Information Processing Letters 120 (2017) 6–10

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Notes on a hierarchical scheduling problem on identical machines ☆

Cheng He *, Hao Lin

School of Science, Henan University of Technology, Zhengzhou, Henan 450052, China

ARTICLE INFO

Article history:
Received 13 September 2015
Received in revised form 23 February 2016
Accepted 8 December 2016
Available online 21 December 2016
Communicated by Nathan Fisher

Keywords:
Hierarchical scheduling
Identical machines
Total flowtime
Worst-case ratio
Approximation algorithms

ABSTRACT

For the hierarchical scheduling problem on identical machines to minimize the maximum T-time of all machines under the condition that the total completion time of all jobs is minimum, where the T-time of a machine is defined as the total completion time of jobs scheduled on the machine, it is NP-hard if the number of the machines is fixed, and strongly NP-hard otherwise. When the number of the machines is fixed, a forward dynamic programming algorithm and a fully polynomial-time approximation scheme (FPTAS) have been presented in a literature. In the literature, it is showed that the worst-case ratio of the classical algorithm SPT is at most $\frac{11}{6}$ and at least $\frac{5}{3}$. In this paper, we give an improved worst-case ratio, which is at most $\frac{9}{5}$ and at least $\frac{7}{4}$, of the algorithm. Another algorithm, whose worst-case ratio is at most $\frac{7}{6}$ and at least $\frac{35}{33}$, is provided for the two-machine case. On the other hand, we present a backward dynamic programming algorithm and an FPTAS with the better time complexities.

1. Introduction

There are n jobs $J_1, J_2, \ldots, J_n$, with processing times $p_1, p_2, \ldots, p_n$, to be processed on m identical machines $M_1, M_2, \ldots, M_m$ without preemption. A *feasible schedule* is a schedule that non-preemptively process the jobs on the machines. Let $\sigma = (\pi_1, \pi_2, \ldots, \pi_m)$ be a schedule of the problem, π_i is the sequence of jobs in machine M_i ($1 \le i \le m$). Denote by $C_j(\pi_i)$ the completion time of job J_j on machine M_i. Then the flowtime of machine M_i is $\sum_{j \in \pi_i} C_j(\pi_i)$. From this, two objective functions considered in this paper are the total flowtime

$$\sum C_j(\sigma) := \sum_{1 \le i \le m} \sum_{j \in \pi_i} C_j(\pi_i)$$

and the maximum flowtime

$$\left(\sum C_j(\sigma)\right)_{\max} := \max_{1 \le i \le m} \sum_{j \in \pi_i} C_j(\pi_i).$$

The makespan of σ is $C_{\max}(\sigma) = \max_{1 \le i \le m,\ j \in \pi_i} C_j(\pi_i)$.

In the paper, we focus on the hierarchical scheduling problem on m identical machines to minimize the maximum flowtime under the condition that the total flowtime is minimum, denoted by $Pm||Lex(\sum C_j, (\sum C_j)_{\max})$ (called as problem P for short) following the three-field notation of [5]. When m is a part of input, the problem is denoted by $P||Lex(\sum C_j, (\sum C_j)_{\max})$ (called as problem P′ for short). [1] showed that the problem $Pm||(\sum C_j)_{\max}$ is NP-hard and the worst-case ratio of algorithm SPT for the problem is at most $3 - \frac{3}{m} + \frac{1}{m^2}$, and so at most 3 for $P||(\sum C_j)_{\max}$. [6] proved that $P||(\sum C_j)_{\max}$ is strongly NP-hard and the worst-case ratio of algorithm SPT for the problem is at most 2.608. [7] presented the following results for problem P and problem P′.

☆ This work was supported by NSFC (11201121, 11571323) and NSFST-DOHN (162300410221).

* Corresponding author.
E-mail address: hech202@163.com (C. He).

http://dx.doi.org/10.1016/j.ipl.2016.12.001
0020-0190/

(1) Problem P is NP-hard and Problem P′ is strongly NP-hard.

(2) The worst-case ratio of algorithm SPT for problem P′ is at most $\frac{11}{6}$ and at least $\frac{5}{3}$.

(3) The worst-case ratio of algorithm RSPT for problem P′ is at most $\frac{3}{2}$ and at least $\frac{11}{9}$.

(4) An $O(m!\cdot n^{m+1}\cdot P_s^{2m})$-time forward dynamic programming algorithm and an FPTAS with $O(m!\cdot n^{m+1}\cdot (\frac{(mn+\epsilon)n}{\epsilon})^{2m})$ time for problem P, where $P_s=\sum_{j=1}^{n} p_j$.

In the present paper, we improved the worst-case ratio of algorithm SPT for problem P′ such that its upper bound and lower bound are $\frac{9}{5}$ and $\frac{7}{4}$, respectively. For $m=2$, we present a better algorithm called Algorithm DLPT and deduce that its worst-case ratio is at most $\frac{7}{6}$ and at least $\frac{35}{33}$. Moreover, we present an $O(m!\cdot n^{m+1}\cdot P_s^{m})$-time backward dynamic programming algorithm and an FPTAS with $O(\frac{m!\cdot n^{2m+1}}{\epsilon^m})$ time for problem P.

The paper is organized as follows. An improved worst-case ratio of Algorithm SPT is discussed in Section 2. In Section 3, we present a backward dynamic programming algorithm and an FPTAS for problem P. In Section 4, another algorithm, called Algorithm DLPT, is provided for $m=2$. We deduce that the worst-case ratio of Algorithm DLPT is at most $\frac{7}{6}$ and at least $\frac{35}{33}$.

2. Algorithm SPT

Recall that we have n jobs and m machines. We may assume that $n=km$ for some positive integer k (otherwise we may add some dummy jobs with processing time 0 and the dummy jobs are scheduled first on the machines without affecting the two objectives in any schedule). Without loss of generality, we assume that $p_1\ge p_2\ge\cdots\ge p_n$. We partition job set $\mathcal{J}=\{J_1,J_2,\ldots,J_n\}$ into k ranks, where $\mathcal{R}_j=\{J_{(j-1)m+1},J_{(j-1)m+2},\ldots,J_{(j-1)m+m}\}$ is the j-th rank, $j=1,\ldots,k$.

Let π be the schedule obtained from schedule σ by interchanging the positions of two jobs with the same processing times. Then π is essentially the same as σ. So we regard the schedules up to the permutations among the jobs of the same processing times as the same schedules throughout the paper. Suppose that the optimal value of $P||\sum C_j$ is T^*. Then $P||Lex(\sum C_j,(\sum C_j)_{\max})\Leftrightarrow P|\sum C_j\le T^*|(\sum C_j)_{\max}$.

Definition 2.1. A schedule σ is called Quasi-SPT (Q-SPT for short) if σ satisfies the following three conditions.

- Each machine receives exactly one job from $\mathcal{R}_j$, $j=1,\ldots,k$.
- Jobs on each machine are scheduled in the non-decreasing order of processing time.
- There is no idle time.

Lemma 2.2. *([2]) A schedule σ is Q-SPT if and only if σ is an optimal schedule of problem $P||\sum C_j$.*

Lemma 2.2 implies that solving the problem $P|\sum C_j\le T^*|(\sum C_j)_{\max}$ equivalents to finding a Q-SPT optimal schedule of $P||(\sum C_j)_{\max}$. Hence we confine our attention on Q-SPT schedules in the following.

Let $\mathcal{J}^{(i)}$ be the current job set of jobs assigned to M_i and TM_i be the sum of processing times of jobs assigned to M_i at present, i.e., $TM_i=\sum_{j\in\mathcal{J}^{(i)}} p_j$.

Algorithm SPT.

Step 0: Let $TM_i:=0$, $\mathcal{J}^{(i)}:=\emptyset$, $i=1,\ldots,m$ and $j:=n$.
Step 1: Let $TM_{i_0}=\min_{1\le i\le m}\{TM_i\}$ (if a tie, then the minimum i_0 first and M_{i_0} is different from the last machine that is chosen to schedule job). Let $\mathcal{J}^{(i_0)}:=\mathcal{J}^{(i_0)}\bigcup\{J_j\}$ and $TM_{i_0}:=TM_{i_0}+p_j$ and schedule job J_j at the end of current schedule on M_{i_0}.
Step 2: If $j>1$, then let $j:=j-1$ and go back to Step 1. Otherwise stop.

Lemma 2.3. *The schedule derived by Algorithm SPT is a Q-SPT schedule.*

Proof. Obviously, the m jobs $J_n,J_{n-1},\ldots,J_{n-m+1}$, i.e., $J_{km},J_{km-1},\ldots,J_{(k-1)m+1}$ (for $n=km$) in $\mathcal{R}_k$ are scheduled first on $M_1,M_2,\ldots,M_m$, respectively, by Algorithm SPT. Then by Algorithm SPT, we have

Claim 1. *Job J_{lm-i} in $\mathcal{R}_l$ is scheduled on M_{i+1} for $1\le l\le k$ and $0\le i\le m-1$.*

Proof of Claim 1. Algorithm SPT shows that the jobs in $\mathcal{R}_k,\mathcal{R}_{k-1},\ldots,\mathcal{R}_1$ are scheduled one by one. We prove Claim 1 by induction on the number l of ranks in $\mathcal{R}_l$. The basic case, $l=k$, is obvious. Assuming that Claim 1 holds for $2\le l\le k-1$, we will show that it also holds for $l=1$.

According to the assumption that Claim 1 holds for $2\le l\le k-1$, we have $TM_1\le TM_2\le\ldots\le TM_m$, just before the jobs in $\mathcal{R}_1$ are scheduled, by $p_1\ge p_2\ge\cdots\ge p_n$. From Algorithm SPT, we have jobs $J_m,J_{m-1},\ldots,J_1$ in $\mathcal{R}_1$ are scheduled one by one. So job J_m in $\mathcal{R}_1$ is scheduled on M_1 by Algorithm SPT. Further, at present $TM_1+p_m=p_{km}+p_{(k-1)m}+\ldots+p_m\ge p_{km}+p_{(k-1)m+1}+\ldots+p_{m+1}=p_{km}+TM_m\ge TM_m\ge TM_{m-1}\ge\ldots\ge TM_2$. Hence, next job J_{m-1} in $\mathcal{R}_1$ is scheduled on M_2 by Algorithm SPT. Similarly, we may prove that $J_{m-2},J_{m-3},\ldots,J_1$ in $\mathcal{R}_1$ are scheduled on $M_3,M_4,\ldots,M_m$, respectively. Therefore Claim 1 also holds for $l=1$.

By Claim 1 and Algorithm SPT, the schedule derived by Algorithm SPT is a Q-SPT schedule. □

So the schedule derived by Algorithm SPT is a feasible schedule for problem $P|\sum C_j\le T^*|(\sum C_j)_{\max}$. By a more elaborate analysis on the upper bound of the worst-case ratio of Algorithm SPT of [7], we receive better upper bound and lower bound.

Theorem 2.4. *The worst-case ratio of Algorithm SPT for the problem $P|\sum C_j\le T^*|(\sum C_j)_{\max}$ is at most $\frac{9}{5}$ and at least $\frac{7}{4}$.*

Proof. Without loss of generality, we may suppose that $k\ge 4$ by adding some dummy jobs with processing time