Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Scheduling to minimize the maximum total completion time per machine[☆]

Long Wan[a], Zhihao Ding[b], Yunpeng Li[b], Qianqian Chen[b], Zhiyi Tan[c,*]

[a] *School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330013, PR China*
[b] *Department of Mathematics, Zhejiang University, Hangzhou 310027, PR China*
[c] *Department of Mathematics, State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, PR China*

ARTICLE INFO

Article history:
Received 17 July 2013
Accepted 28 September 2014
Available online 7 October 2014

Keywords:
Scheduling
Parallel machine
Worst-case ratio

ABSTRACT

In this paper, we study the problem of minimizing the maximum total completion time per machine on m parallel and identical machines. We prove that the problem is strongly *NP*-hard if m is a part of the input. When m is a given number, a pseudo-polynomial time dynamic programming is proposed. We also show that the worst-case ratio of *SPT* is at most 2.608 and at least 2.5366 when m is sufficiently large. We further present another algorithm which has a worst-case ratio of 2.

1. Introduction

Minimizing the total (weighted) completion time is one of the most commonly studied criteria in scheduling theory (Smith, 1956; Kawaguchi, & Kyan, 1986). It is well known that the problem of minimizing the total completion time on parallel and identical machines can be solved optimally by the *Shortest Processing Time first* (*SPT* for short) algorithm (Conway, Maxwell, & Miller, 1967). Here *SPT* first sorts all jobs in the order of nondecreasing processing times, then assigns the first unprocessed job in the sequence to the machine which can complete it as early as possible.

However, in the schedule produced by *SPT*, the total completion time of the jobs assigned to each machine may vary quite widely. To keep the total completion time per machine as equal as possible, it is convenient to consider the objective of minimizing the maximum total completion time per machine. Suppose a manager of a company wants to assign some late orders to several parallel subsidiaries. Since the orders are delayed, a penalty should be paid for each order, which is proportional to its completion time. The manager wants to be fair such that the total penalty that each subsidiary needs to be paid is as equal as possible. The situation can be viewed as a scheduling problem, in which orders are modeled as jobs and subsidiaries are modeled as machines. Taking the above objective into consideration, the manager may be provided with constructive assistance and receive much less complaints from subsidiaries.

The problem can be stated formally as follows. We are given a sequence of n independent jobs $J_1, J_2, \dots, J_n$, which need to be non-preemptively processed on m parallel and identical machines $M_1, M_2, \dots, M_m$. The size (processing time) of J_j is $p_j, j = 1, \dots, n$. Without loss of generality, we assume $p_1 \le p_2 \le \cdots \le p_n$. Given a schedule σ^A, let C_j^A be the completion time of J_j in σ^A, $j = 1, \dots, n$. The total completion time of the jobs scheduled on M_i is denoted TC_i^A, $i = 1, \dots, m$. Then $TC^A = \max_{i=1,\dots,m} TC_i^A$ is the objective value of σ^A. By extending the three-field notation, the problem can be denoted as $Pm||(\sum C_j)_{\max}$, while the corresponding problem with the objective of minimizing the total completion time is denoted by $Pm||\sum C_j$ as usual. The problem of finding such a schedule among optimal schedules of $Pm||\sum C_j$ that the maximum total completion time per machine is minimized is denoted $Pm||Lex(\sum C_j, (\sum C_j)_{\max})$. We will use σ^* and σ^{**} to denote the optimal schedules of $Pm||(\sum C_j)_{\max}$ and $Pm||Lex(\sum C_j, (\sum C_j)_{\max})$, respectively.

The problem $Pm||(\sum C_j)_{\max}$ was first proposed by Angel, Bampis, and Pascual (2008). They proved that to find schedule σ^* or σ^{**} is NP-hard. They also proved that the worst-case ratio of *SPT* for $Pm||(\sum C_j)_{\max}$ lies in the interval $[2 - \frac{2}{m^2+m}, 3 - \frac{3}{m} + \frac{1}{m^2}]$. In this paper, we show that the problem is strongly *NP*-hard if m is a part of the input. When m is a given number, a pseudo-polynomial time dynamic programming algorithm is proposed, which will directly lead to a *Fully Polynomial Time Approximation Scheme* (FPTAS). We also show that the worst-case ratio of *SPT* is at most 2.608 and at least 2.5366 when m is sufficiently large. The lower and upper bounds of the worst-case ratio of *SPT* when m is a fixed number are also improved. We further present another algorithm *Reverse SPT* (*RSPT* for

☆ This work was supported by the National Natural Science Foundation of China (10971191, 11271324) and Zhejiang Provincial Natural Science Foundation of China (LR12A01001).

* Corresponding author. Tel.: +86 0571 87953867.
E-mail address: tanzy@zju.edu.cn (Z. Tan).

http://dx.doi.org/10.1016/j.ejor.2014.09.063
0377-2217/© 2014 Elsevier B.V. All rights reserved.

short) which is also optimal for $Pm||\sum C_j$ and has a worst-case ratio of 2 for $Pm||(\sum C_j)_{\max}$.

Comparing with the FPTAS, the advantage of *SPT* and *RSPT* is that both are optimal algorithms for $Pm||\sum C_j$. Thus corresponding results have some similarity with *hierarchical optimization*, an approach widely used for multicriteria scheduling (Hoogeveen, 2005; T'kindt, & Billaut, 2006; Perez-Gonzalez, & Framinan, 2014). Inspired by our work, Wan, Ma, and Yuan (2014) considered the problem of $Pm||Lex(\sum C_j, (\sum C_j)_{\max})$, and proved that the worst-case ratios of *SPT* and *RSPT* are at most $\frac{11}{6}$ and $\frac{3}{2}$, respectively. The difference on the worst-case ratios of the same algorithm indicates that σ^* and σ^{**} usually are not identical.

There are fruitful results in the field of multicriteria scheduling problems on parallel machines. Most researches are concerned with two objectives of minimizing the makespan and minimizing the total completion time (Leung, & Young, 1989; Lin, Fowler, & Pfund, 2013; Lee, Leung, Jia, Li, & Pinedo, 2014), which are regarded to be egalitarian and utilitarian objectives, respectively (Myerson, 1981). In fact, the objective of minimizing the maximum total completion time per machine shares the above two characteristic features, which makes our results new and interesting. Another line of research about multicriteria scheduling is the so-called *multi-agent scheduling* (Baker & Smith, 2003; Agnetis, Mirchandani, Pacciarelli, & Pacifici, 2004; Leung, Pinedo, & Wan, 2010; Mor, & Mosheiov, 2011; Nong, Cheng, & Ng, 2011). It arises in situations where multiple agents (customers), each having a different objective, compete for a common processing resource. However, for our problem, jobs belong to the same agent and contribute to the single objective.

The rest of the paper is organized as follows. In Section 2, we present complexity result and dynamic programming. The analyses of *SPT* and *RSPT* are given in Sections 3 and 4, respectively. Finally some conclusions are made in Section 5.

2. Complexity, dynamic programming and FPTAS

In Angel et al. (2008), it is proved that finding a schedule among all schedules or among optimal schedules of $Pm||\sum C_j$ such that the maximum total completion time per machine is minimized is *NP*-hard. However, if m is a part of the input, to find a schedule minimizing the maximum total completion time per machine is strongly *NP*-hard. The following helpful lemma can be proved by a job interchange argument.

Lemma 2.1. *There always exists an optimal schedule of $Pm||(\sum C_j)_{max}$ in which all machines process the jobs in non-decreasing order of the size without idle time.*

Theorem 2.1. *$P||(\sum C_j)_{max}$ is strongly NP-hard.*

Proof. The theorem will be proved by reduction from the numerical three-dimensional matching problem, which is known to be strongly NP-complete (Garey, and Johnson (1978)).

Numerical three-dimensional matching problem: Given three multisets of integers $U = \{u_1, \dots, u_m\}$, $V = \{v_1, \dots, v_m\}$, $W = \{w_1, \dots, w_m\}$ and an integer B such that $\sum_{i=1}^{m}(u_i + v_i + w_i) = mB$, do there exist two permutations ϕ and φ of $\{1, \dots, m\}$ such that $u_i + v_{\phi(i)} + w_{\varphi(i)} = B$ for $i = 1, \dots, m$?

Let I_M be an instance of numerical three-dimensional matching. Construct an instance with m machines and $3m$ jobs. Each job belongs to one of three types, U-job, V-job, or W-job. Corresponding to each element u_i of U, there is a U-job J_{U_i} with size $\frac{u_i}{3}$. Corresponding to each element v_i of V, there is a V-job J_{V_i} with size $\frac{v_i}{2} + B$. Corresponding to each element w_i of W, there is a W-job J_{W_i} with size $w_i + 2B$. We will show that I_M has "yes" answer if and only if there is a feasible schedule such that the maximum total completion time per machine is no more than $5B$.

If I_M has "yes" answer, then there exist two permutations ϕ and φ of $\{1, \dots, m\}$ such that $u_i + v_{\phi(i)} + w_{\varphi(i)} = B$ for $i = 1, \dots, m$. Let $J_{U_i}, J_{V_{\phi(i)}}, J_{W_{\varphi(i)}}$ be the three jobs successively scheduled on M_i, $i = 1, \dots, m$. The total completion time of M_i, $i = 1, \dots, m$, is

$$3\frac{u_i}{3} + 2\left(\frac{v_{\phi(i)}}{2} + B\right) + (w_{\varphi(i)} + 2B) = 5B.$$

On the other hand, suppose there exist feasible schedules such that the maximum total completion time per machine is no more than $5B$. By Lemma 2.1, we can focus on the one, denoted by σ^Y, that jobs are sequenced in non-decreasing order of the sizes on each machine. Clearly, $TC_i^Y \le 5B$, $i = 1, \dots, m$. We show the structure of σ^Y step by step. Firstly, there is exactly one W-job on each machine. Otherwise, there exists machine M_i which processes at least two W-jobs. Thus $TC_i^Y \ge 2 \cdot 2B + 2B = 6B > 5B$, a contradiction. Let the W-job scheduled on M_i be $J_{W_{\varphi(i)}}$, $i = 1, \dots, m$, thus φ is a permutation of $\{1, \dots, m\}$. Secondly, there is at most one V-job on each machine. Otherwise, there exists machine M_i which processes at least two V-jobs. Thus $TC_i^Y \ge 3 \cdot B + 2 \cdot B + 2B = 7B > 5B$, also a contradiction. Let the V-job scheduled on M_i be $J_{V_{\phi(i)}}$, $i = 1, \dots, m$, thus ϕ is also a permutation of $\{1, \dots, m\}$. Finally, let $\mathcal{U}^i$ be the set of U-jobs which are scheduled on M_i, $i = 1, \dots, m$. We claim that $|\mathcal{U}^i| = 1$ for any i. In fact, if $\mathcal{U}^i = \{J_{U_i}\}$, then

$$\begin{aligned} TC_i^Y &= 3\frac{u_i}{3} + 2\left(\frac{v_{\phi(i)}}{2} + B\right) + (w_{\varphi(i)} + 2B) \\ &= u_i + v_{\phi(i)} + w_{\varphi(i)} + 4B. \end{aligned}$$

If $|\mathcal{U}^i| \ge 2$, then

$$\begin{aligned} TC_i^Y &> 3\sum_{J_{U_i} \in \mathcal{U}^i} \frac{u_i}{3} + 2\left(\frac{v_{\phi(i)}}{2} + B\right) + (w_{\varphi(i)} + 2B) \\ &= \sum_{J_{U_i} \in \mathcal{U}^i} u_i + v_{\phi(i)} + w_{\varphi(i)} + 4B. \end{aligned}$$

Hence, if there exists i, such that $|\mathcal{U}^i| \ge 2$, then

$$\begin{aligned} \sum_{i=1}^{m} TC_i^Y &> \sum_{i=1}^{m}\left(\sum_{J_{U_i} \in \mathcal{U}^i} u_i + v_{\phi(i)} + w_{\varphi(i)} + 4B\right) \\ &= 4mB + \sum_{i=1}^{m}(u_i + v_i + w_i) = 5mB \ge mTC^Y, \end{aligned}$$

which is a contradiction. Therefore, $|\mathcal{U}^i| = 1$ for any i and we assume that $\mathcal{U}^i = \{J_{U_i}\}$. Moreover, since $\sum_{i=1}^{m} TC_i^Y = 5mB \ge mTC^Y = m\max_{i=1,\dots,m} TC_i^Y$, $TC_i^Y = 5B$ for any i. Thus, we find two permutations ϕ and φ of $\{1, \dots, m\}$ such that $u_i + v_{\phi(i)} + w_{\varphi(i)} = B$ for any $i = 1, \dots, m$, which implies that I_M has "yes" answer. □

The result of Theorem 2.1 excludes the possibility that the $P||(\sum C_j)_{\max}$ admits a FPTAS. However, if m is a fixed number, we can develop a pseudo-polynomial time dynamic programming algorithm *DP*. In other words, the problem is *NP*-hard in the ordinary sense.

Denote by $[c_1, \dots, c_m, t_1, \dots, t_m]$ the state vector of a partial schedule without idle time, where c_i and t_i are the current completion time and current total completion time of M_i, respectively, $i = 1, \dots, m$. Let the initial state space be $F_0 = \{[0, \dots, 0, 0, \dots, 0]\}$. Let F_j be the set of state vectors for schedules of the first j jobs, $j = 1, 2, \dots, n$. Then the state space F_j can be recursively generated from F_{j-1} as follows.

For any schedule of the first $j-1$ jobs, which corresponds to a state vector $[c_1, \dots, c_m, t_1, \dots, t_m]$ of F_{j-1}, J_j can be assigned to any one of the m machines. By Lemma 2.1, we can assume that J_j is the current last job on the machine where it is assigned. Specifically, if