Computers & Operations Research 73 (2016) 115–118

Contents lists available at ScienceDirect

Computers & Operations Research

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

A single machine scheduling problem to minimize total early work

Yoav Ben-Yehoshua, Gur Mosheiov *

School of Business Administration, The Hebrew University, Jerusalem 91905, Israel

ARTICLE INFO

Article history:
Received 4 May 2015
Received in revised form
25 November 2015
Accepted 19 March 2016
Available online 25 March 2016

Keywords:
Scheduling
Single machine
Total early work
Dynamic programming

ABSTRACT

We study a single machine scheduling problem, where the objective is minimum total early work. In this setting, a job is penalized according to the duration of the parts of the job completed prior to its due-date. First we prove that the problem is NP-hard. Then, based on a number of properties of an optimal schedule, we introduce a pseudo-polynomial dynamic programming algorithm, verifying NP-hardness in the ordinary sense. Our numerical tests indicate that the dynamic programming solves problems of hundreds of jobs in very reasonable time.

1. Introduction

In a recent survey, Sterna [7] introduced the various published papers on "scheduling papers with late work criteria". In this class of scheduling models, the quality of the solution is measured by the duration of the parts of the jobs scheduled after their due-dates. Thus, the penalty on job lateness is bounded by the job processing time. One can describe late work as "the number of tardy job units" (see [3]). Potts and Van Wassenhove [5] provided the most significant results in scheduling with total late work: they proved NP-hardness, introduced an exact dynamic pseudo-polynomial programming algorithm, and solved several special cases. Sterna [7] provides a number of applications of the late work criterion, and claims that in general, it appears to be relevant to all settings where the penalty depends on the number of tardy jobs performed in the system, regardless of how late these are. Some applications arise in control systems [2,5], computerized control systems [4], batch scheduling problems [6], various issues in agriculture [1], and systems containing perishable items [5].

In this paper we focus on a single machine scheduling problem, where the objective is to minimize the *total early work*. In contrast to the measure of total late work, in this setting, a job is penalized according to the duration of the parts of the job completed prior to its due-date. Thus, the maximum penalty of a job is equal to its processing time (in the case that it is fully early). A typical application of such system is in a distributed computing setting, where a continuously working server needs to transfer calculation results from one computer to another. In the case that the results arrive at the server before the receiver is available, the server will have to write these large data sets onto its hard disk, rather than transferring them directly to the receiver. Clearly, if the receiver is available, the process of writing the data as an intermediate process is avoided. Thus, the goal is to minimize the server's writing process, which becomes significant, as mentioned, when transferring large data sets that must be written onto a hard disk. In such systems, only early completed jobs are penalized, and the penalty is proportional to the length/size of the job.

As commonly assumed in scheduling problems involving earliness measures, we consider here only *non-delay* schedules. (Otherwise, optimality is trivially obtained by sufficiently delaying the jobs.) Such schedules (with no idle times prior to the first job and between consecutive jobs) are justified in many manufacturing systems, where the production process cannot be stopped until the entire set of jobs is finished. We prove that the problem studied here is NP-hard. Then, we propose a pseudo-polynomial dynamic programming (DP) algorithm, implying that the problem is NP-hard in the ordinary sense. Our numerical tests indicate that the proposed DP is very efficient, and the solution of problems of medium size (up to 200 jobs) requires very reasonable computational effort.

In Section 2 we provide the notation and formulation of the problem. Section 3 contains the proof of NP-hardness. In Section 4 we introduce the dynamic programming algorithm, and report the results of the numerical tests.

2. Formulation

Consider a set of n jobs that need to be processed on a single machine with no idle times. The processing time of job j is

* Corresponding author.
E-mail address: msomer@mscc.huji.ac.il (G. Mosheiov).

http://dx.doi.org/10.1016/j.cor.2016.03.010
0305-0548/

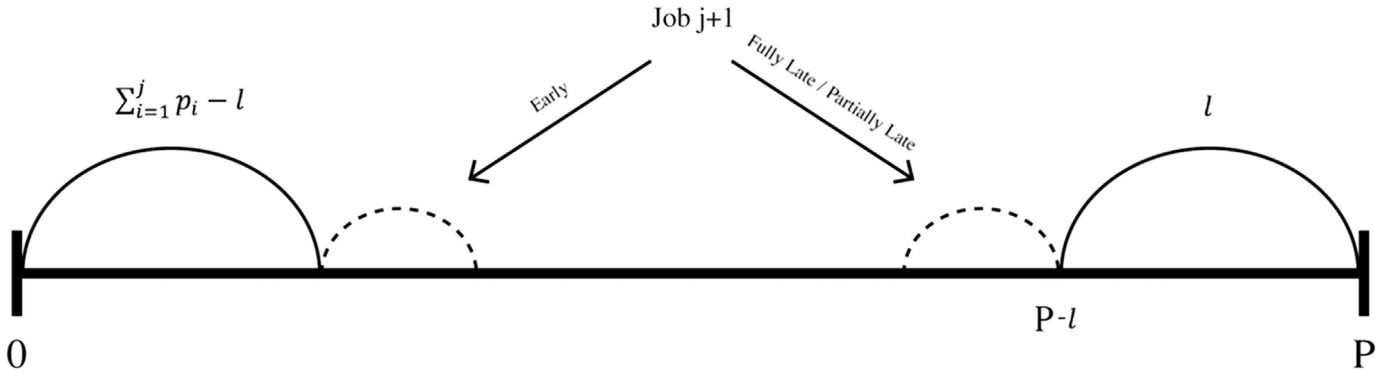

Fig. 1. The state variables and the possible decisions at stage $j+1$ of the DP.

Table 1
Data for Example 1.

Job	1	2	3	4	5	6	7
p_j	9	6	10	6	10	2	5
d_j	6	16	22	32	40	50	55

denoted by $p_j, j = 1, \ldots, n$. d_j is the due-date of job j, $j = 1, \ldots, n$. We denote by P the sum of all the processing times ($P = \Sigma p_j$). For a given job sequence, S_j and C_j are the job starting time and completion time, respectively, $j = 1, \ldots, n$. We note that if job j starts processing after its due-date, it is not penalized. Otherwise, the penalty is a function of the early part of the job. Thus, if a job is completed prior to the due-date, its penalty is identical to its entire processing time. Formally, let $EW_j=\min\{\max\{d_j-S_j,0\}, p_j\}$ denote the *Early Work* of job $j, j = 1, \ldots, n$. An identical definition is the following:

$$EW_j=\begin{cases} p_j & \text{if } C_j\le d_j \\ d_j-S_j & \text{if } S_j\le d_j\le C_j \\ 0 & \text{if } d_j\le S_j \end{cases}$$

The objective function considered here is minimum total Early Work, i.e.,

$$EW=min\left\{\sum_{j=1}^{n} EW_j\right\}.$$

3. NP-hardness

In this section we prove by a standard reduction from 2-Partition, NP-hardness of the problem of minimizing total early work on a single machine.

2-Partition: Given a set N of n integers: $a_j, j = 1, \ldots, n$, and $\sum_{j\in N} a_j=2b$, is there a subset $S \subseteq N$, such that $\sum_{j\in S} a_j=b$?

We construct the following instance of the recognition version of our scheduling problem (denoted RC) as follows:

There are $n + 1$ jobs with:

$p_j=a_j, j=1, \ldots,n; p_{n+1}=1.$

$d_j=b+1, j=1, \ldots, n; d_{n+1}=b.$

Is there a schedule with total early work not larger than b?

(2-Partition $\Longrightarrow$ RC): If a solution to 2-Partition exist, schedule the jobs of the set S to start at time zero (and completed at b), then schedule job $n + 1$ and then the remaining jobs. The total early work is clearly b.

(RC $\Longrightarrow$ 2-Partition): We claim that if a schedule exist with total early work not larger than b, it must contain job $n + 1$ to start at time b (implying that a subset of the jobs must be completed exactly at time b, as required). Consider an optimal schedule q with job $n + 1$ starting at time $b + \Delta$, $\Delta > 0$. Due to the non-delay property, the total early work is $b + \min\{\Delta, 1\} > b$, contradicting the optimality of q. Consider now an optimal schedule q' with job $n + 1$ starting at time $b - \Delta$, $\Delta > 0$. Job $j + 1$ must be either early or partially early, and its earliness work is $\min\{\Delta, 1\}$. As above, due to the non-delay property, the total early work is $b + \min\{\Delta, 1\} > b$, contradicting the optimality of q'. ∎

4. A dynamic programming solution algorithm

In this section we present a pseudo-polynomial dynamic programming algorithm for the problem, implying that the problem is NP-hard in the ordinary sense. We begin by introducing two properties of an optimal schedule.

Property 1. There exists an optimal schedule consisting of a set of fully early jobs, followed by a set of fully late or partially late jobs.

Proof. Consider an optimal schedule S in which job k precedes job j, job k is fully or partially late and job j is fully early. Assume that in S a set of jobs B is scheduled prior to k, and a set of jobs A is scheduled after job j. Create a new schedule S' by moving job j to be scheduled at time zero. Note that: (i) $EW_j(S')=EW_j(S)=p_j$; (ii) $EW_i(S') \le EW_i(S)$, $i \in B$ (the jobs in B are delayed in S' by an amount of p_j); (iii) $EW_k(S') \le EW_k(S)$ (job k is delayed in S' by an amount of p_j); (iv) $EW_l(S') = EW_l(S)$, $l \in A$ (the jobs in A are not affected when moving job j). It follows that S' is optimal as well. By repeating this procedure for all pairs of jobs where the first is not fully early and the second is fully early, we obtain an optimal schedule consisting of a first set of fully early jobs followed by a set of fully or partially late jobs. ∎

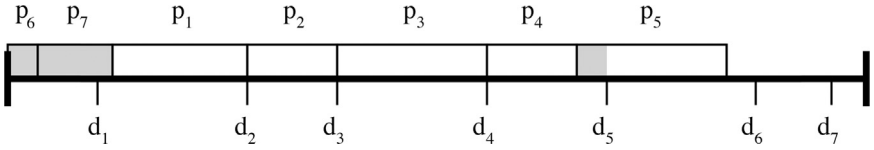

Fig. 2. Solution of Example 2 (total early work contains the processing times of jobs 6 and 7, and two units of job 5).