Doctopic: Algorithms, automata, complexity and games ARTICLE IN PRESS TCS:11317

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

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

An approximation scheme for minimizing the makespan of the parallel identical multi-stage flow-shops [☆]

Weitian Tong [a,b], Eiji Miyano [c], Randy Goebel [b], Guohui Lin [b,*]

[a] Department of Computer Sciences, Georgia Southern University, Statesboro, GA 30460, USA
[b] Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada
[c] Department of Systems Design and Informatics, Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan

A R T I C L E   I N F O

Article history:
Received 5 August 2016
Received in revised form 7 July 2017
Accepted 21 September 2017
Available online xxxx

Keywords:
Multiprocessor scheduling
Flow-shop scheduling
Makespan
Linear program
Polynomial-time approximation scheme

A B S T R A C T

In the *parallel k-stage flow-shops* problem, we are given m identical k-stage flow-shops and a set of jobs. Each job can be processed by any one of the flow-shops but switching between flow-shops is not allowed. The objective is to minimize the makespan, which is the finishing time of the last job. This problem generalizes the classical parallel identical machine scheduling (where $k = 1$) and the classical flow-shop scheduling (where $m = 1$) problems, and thus it is NP-hard. We present a polynomial-time approximation scheme (PTAS) for the problem, when m and k are fixed constants. The key technique is to partition the jobs into *big* jobs and *small* jobs, enumerate over all feasible schedules for the big jobs, and handle the small jobs by solving a linear program and employing a "sliding" method. Such a technique has been used in the design of PTAS for several flow-shop scheduling variants. Our main contributions are the non-trivial application of this technique and a valid answer to the open question in the literature.

1. Introduction

In the *parallel k-stage flow-shop* problem, we are given m parallel identical k-stage flow-shops $F_1, F_2, \ldots, F_m$ and a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. These k-stage flow-shops are the *classic* flow-shops, each contains exactly one machine at every stage, *i.e.*, k sequential machines. Every job has k operations, and it can be assigned to exactly one of the m flow-shops for processing; once it is assigned to the flow-shop, its k operations are then respectively processed on the k sequential machines in the flow-shop. The goal is to minimize the makespan, which is the completion time of the last job. We denote the problem for simplicity as (m, k)-PFS. Let $M_{\ell,1}, M_{\ell,2}, \ldots, M_{\ell,k}$ denote the k sequential machines in the flow-shop F_ℓ, for every ℓ. The job J_i is represented as a k-tuple $(p_{i,1}, p_{i,2}, \ldots, p_{i,k})$, where $p_{i,j}$ is the processing time for the j-th operation, that is, J_i needs to be processed non-preemptively on the j-th machine in the flow-shop to which the job is assigned. For all i and j, the processing time $p_{i,j}$ is a non-negative real number.

It is clear to see that, when $m = 1$, the (m, k)-PFS problem is the classic *flow-shop scheduling* [5] (a k-stage flow-shop); when $k = 1$, the (m, k)-PFS problem is the classic *multiprocessor scheduling* [5] (m parallel identical machines). When the two-stage flow-shops are involved, *i.e.*, $k = 2$, the $(m, 2)$-PFS problem has been previously studied in [13,25,28,4]. Here

☆ An extended abstract appears in the Proceedings of FAW 2016.
* Corresponding author.
E-mail addresses: wtong@georgiasouthern.edu (W. Tong), miyano@ces.kyutech.ac.jp (E. Miyano), rgoebel@ualberta.ca (R. Goebel), guohui@ualberta.ca (G. Lin).

https://doi.org/10.1016/j.tcs.2017.09.018
0304-3975/

we first review the complexity and the approximation algorithms for the flow-shop scheduling and the multiprocessor scheduling problems.

For the k-stage flow-shop problem, it is known that when $k = 2$ or 3, there exists an optimal schedule that is a *permutation schedule*, in which the jobs are processed on all the k machines in the same order; but when $k \geq 4$, it is shown [3] that there may exist no optimal schedule that is a permutation schedule. Johnson [18] presented an $O(n \log n)$-time algorithm for the two-stage flow-shop problem, where n is the number of jobs; the k-stage flow-shop problem becomes *strongly* NP-hard when $k \geq 3$ [6]. After several efforts [18,6,7,2], Hall [12] designed a polynomial-time approximation scheme (PTAS) for the k-stage flow-shop problem, for any fixed constant $k \geq 3$. Due to the strong NP-hardness, such a PTAS is the best possible unless P = NP. When k is a part of the input (*i.e.*, an arbitrary integer), Williamson et al. [27] showed that the flow-shop scheduling cannot be approximated within 1.25; nevertheless, it remains unknown whether this case is APX-complete, that is, whether the problem admits a constant ratio approximation algorithm.

Note that the m-parallel identical machine scheduling problem is NP-hard when $m \geq 2$ [5]. When m is a fixed integer, the problem admits a pseudo-polynomial time exact algorithm [5], and Sahni [23] showed that this exact algorithm can be used to construct a *fully* PTAS (FPTAS); when m is a part of the input, the problem becomes strongly NP-hard, but still admits a PTAS by Hochbaum and Shmoys [14].[1] The *list-scheduling* algorithm by Graham [8] is a $(2 - 1/m)$-approximation, for arbitrary m.

The APX-hardness of the classic k-stage flow-shop problem when k is a part of the input implies the APX-hardness of the (m, k)-PFS problem when k is a part of the input. When k is a fixed integer, the (m, k)-PFS problem could admit a PTAS; however, since the classic k-stage flow-shop problem is strongly NP-hard for a fixed $k \geq 3$, the (m, k)-PFS problem would not admit an FPTAS unless P = NP. In this paper, we present a PTAS for the (m, k)-PFS problem when both k and m are fixed integers, which is the best possible approximability result. On the other hand, the (in-)approximability of the (m, k)-PFS problem when m is a part of the input while k is a fixed integer is left open.

Besides the (m, k)-PFS problem, another generalization of the flow-shop scheduling and the multiprocessor scheduling is the so-called *hybrid k-stage flow-shop* problem [20,22]. A hybrid k-stage flow-shop is a *flexible* flow-shop, that contains $m_j \geq 1$ parallel identical machines in the j-th stage, for $j = 1, 2, \ldots, k$. The problem is abbreviated as $(m_1, m_2, \ldots, m_k)$-HFS. A job J_i is again represented as a k-tuple $(p_{i,1}, p_{i,2}, \ldots, p_{i,k})$, where $p_{i,j}$ is the processing time for the j-th operation, which can be processed non-preemptively on any one of the m_j machines in the j-th stage. The objective of the $(m_1, m_2, \ldots, m_k)$-HFS problem is also to minimize the makespan. One clearly sees that when $m_1 = m_2 = \ldots = m_k = 1$, the problem reduces to the classic k-stage flow-shop problem; when $k = 1$, the problem reduces to the classic m-parallel identical machine scheduling problem.

As a toy example, suppose there is a set of three jobs, $\mathcal{J} = \{J_1 = (p_{1,1}, p_{1,2}, p_{1,3}), J_2 = (p_{2,1}, p_{2,2}, p_{2,3}), J_3 = (p_{3,1}, p_{3,2}, p_{3,3})\}$, that need to be processed. When we are provided with a $(2, 3)$-PFS (that is, two parallel identical 3-stage flow-shops), we may assign J_1 to the second flow-shop; then J_1 will be processed on the first machine of the second flow-shop for $p_{1,1}$ units of time, then on the second machine of the second flow-shop for $p_{1,2}$ units of time, and lastly on the third machine of the second flow-shop for $p_{1,3}$ units of time. On the other hand, if we are provided with a $(2, 1, 3)$-HFS, then we may process J_1 on any one of the two first-stage machines for $p_{1,1}$ units of time, then on the (only) second-stage machine for $p_{1,2}$ units of time, and lastly on any one of the three third-stage machine for $p_{1,3}$ units of time.

The literature on the hybrid k-stage flow-shop problem $(m_1, m_2, \ldots, m_k)$-HFS is also rich [20,22], especially on the hybrid two-stage flow-shop problem (m_1, m_2)-HFS. First, $(1, 1)$-HFS is the classic two-stage flow-shop problem which can be optimally solved in $O(n \log n)$ time [18], where n is the number of jobs. When $\max\{m_1, m_2\} \geq 2$, Hoogeveen et al. [15] showed that the (m_1, m_2)-HFS problem is strongly NP-hard. The special cases $(m_1, 1)$-HFS and $(1, m_2)$-HFS have attracted many researchers' attention [9,11,1,10]; the interested reader might refer to [26] for a survey on the hybrid two-stage flow-shop problem with a single machine in one stage.

For the general hybrid k-stage flow-shop problem $(m_1, m_2, \ldots, m_k)$-HFS, when all the m_1, m_2, ..., m_k are fixed integers, Hall [12] claimed that the PTAS for the classic k-stage flow-shop problem can be extended to a PTAS for the $(m_1, m_2, \ldots, m_k)$-HFS problem. Later, Schuurman and Woeginger [24] presented a PTAS for the hybrid two-stage flow-shop problem (m_1, m_2)-HFS, even when the numbers of machines m_1 and m_2 in the two stages are a part of the input. Jansen and Sviridenko [17] generalized this result to the hybrid k-stage flow-shop problem $(m_1, m_2, \ldots, m_k)$-HFS, where k is a fixed integer while $m_1, m_2, \ldots, m_k$ can be a part of the input. Due to the inapproximability of the classic k-stage flow-shop problem, when k is arbitrary, the $(m_1, m_2, \ldots, m_k)$-HFS problem cannot be approximated within 1.25 unless P = NP [27]. Table 1 summarizes the results we reviewed thus far.[2] In addition, there are plenty of heuristic algorithms in the literature for the general hybrid k-stage flow-shop problem, and the interested readers can refer to the survey by Ruiz et al. [22].

Compared to the rich literature on the hybrid k-stage flow-shop problem, the parallel k-stage flow-shop problem is much less studied. In fact, the general (m, k)-PFS problem is almost untouched, except only the two-stage flow-shops are involved [13,25,28,4]. He et al. [13] first studied the m parallel identical two-stage flow-shop problem $(m, 2)$-PFS, motivated

[1] We note that there are sequences of work in developing faster PTASes, which are not the intended subject in this paper. The interested readers might refer to [16] for major references.

[2] We do not list the detailed running time of these algorithms. Again, we note that there are sequences of work in developing faster PTASes, which are not the intended subject in this paper. The interested readers might refer to [16] for major references.

Please cite this article in press as: W. Tong et al., An approximation scheme for minimizing the makespan of the parallel identical multi-stage flow-shops, Theoret. Comput. Sci. (2017), https://doi.org/10.1016/j.tcs.2017.09.018