(G_c)$. Moreover, the schedules $\mathcal{S}$ and $\mathcal{S}_c$ that are constructed by Algorithm 3 for the job sets G and G_c, respectively, have the same makespan, which is achieved on the flowshop M_{h^*} in both schedules.*

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

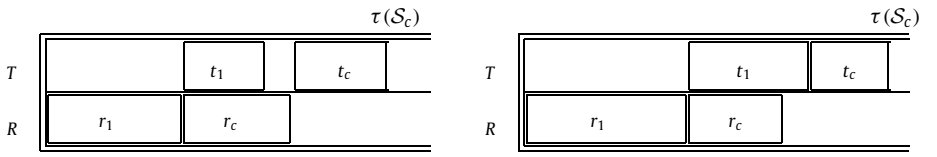

Fig. 2. The configuration of the flowshop M_{h^*} for $P|2\text{FL}_{R\geq T}|C_{\max}$ when $r_c > \text{Opt}(G)/3$.

Proof. To see $\text{Opt}(G) \geq \text{Opt}(G_c)$, observe that removing the jobs in $G \setminus G_c$ from an optimal schedule $\mathcal{S}_{\text{opt}}$ of the job set G gives a schedule for the job set G_c whose makespan is not larger than that of $\mathcal{S}_{\text{opt}}$, which is $\text{Opt}(G)$.

Now we prove the second part of the lemma. By our assumption, the schedule $\mathcal{S}$ assigns the job sequence $\langle J_1, J_2, \ldots, J_c\rangle$ in the job set G to the flowshop M_{h^*} that achieves the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$. By Algorithm 3, the schedule $\mathcal{S}_c$ for the job set G_c also assigns the same job sequence $\langle J_1, J_2, \ldots, J_c\rangle$ (which are in G_c) to the flowshop M_{h^*}, and ignores the jobs after J_c in the sorted sequence given by step 1 of the algorithm that are assigned to *other* flowshops by the schedule $\mathcal{S}$. Thus, it is obvious that the schedule $\mathcal{S}_c$ also achieves its makespan (whose value $\tau(\mathcal{S}_c)$ is equal to $\tau(\mathcal{S})$) on the flowshop M_{h^*}. □

In particular, in the schedule $\mathcal{S}_c$ constructed by Algorithm 3 for the job set G_c, we also have d as the smallest index in the job sequence $\langle J_1, J_2, \ldots, J_c\rangle$ assigned to the flowshop M_{h^*} such that the T-operations of the jobs J_d, J_{d+1}, …, J_c on M_{h^*} are executed continuously. Finally, J_c now becomes the last job in the sorted job sequence constructed by step 1 of Algorithm 3 for the job set G_c, and hence has the minimum R-time over all jobs in G_c.

By Lemma 3.7, the makespan $\tau(\mathcal{S}_c)$ of the schedule $\mathcal{S}_c$ for the job set G_c is equal to the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$ for the job set G, and $\text{Opt}(G) \geq \text{Opt}(G_c)$. This derives immediately $\tau(\mathcal{S})/\text{Opt}(G) \leq \tau(\mathcal{S}_c)/\text{Opt}(G_c)$. Therefore, any upper bound for the ratio $\tau(\mathcal{S}_c)/\text{Opt}(G_c)$ is also an upper bound for the ratio $\tau(\mathcal{S})/\text{Opt}(G)$. Thus, it suffices to study the approximation ratio of the schedule $\mathcal{S}_c$ for the job set G_c.

Let $J_c = (r_c, t_c)$. The analysis of the approximation ratio for Algorithm 3 is divided into two cases, depending on whether the R-time r_c of job J_c is larger than $\text{Opt}(G_c)/3$. We first consider the case where $r_c > \text{Opt}(G_c)/3$.

Lemma 3.8. *Suppose that $r_c > \text{Opt}(G_c)/3$. Then in the schedule $\mathcal{S}_c$ for the job set G_c, no flowshop has its ρ-value larger than $\text{Opt}(G_c)$.*

Proof. Consider the R-partial job set G_c^r of the job set G_c. We have $\text{Opt}(G_c) \geq \text{Opt}(G_c^r)$. The assumption that r_c, the minimum value over the R-times of all jobs in G_c, is larger than $\text{Opt}(G_c)/3$ implies that every job in G_c has its R-time larger than $\text{Opt}(G_c^r)/3$. According to the definition of R-partial jobs, the R-times of the corresponding jobs in G_c and G_c^r are the same. Thus, the R-time of each job in G_c^r is also strictly larger than $\text{Opt}(G_c^r)/3$.

Let $\mathcal{S}_c^r$ be an optimal schedule for the job set G_c^r. If $\mathcal{S}_c^r$ assigns more than two jobs to a flowshop M_j, then the ρ-value of M_j is larger than $3 \cdot \text{Opt}(G_c^r)/3 = \text{Opt}(G_c^r)$, thus the makespan of $\mathcal{S}_c^r$ would be larger than $\text{Opt}(G_c^r)$, contradicting the assumption that $\mathcal{S}_c^r$ is an optimal schedule for G_c^r. Thus, $\mathcal{S}_c^r$ assigns at most two jobs of G_c^r to each flowshop.

This derives that the total number n of jobs in G_c^r is bounded by $2m$, and that an optimal schedule $\mathcal{S}_c^r$ of G_c^r is to sort the jobs in G_c^r in non-increasing order by their R-times, then, for each $i \leq m$, to place the i-th job in the flowshop M_i, and for each $i > m$, to place the i-th job in the flowshop M_{2m+1-i}. This is exactly what Algorithm 3 does (note that here we have used specifically our rules of breaking ties in Step 3 of the algorithm). Therefore, Algorithm 3 on the job set G_c^r will give an optimal schedule $\mathcal{S}_c^r$ for G_c^r, in which the ρ-value of every flowshop is bounded by $\text{Opt}(G_c^r)$. By Lemma 3.6, for each flowshop M_i, the ρ-value of M_i in the schedule $\mathcal{S}_c$ constructed by Algorithm 3 for the job set G_c is the same as the ρ-value of M_i in the schedule $\mathcal{S}_c^r$ constructed by Algorithm 3 for the job set G_c^r. As a consequence, in the schedule constructed by Algorithm 3 for the job set G_c, no flowshop has its ρ-value larger than $\text{Opt}(G_c^r)$. Now the lemma follows because $\text{Opt}(G_c^r) \leq \text{Opt}(G_c)$. □

Based on Lemma 3.8, the approximation ratio of Algorithm 3 under the condition $r_c > \text{Opt}(G_c)/3$ is given as follows:

Lemma 3.9. *If $r_c > \text{Opt}(G_c)/3$, then the makespan $\tau(\mathcal{S}_c)$ of the schedule $\mathcal{S}_c$ constructed by Algorithm 3 for the job set G_c is bounded by $3 \cdot \text{Opt}(G_c)/2$.*

Proof. Fig. 2 demonstrates the configuration of the flowshop M_{h^*}, which achieves the makespan $\tau(\mathcal{S}_c)$ of the schedule $\mathcal{S}_c$. Since $r_c > \text{Opt}(G_c)/3$, all jobs in G_c have their R-times larger than $\text{Opt}(G_c)/3$. This together with Lemma 3.8 implies that each flowshop is assigned at most two jobs. If there is only one job assigned to M_{h^*} by the schedule $\mathcal{S}_c$, then by Lemma 3.1, the makespan $\tau(\mathcal{S}_c)$, which is the completion time of the flowshop M_{h^*}, is not larger than $\text{Opt}(G_c)$. Thus, assume that there are two jobs J_1 and J_c that are assigned to M_{h^*} by the schedule $\mathcal{S}_c$.

The analysis is divided into two cases based on whether the T-processor of M_{h^*} executes continuously, as given in Fig. 2. When there is an execution gap in the process of the T-operations of J_1 and J_c, as given in the left figure in Fig. 2, we have:

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

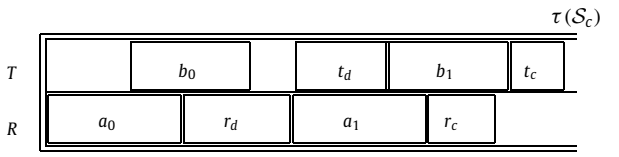

Fig. 3. The configuration of the flowshop M_{h^*} for $P|2\text{FL}_{R\geq T}|C_{\max}$ when $r_c \leq \text{Opt}(G)/3$.

$$\begin{aligned}\tau(\mathcal{S}_c) &= r_1 + r_c + t_c = (r_1 + r_c) + t_c \\ &\leq \text{Opt}(G_c) + \frac{1}{2}\cdot\text{Opt}(G_c) = \frac{3}{2}\cdot\text{Opt}(G_c),\end{aligned} \tag{4}$$

where the inequality $r_1 + r_c \leq \text{Opt}(G_c)$ follows from Lemma 3.8, while the inequality $t_c \leq \text{Opt}(G_c)/2$ follows from Lemma 3.1.

Now consider the case when the T-operations of J_1 and J_c are executed continuously, as shown in the right figure in Fig. 2. We have

$$\begin{aligned}\tau(\mathcal{S}_c) &= r_1 + t_1 + t_c = (r_1 + t_1) + t_c \\ &\leq \text{Opt}(G_c) + \frac{1}{2}\cdot\text{Opt}(G_c) = \frac{3}{2}\cdot\text{Opt}(G_c),\end{aligned} \tag{5}$$

where both inequalities $r_1 + t_1 \leq \text{Opt}(G_c)$ and $t_c \leq \text{Opt}(G_c)/2$ follow from Lemma 3.1.

Now the lemma follows directly by combining these two cases. □

We now consider the other case where the last job J_c on the flowshop M_{h^*} has its R-time r_c no larger than $\text{Opt}(G_c)/3$. In this case, the configuration of the flowshop M_{h^*} under the schedule $\mathcal{S}_c$ constructed by Algorithm 3 is shown in Fig. 3, where we have used the notations similar to those used in section 3.1. Thus, d is the smallest index such that the T-operations of the jobs $J_d, J_{d+1}, \ldots, J_c$ are executed continuously on the flowshop M_{h^*}, $a_0 = \sum_{i=1}^{d-1} r_i$, $b_0 = \sum_{i=1}^{d-1} t_i$, $a_1 = \sum_{i=d+1}^{c-1} r_i$ and $b_1 = \sum_{i=d+1}^{c-1} t_i$. Since Algorithm 3 sorts the jobs in non-increasing order by R-time and J_c is the last job in this sorted sequence, $r_c \leq r_d$. Note that Lemmas 3.1 and 3.2 hold true for arbitrary job sequences, so are still valid here. Lemma 3.3 also holds because M_{h^*} had the minimum ρ-value over all flowshops before the job J_c is assigned.

Lemma 3.10. *If $r_c \leq Opt(G_c)/3$, then the makespan $\tau(\mathcal{S}_c)$ of the schedule $\mathcal{S}_c$ constructed by Algorithm 3 for the job set G_c is bounded by $11\cdot Opt(G_c)/6$.*

Proof. The makespan of the schedule $\mathcal{S}_c$, which is achieved on the flowshop M_{h^*}, is expressed as:

$$\tau(\mathcal{S}_c) = a_0 + r_d + t_d + b_1 + t_c \tag{6}$$

$$\leq a_0 + r_d + t_d + a_1 + r_c = (a_0 + r_d + a_1) + t_d + r_c \tag{7}$$

$$\leq \text{Opt}(G_c) + \frac{1}{2}\cdot\text{Opt}(G_c) + \frac{1}{3}\cdot\text{Opt}(G_c) \tag{8}$$

$$= \frac{11}{6}\cdot\text{Opt}(G_c).$$

The equality (6) derives directly from Fig. 3. By the definition of the index d, there is no execution gap in the block b_1. Thus, $b_1 \leq a_1$, and $t_c \leq r_c$ since all jobs have their T-time bounded by their R-time. This explains the inequality in (7). The inequality $a_0 + r_d + a_1 \leq \text{Opt}(G_c)$ follows from Lemma 3.3, and the inequality $t_d \leq \text{Opt}(G_c)/2$ follows from Lemma 3.1. Combining all these with the condition $r_c \leq \text{Opt}(G_c)/3$ gives the inequality in (8), which proves the lemma. □

Now we are ready to derive the approximation ratio for Algorithm 3.

Theorem 3.11. *Algorithm 3 is a $\frac{11}{6}$-approximation algorithm for the problem $P|2\text{FL}_{R\geq T}|C_{\max}$ of scheduling two-stage jobs on multiple flowshops, where the number m of flowshops is part of the input and jobs are given offline.*

Proof. For an arbitrary given job set G in the model $P|2\text{FL}_{R\geq T}|C_{\max}$, let G_c be the job set as defined above. Let $\mathcal{S}$ and $\mathcal{S}_c$ be the schedules constructed by Algorithm 3 for the job sets G and G_c, respectively. By Lemmas 3.9 and 3.10, the makespan $\tau(\mathcal{S}_c)$ of the schedule $\mathcal{S}_c$ is bounded by $11\cdot\text{Opt}(G_c)/6$. By Lemma 3.7, $\text{Opt}(G) \geq \text{Opt}(G_c)$, and the schedules $\mathcal{S}$ and $\mathcal{S}_c$ have the same makespan. Thus, the makespan $\tau(\mathcal{S})$ of the schedule $\mathcal{S}$ is bounded by $11\cdot\text{Opt}(G)/6$. The theorem follows. □

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Algorithm 4 The online algorithm for $P|2\text{FL}_{R\leq T}|C_{\max}$.

1: When a job $\bar{J}_i$ arrives, assign $\bar{J}_i$ to the flowshop M_q with the minimum ψ-value;
2: $\psi_q = \psi_q + t_i$ and update the ρ and τ values of M_q using Algorithm 1.

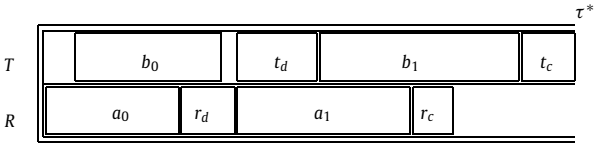

Fig. 4. The configuration of M_{h^*} in the online algorithm for $P|2\text{FL}_{R\leq T}|C_{\max}$.

4. Scheduling on model $P|2\text{FL}_{R\leq T}|C_{\max}$

In this section, we consider the model $P|2\text{FL}_{R\leq T}|C_{\max}$, in which the R-time r_i of each job $J_i = (r_i, t_i)$ is bounded by its T-time t_i. This corresponds to the situation where clusters consist of powerful servers in which disk I/O transfer-rate is much faster compared to that of network transmission.

We first consider the problem in online setting, where jobs come in a sequence $\langle \bar{J}_1, \bar{J}_2, \ldots, \bar{J}_n \rangle$ and algorithms have no information about latter jobs when scheduling a job. For convenience, we define an array set $\psi = \{\psi_1, \ldots, \psi_m\}$, where ψ_q $(1 \leq q \leq m)$ records the sum of the T-times of the jobs currently scheduled on the flowshop M_q. When a new job $\bar{J}_i$ is coming, this online algorithm always chooses the flowshop with the minimum ψ-value over all flowshops for $\bar{J}_i$. The online algorithm is shown as Algorithm 4.

The only difference between Algorithm 2 and Algorithm 4 is that the scheduling strategy of Algorithm 4 is based on the sum of the T-times instead of the sum of the R-times as in Algorithm 2. Therefore, we also use a min-heap to organize the flowshops but in terms of their ψ-values. The min-heap supports finding a flowshop with the minimum ψ-value in time $O(1)$ and re-organizing the flowshops in time $O(\log m)$ after assigning a new job [6]. Thus, the data structure again makes Algorithm 4 "effective" according to [5]. Let $\text{Opt}(G)$ be the makespan of an optimal schedule for a two-stage job sequence G in a given system of m flowshops. Suppose that the schedule $\mathcal{S}$ constructed by Algorithm 4 for the job sequence G achieves its makespan τ^* on the flowshop M_{h^*}, and that the job sequence $\langle J_1, J_2, \ldots, J_c \rangle$ is the schedule on M_{h^*}. Let d be the minimum job index such that M_{h^*} executes the T-operations of the jobs J_d, J_{d+1}, …, J_c continuously. The same notations in section 3.1 are also used here: $a_0 = \sum_{i=1}^{d-1} r_i$, $b_0 = \sum_{i=1}^{d-1} t_i$, $a_1 = \sum_{i=d+1}^{c-1} r_i$ and $b_1 = \sum_{i=d+1}^{c-1} t_i$. The configuration of M_{h^*} under the schedule $\mathcal{S}$ is illustrated in Fig. 4. Note that for convenience, we let b_0 denote the sum of the T-times of the jobs before J_d on M_{h^*}, though the execution of these T-operations may not be continuous. The proofs for the following lemmas are completely similar to that for Lemmas 3.1, 3.2, and 3.3, respectively, thus are omitted.

Lemma 4.1. *For any job $J_i = (r_i, t_i)$ in the job sequence G, the optimal makespan $\text{Opt}(G)$ is at least $r_i + t_i$, and under the model $P|2\text{FL}_{R\leq T}|C_{\max}$, is not smaller than $2r_i$.*

Lemma 4.2. *a_0 is not larger than b_0.*

Lemma 4.3. *$\text{Opt}(G)$ is not smaller than $b_0 + t_d + b_1$.*

The analysis for the online algorithm is shown as Theorem 4.4.

Theorem 4.4. *Algorithm 4 is a $\frac{5}{2}$-competitive algorithm for the problem $P|2\text{FL}_{R\leq T}|C_{\max}$ of scheduling two-stage jobs on multiple flowshops, where the number m of flowshops is part of the input and the jobs are given online.*

Proof. The makespan of the schedule $\mathcal{S}$ constructed by Algorithm 4, which is achieved on the flowshop M_{h^*}, is expressed as (see Fig. 4):

$$\begin{aligned}
\tau^* &= a_0 + r_d + t_d + b_1 + t_c \\
&\leq b_0 + r_d + t_d + b_1 + t_c = (b_0 + t_d + b_1) + r_d + t_c && (9) \\
&< \text{Opt}(G) + \frac{1}{2} \cdot \text{Opt}(G) + \text{Opt}(G) && (10) \\
&= \frac{5}{2} \cdot \text{Opt}(G).
\end{aligned}$$

The inequality in (9) follows from Lemma 4.2. By Lemma 4.3, $b_0 + t_d + b_1 \leq \text{Opt}(G)$. By Lemma 4.1, $r_d < \text{Opt}(G)/2$ and $\text{Opt}(G) \geq r_c + t_c$ for the job J_c in G, so $\text{Opt}(G) \geq t_c$. Combining all these gives the inequality in (10), which proves the lemma. □

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Algorithm 5 The offline algorithm for $P|2\text{FL}_{R\leq T}|C_{\max}$.

1: let $G^d = \{J_1^d, \ldots, J_n^d\}$, where for each i, $J_i^d = (t_i, r_i)$;
2: call Algorithm 3 on G^d, which returns a schedule $\mathcal{S}^d$ for G^d;
3: construct a schedule for G from the schedule $\mathcal{S}^d$ for G^d.

Now we discuss how to deal with the problem $P|2\text{FL}_{R\leq T}|C_{\max}$ in offline setting. Given a job set G of n two-stage jobs in the model $P|2\text{FL}_{R\leq T}|C_{\max}$ and m flowshops, we first construct the dual job set G^d of G. Since the job set G is in the model $P|2\text{FL}_{R\leq T}|C_{\max}$, the dual job set G^d is in the model $P|2\text{FL}_{R\geq T}|C_{\max}$. Thus, by Theorems 3.5 and 3.11, applying the offline Algorithm 3 on the job set G^d constructs in time $O(n\log n)$ a schedule $\mathcal{S}^d$ for G^d with its makespan bounded by $11\cdot \text{Opt}(G^d)/6$. By Theorem 2.2, under the schedule $\mathcal{S}^d$, if we construct for each flowshop M_i the schedule that is the dual of the schedule for M_i under $\mathcal{S}^d$, we will get a schedule for the original job set G, with the completion time of each flowshop unchanged. As a result, this will give a schedule $\mathcal{S}$ for the job set G on m flowshops whose makespan is equal to the makespan of the schedule $\mathcal{S}^d$ for the dual job set G^d. This, when combined with Theorem 2.3 that $\text{Opt}(G^d) = \text{Opt}(G)$, shows that the makespan of the schedule $\mathcal{S}$ constructed by this process for the job set G is bounded by $11\cdot \text{Opt}(G)/6$. This offline algorithm for the model $P|2\text{FL}_{R\leq T}|C_{\max}$ is given as Algorithm 5.

Compared to Algorithm 3, the additional steps in Algorithm 5, which build the dual job set G^d of G and construct the schedule for G from the schedule constructed by Algorithm 3 on G^d, take time $O(n)$. Therefore, the time complexity of Algorithm 5 is still $O(n\log n)$.

Theorem 4.5. *Algorithm 5 is a $\frac{11}{6}$-approximation algorithm for the problem $P|2\text{FL}_{R\leq T}|C_{\max}$ of scheduling two-stage jobs on multiple flowshops, where the number m of flowshops is part of the input and jobs are given offline.*

5. Conclusion

Motivated by the current research in data centers and cloud computing, we studied the problem that schedules two-stage jobs on m multiple two-stage flowshops to minimize the makespan. In particular, we considered the problem in the situation where the number m of flowshops is part of the input, which is a strongly NP-hard problem. To meet the practice demand, this paper studied two restricted versions of the problem based on whether the R-time is smaller than the T-time for each job. The restricted versions of the problem remain strongly NP-hard, and are of practical importance.

Online and offline algorithms for both versions have been provided. For the first case that assumes that the R-time is not smaller than the T-time for each job, an online 2-competitive algorithm has been given, whose competitive ratio almost matches that of the best online algorithm for the classical Makespan problem, which can be regarded as a simpler case of our problem. An offline $\frac{11}{6}$-approximation algorithm has also been developed. For the other case that restricts that the R-time is not larger than the T-time for each job, we gave an online $\frac{5}{2}$-competitive algorithm. Using the concept and theorems of dual jobs and dual schedules, we showed that by applying the offline algorithm in the first case on the dual job set, we can get an offline $\frac{11}{6}$-approximation algorithm for the original job set in the second case.

Further research includes improved approximation algorithms for the problems studied in the current paper. For the online setting, it seems non-trivial to significantly improve the results presented in the current paper, since such improvement would imply improvement on the competitive ratio of the best online algorithms for the classical Makespan problem, which has been of interests in the scheduling literature for many years. On the other hand, for the offline setting, the Makespan problem has polynomial-time approximation schemes, which is significantly better than the approximation ratio 11/6 presented in the current paper, though the $P|2\text{FL}_{R\leq T}|C_{\max}$ problem seems much harder than the Makespan problem.

In our models, a two-stage job $J = (r, t)$ always has two stages even when a stage operation takes time 0. As a result, before the T-operation of J can start, it will have to wait until the R-operation of J "passes through" the R-processor even in case the R-operation takes time 0. A natural extension of this model is to also allow one-stage jobs. Thus, in a job $J = (r, t)$, the values r and t can be either a non-negative integer or a special symbol "$-$", meaning that the job does not have the corresponding operation. In such an extended model, the T-operation of a job $J = (r, t)$ with $r =$ "$-$" can start as soon as the T-processor of the flowshop is available.

For this extended model, our algorithms presented in the current paper are still valid, with the same competitive/approximation ratio. To see this, let G_{ext} be a set of two-stage jobs in the extended model. Replacing each "$-$" in G_{ext} with the value 0, we get a set G_{org} of two-stage jobs in our original model. Let $\mathcal{S}_{org}^{on}$ and $\mathcal{S}_{org}^{off}$ be the schedules constructed by the online and offline algorithms, respectively, for the job set G_{org}, which have the competitive ratio and approximation ratio as given in the current paper. It is easy to see that $\mathcal{S}_{org}^{on}$ and $\mathcal{S}_{org}^{off}$ are also valid for the job set G_{ext} in the extended model, with the same makespan. On the other hand, all our estimations on the optimal makespan $Opt(G)$ in the current paper take no advantage of the original model: they are all based on the sum of T-times or R-times of the jobs, so are still valid lower bounds under the extended model. In conclusion, the competitive/approximation ratio of the algorithms still hold true under the extended model.

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grants 61420106009, 61672536 and 61472449, and by Scientific Research Fund of Hunan Provincial Education Department under grant 16C1660.

References

[1] D. Abts, B. Felderman, A guided tour through data-center networking, ACM Queue 10 (5) (2012) 10–23.
[2] S. Albers, Better bounds for online scheduling, SIAM J. Comput. 29 (2) (1999) 459–473.
[3] L. Barroso, J. Clidaras, U. Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Synthesis Lectures on Computer Architecture, USA:Morgan & Claypool, San Rafael, CA, 2013.
[4] Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, ACM, 1992, pp. 51–58.
[5] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, New York, 2005.
[6] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, third edition, MIT Press, Cambridge, 2009.
[7] J. Dong, J. Hu, M. Kovalyov, G. Lin, T. Luo, W. Tong, X. Wang, Y. Xu, Corrigendum to "An FPTAS for the parallel two-stage flowshop problem", Theoret. Comput. Sci. 687 (2017) 93–94.
[8] J. Dong, W. Tong, T. Luo, X. Wang, J. Hu, Y. Xu, G. Lin, An FPTAS for the parallel two-stage flowshop problem, Theoret. Comput. Sci. 657 (2017) 64–72.
[9] G. Galambos, G. Woeginger, An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling, SIAM J. Comput. 22 (2) (1993) 349–355.
[10] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York, 1979.
[11] R. Graham, Bounds for certain multiprocessing anomalies, Bell Syst. Tech. J. 45 (9) (1966) 1563–1581.
[12] R. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (2) (1969) 416–429.
[13] R. Graham, E. Lawler, J. Lenstra, R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1) (1979) 287–326.
[14] D. He, A. Kusiak, A. Artiba, A scheduling problem in glass manufacturing, IIE Trans. 28 (2) (1996) 129–139.
[15] S. Johnson, Optimal two- and three-stage production schedules with setup times included, Nav. Res. Logist. Q. 1 (1) (1954) 61–68.
[16] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer Science, New York, 2016.
[17] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristics, European J. Oper. Res. 165 (2) (2005) 479–494.
[18] W. Tong, E. Miyano, R. Goehel, G. Lin, A PTAS for the multiple parallel identical multi-stage flow-shops to minimize the makespan, in: Proc. 10th International Workshop on Frontiers in Algorithmics, in: Lecture Notes in Computer Science, vol. 10336, Springer, 2016, pp. 227–237.
[19] G. Vairaktarakis, M. Elhafsi, The use of flowlines to simplify routing complexity in two-stage flowshops, IIE Trans. 32 (8) (2000) 687–699.
[20] D. Williamson, D. Shmoys, The Design of Approximation Algorithms, Cambridge University Press, New York, 2011.
[21] G. Wu, J. Chen, J. Wang, On Scheduling Two-Stage Jobs on Multiple Two-Stage Flowshops, Tech. Rep., School of Information Science and Engineering, Central South University, 2016.
[22] G. Wu, J. Chen, J. Wang, On scheduling multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.017.
[23] X. Zhang, S. van de Velde, Approximation algorithms for the parallel flow shop problem, European J. Oper. Res. 216 (3) (2012) 544–552.

Please cite this article in press as: G. Wu et al., On scheduling inclined jobs on multiple two-stage flowshops, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.005