Discrete Optimization 17 (2015) 69–79

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Single-machine scheduling with supporting tasks

Alexander V. Kononov[a], Bertrand M.T. Lin[b,*], Kuei-Tang Fang[b]

[a] *Sobolev Institute of Mathematics, Russian Academy of Sciences, Novosibirsk, Russia*
[b] *Institute of Information Management, National Chiao Tung University, Hsinchu, Taiwan*

A R T I C L E I N F O

Article history:
Received 18 November 2014
Received in revised form 24 April 2015
Accepted 4 May 2015
Available online 28 May 2015

Keywords:
Single-machine scheduling
Supporting precedence
Total weighted completion time
Number of late jobs

A B S T R A C T

This paper investigates a single-machine scheduling problem with a set of supporting tasks and a set of jobs. Each job is preceded by a subset of supporting tasks, that is, the job cannot start its processing until all of its supporting tasks are finished. The objective functions are defined solely in job completion times. We discuss the complexities of several special cases for the number of late jobs and the total weighted completion time. This study adds new complexity results to the two standard objective functions under precedence constraints.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

We consider single-machine scheduling problems with supporting tasks. Two disjoint sets of activities $A = \{a_1, a_2, \dots, a_m\}$ and $B = \{b_1, b_2, \dots, b_n\}$ are to be processed on a single machine. The elements of set A are called *supporting tasks*, and the elements of set B are called *jobs*. The processing times of task a_i and job b_j are denoted by α_i and β_j, respectively. A supporting relation $\mathcal{R} : A \to B$ is a mapping from set A to set B such that for $a_i \in A$ and $b_j \in B$, if (a_i, b_j) belongs to $\mathcal{R}$ then job b_j cannot start unless task a_i is completed. Each job b_j is associated with a weight w_j and a due date d_j. Denote the completion time of job $b_j \in B$ in a particular schedule by C_j. If job b_j is late, that is $C_j - d_j > 0$, then we set binary variable $U_j = 1$; 0, otherwise. For each task $a_i \in A$, denote $\mu_i = \{b_j \in B | (i, j) \in \mathcal{R}\}$ as the set of jobs supported by a_i. Similarly, for each job $b_j \in B$, denote $\nu_j = \{a_i \in A | (i, j) \in \mathcal{R}\}$ the set of tasks that support b_j. Although all supporting tasks are required to be processed, they do not contribute to the objective functions because their roles are simply preparatory operations for the jobs that they support.

The proposed model is motivated by real-world applications where preparatory operations are required before jobs can be processed. In the context of multi-media scheduling, a playback comprises several media

* Corresponding author. Tel.: +886 3 5131472; fax: +886 3-5723792.
E-mail address: bmtlin@mail.nctu.edu.tw (B.M.T. Lin).

http://dx.doi.org/10.1016/j.disopt.2015.05.001
1572-5286/© 2015 Elsevier B.V. All rights reserved.

objects, including for example, audio, video, icons, and texts. Once a media object is prepared, either downloaded from a digital archive or created at the local site, it can be embedded in several playbacks. Media objects and playbacks are referred to as supporting tasks and jobs, respectively, in the defined scheduling setting. The roaring lion of the MGM films and special animation effects of some TV programs are examples of media objects that are produced once and used many times as appropriate. This unique property introduces a new type of bill of materials and makes the scheduling of on-line media production different from the manufacturing of tangible products, like for example car, cloth, and drink. Another scenario describing the setting is tool or material preparation in a manufacturing environment. A job may require a set of tools installed before its processing. On the other hand, a tool, once installed, may be required by several jobs. In summary, the scheduling setting is applicable to scheduling contexts where preparatory operations occupy a limited resource (the machine) but are not taken into account in the objective functions.

The rest of this paper is organized as follows. In Section 2, we review the related works and summarize the new results conveyed by this paper. Section 3 is dedicated to the objective function of the number of tardy jobs. Minimization of the total weighted completion time is addressed in Section 4. Section 5 discusses special cases where either a task sequence or a job sequence is given *a priori*. We conclude this study and suggest potential research issues in Section 6.

2. Related results and our contributions

This paper discusses two min-sum objective functions: the number of late jobs ($\sum_j U_j$) and the total weighted completion time of jobs ($\sum_j w_j C_j$). Denote the problem by the three-field notation $1|s-prec|\gamma$, where $s-prec$ dictates the supporting precedence and $\gamma \in \{\sum w_j C_j, \sum U_j\}$. We note that the classical min–max criteria in this setting can be easily solved. The makespan of any feasible schedule is fixed. As for minimization of the maximum lateness or tardiness, we can associate the supporting tasks with an infinite due date and then deploy the dynamic programming algorithm of Lawler [1].

In the next section, we consider the minimization of the number of late jobs. Karp [2] shows that the single-machine problem of minimizing the weighted number of late jobs ($1 \parallel \sum w_j U_j$) is NP-hard and proposes a pseudo-polynomial dynamic programming algorithm. The case where all jobs are equally weighted can be solved in polynomial time by Moore–Hodgson algorithm [3]. If all jobs have a unit execution time ($1|p_j = 1|\sum w_j U_j$), then the problem can be solved as follows: For each position starting backward from $\max\{d_j\}$, the non-tardy job with the largest weight is assigned, breaking a tie on weights by assigning the job with the latest due date. The problem with general precedence constraints is strongly NP-hard, even if all jobs are equally weighted and have a unit processing time [4]. The proof is based on a reduction from the Maximum Clique problem to $1|prec, p_j = 1|\sum w_j U_j$. The technique also works in our study. Given a graph G, we treat each vertex as a supporting task and each edge as a job. Let all jobs have a common deadline. Choosing this deadline appropriately we can prove the strong NP-hardness of $1|s-prec|\sum U_j$, even if $\alpha_i = \beta_j = 1$ for all a_i and all b_j or if $\alpha_i = 1$ for all a_i and $\beta_j = 0$ for all b_j. In both cases each job requires exactly two supporting tasks, i.e. $|\nu_j| = 2$. Moreover, the second case is equivalent to the well-known Dense-K-Subgraph problem studied by Feige, Peleg and Kortsarz [5]. In Section 3 we consider several restricted cases where each job is supported by exactly one task, i.e. $|\nu_j| = 1$. More precisely, we prove the strong NP-hardness of $1|s-prec, |\nu_j| = 1, \beta_j = 0|\sum U_j$, $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 1|\sum U_j$, $1|s-prec, |\mu_i| = 3, |\nu_j| = 1, \alpha_i = 1|\sum U_j$ and the NP-hardness of $1|s-prec, |\mu_i| = 2, |\nu_j| = 1|\sum U_j$. Certainly, these results immediately imply the NP-hardness of the corresponding problems of the weighted number of tardy jobs. We also show that $1|s-prec, |\nu_j| = 1, \alpha_i = 1, \beta_j = 0|\sum w_j U_j$ can be solved in $O(n + m^3)$ time. Please see Table 1 for a summary of the results.

Lenstra and Rinnooy Kan [6] prove the strong NP-hardness of the $1|chains, p_j = 1|\sum U_j$ problems. But this approach is not applicable to our problem. In their proof, the precedence graph consists of several long

Download English Version:

https://daneshyari.com/en/article/1141662

Download Persian Version:

https://daneshyari.com/article/1141662

Daneshyari.com