Information Sciences 445–446 (2018) 1–5

Contents lists available at ScienceDirect

Information Sciences

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

Single-machine scheduling with operator non-availability to minimize total weighted completion time

Long Wan[a], Jinjiang Yuan[b,*]

[a] School of Information Technology, Jiangxi University of Finance and Economics, Nanchang, Jiangxi 310013, PR China
[b] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, PR China

ARTICLE INFO

Article history:
Received 21 June 2017
Revised 29 January 2018
Accepted 3 March 2018
Available online 5 March 2018

MSC:
90B35
90C27

Keywords:
Single-machine scheduling
Non-availability period
Dynamic programming
Fully polynomial-time approximation scheme

ABSTRACT

In this paper, we investigate the single-machine scheduling with an operator non-availability period to minimize total weighted completion time, where the operator non-availability period is an open time interval in which no job can be started or be completed. For this problem, we present a pseudo-polynomial-time algorithm and a fully polynomial-time approximation scheme. Our results address two open problems proposed in Chen et al. [2].

1. Introduction

Brauner et al. [1] introduced the scheduling model on a single machine with an operator non-availability period, where the operator non-availability period is an open time interval (a, b) such that no jobs can be started or completed in (a, b). Practical applications of this scheduling model can be found in Brauner et al. [1] and Rapine et al. [8]. We denote this scheduling model by $1|ON|f$, where "ON" means the operator non-availability constraint and f is the scheduling cost to be minimized.

The scheduling model $1|ON|f$ is closely related to the single-machine non-preemptive scheduling with a machine non-availability period (a, b), denoted $1, h_1||f$, which was extensively studied in the last decades. Rich achievements on scheduling with machine availability constraints can be found in Lee [6], Ma et al. [7], and Schmidt [9]. Especially, Kacem and Mahjoub [3] presented an $O(n^2/\varepsilon^2)$-time FPTAS for problem $1, h_1||\Sigma w_j C_j$. Differential approximability of problem $1, h_1||\Sigma w_j C_j$ was also analyzed in Kacem and Paschos [5].

Brauner et al. [1] noticed that when $p_{\max} < b - a$, where $p_{\max}$ is the maximum processing time of the jobs, the machine cannot process any job in the interval (a, b), and so, the two problems $1|ON|f$ and $1, h_1||f$ are equivalent. Lee [6] showed that problems $1, h_1||C_{\max}$ and $1, h_1||\Sigma C_j$ are binary NP-hard. As consequences, both problems $1|ON|C_{\max}$ and $1|ON|\Sigma C_j$ are binary NP-hard.

* Corresponding author.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

https://doi.org/10.1016/j.ins.2018.03.005
0020-0255/

Interestingly, Brauner et al. [1] showed that problem $1|\mathrm{ON}|C_{\max}$ is solvable in polynomial time when $p_{\min} \geq b - a$, where $p_{\min}$ is the minimum processing time of the jobs, but this is not the case for problem $1, h_1||C_{\max}$, which is still binary NP-hard even if $p_{\max} \geq b - a$.

Chen et al. [2] studied problem $1|\mathrm{ON}|\Sigma C_j$. They showed that the problem is binary NP-hard even if $p_{\min} \geq b - a$, and presented a $\frac{20}{17}$-approximation algorithm for problem $1|\mathrm{ON}|\Sigma C_j$. The approximation result in Chen et al. [2] matches the $\frac{20}{17}$-approximation algorithm for problem $1, h_1||\Sigma C_j$ presented in Sadfi et al. [10]. Recently, by borrowing the FPTAS for scheduling with machine non-availability, Kacem et al. [4] presented an FPTAS for problem $1|\mathrm{ON}|L_{\max}$.

The analysis and discussion for problem $1|\mathrm{ON}|\Sigma C_j$ in Chen et al. [2] are more complicated than that for problem $1, h_1||\Sigma C_j$ in the corresponding literature, for example, Lee [6] and Sadfi et al. [10]. Then Chen et al. [2] presented the following conclusions in their paper:

It appears that the problems with machine operator non-availability periods are more difficult than problems with machine non-availability periods. Therefore, it is challenging to design approximation schemes or study the problem where job have different weights. In fact, whether the problem considered in this paper is strongly NP-hard or even APX-hard remains open.

In this paper, we address the open problems presented in Chen et al. [2] by establishing the following two results:

- Problem $1|\mathrm{ON}|\Sigma w_jC_j$ is solvable in $O(n^2aW)$ time, and problem $1|\mathrm{ON}|\Sigma C_j$ is solvable in $O(n^3a)$ time, where W is the total weight of the jobs.
- Problem $1|\mathrm{ON}|\Sigma w_jC_j$ admits a $(1+\varepsilon)$-approximation algorithm with time complexity $O(n^3/\varepsilon^3)$. In our algorithm, we borrow the $O(n^2/\varepsilon^2)$-time FPTAS for problem $1, h_1||\Sigma w_jC_j$ presented in Kacem and Mahjoub [3].

The paper is organized as follows. We state some useful notations and present the problem formulation in the next section. In Section 3, we develop a dynamic programming algorithm to achieve an optimal solution. We construct a fully polynomial-time approximation scheme in Section 4.

2. Notations and problem statement

Suppose that we have n jobs $J_1, J_2, \ldots, J_n$ to be processed on a single machine non-preemptively. Each job J_j is available at time 0 and has a processing time $p_j \geq 0$ and a weight $w_j \geq 0$. There is an operator non-availability period, denoted (a, b), on the machine, where $0 < a < b$. This means that, in a feasible schedule σ, each job J_j must satisfy the condition

$$S_j(\sigma) \notin (a, b) \text{ and } C_j(\sigma) \notin (a, b), \tag{1}$$

where $S_j(\sigma)$ and $C_j(\sigma)$ are the starting time and completion time of job J_j in σ. We assume in this paper that all the parameters a, b, p_j, and w_j are integers.

Let $\Delta = b - a$ be the length of the operator non-availability period. We assume that $\sum_{j=1}^{n} p_j > a$, for otherwise, we can complete all the jobs prior to the interval (a, b) and the operator non-availability will be meaningless. The scheduling cost to be minimized is the total weighted completion time, i.e., Σw_jC_j. By using the well-known three-field notation, we denote the problem by $1|\mathrm{ON}|\Sigma w_jC_j$, where "ON" means the operator non-availability.

In a feasible schedule σ, if J_j is a job such that $S_j(\sigma) \leq a$ and $C_j(\sigma) \geq b$, we call J_j a *crossover* job. Then there is at most one crossover job in any feasible schedule and, if J_j is a crossover job, we have $p_j \geq \Delta$. For convenience, we introduce a dummy job J_{n+1} with $p_{n+1} = \Delta$ and $w_{n+1} = 0$ so that the following property holds. Let $\mathcal{J} = \{J_1, J_2, \ldots, J_{n+1}\}$ and $\mathcal{J}_c = \{J_j \in \mathcal{J} : p_j \geq \Delta\}$.

Lemma 2.1. *For problem $1|\mathrm{ON}|\Sigma w_jC_j$, there is an optimal schedule such that a certain job in $\mathcal{J}_c$ is crossover.*

Based on Lemma 2.1, for each $J' \in \mathcal{J}_c$, we use $1|\mathrm{ON}, J'|\Sigma w_jC_j$ to denote the restricted problem of $1|\mathrm{ON}|\Sigma w_jC_j$ in which J' must act as a crossover job. Then we can solve all the problems $1|\mathrm{ON}, J'|\Sigma w_jC_j$ for $J' \in \mathcal{J}_c$ and pick the best one for solving the original problem. From the condition in (1), we have

Lemma 2.2. *Given $J' \in \mathcal{J}_c$, for each feasible schedule σ of problem $1|\mathrm{ON}, J'|\Sigma w_jC_j$, we have $\max\{0, b - p'\} \leq S'(\sigma) \leq a$.*

3. A dynamic programming algorithm

Let J' be a job in $\mathcal{J}_c$. Let p' and w' be the processing time and weight of J', respectively. We renumber the n jobs other than J' in $\mathcal{J}$ such that $\mathcal{J} \setminus \{J'\} = \{J_1, J_2, \ldots, J_n\}$ and $p_1/w_1 \leq p_2/w_2 \leq \cdots \leq p_n/w_n$. Moreover, we use $\sigma = (\sigma', J', \sigma'')$ to denote a schedule of problem $1|\mathrm{ON}, J'|\Sigma w_jC_j$, where σ' is the subschedule before J' in σ, and σ'' is the subschedule after J' in σ. The following lemma, which can be proved by job-exchanging argument, is useful for our discussion.

Lemma 3.1. *For problem $1|\mathrm{ON}, J'|\Sigma w_jC_j$, there is an optimal schedule $\sigma = (\sigma', J', \sigma'')$ such that (i) the jobs in σ' are scheduled consecutively with no idle time from time 0 in their index order, (ii) the jobs in (J', σ'') are scheduled consecutively with no idle time, (iii) the jobs in σ'' are scheduled in their index order, and (iv) subject to the condition in (1), J' is scheduled as earlier as possible.*