Applied Mathematical Modelling 35 (2011) 2005–2015

Contents lists available at ScienceDirect

Applied Mathematical Modelling

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

Scheduling linearly shortening jobs under precedence constraints

Stanisław Gawiejnowicz [a,*], Tsung-Chyan Lai [b], Ming-Huang Chiang [b]

[a] Adam Mickiewicz University, Faculty of Mathematics and Computer Science, Umultowska 87, 61-614 Poznań, Poland
[b] National Taiwan University, College of Management #2, Roosevelt Road 85, Section 4, Taipei 106, Taiwan

ARTICLE INFO

Article history:
Received 26 April 2009
Received in revised form 27 September 2010
Accepted 12 November 2010
Available online 19 November 2010

Keywords:
Time-dependent scheduling
Shortening jobs
Precedence constraints
Polynomial algorithms

ABSTRACT

We consider the problem of scheduling a set of dependent jobs on a single machine with the maximum completion time criterion. The processing time of each job is variable and decreases linearly with respect to the starting time of the job. Applying a uniform approach based on the calculation of ratios of expressions that describe total processing times of chains of jobs, we show basic properties of the problem. On the basis of these properties, we prove that if precedence constraints among jobs are in the form of a set of chains, a tree, a forest or a series–parallel digraph, the problem can be solved in $O(n\log n)$ time, where n denotes the number of the jobs.

1. Introduction

In the classic scheduling theory (Conway et al. [1]), one assumes that job processing times are known in advance fixed values. The assumption, however, strongly restricts the applicability of the theory, since in many production environments job processing times are variable. For example, the processing times of jobs increase (decrease) if the jobs are executed on machines with decreasing (increasing) efficiency, jobs processed by robots or automated guided vehicles need variable amount of time to execute in dependence of the machines processing speed etc.

The phenomenon of variability of job processing times is modelled in scheduling literature in a few different ways. For example, the processing time of a job can be a function of a continuous resource (Gawiejnowicz [2]), the job waiting time (Sriskandarajah and Goyal [3]), the number of already executed jobs (Gawiejnowicz [4]) or the position of the job in schedule (Bachman and Janiak [5], Biskup [6], Wu and Lee [7]).

Scheduling problems with variable job processing times can also be considered in the framework of *time-dependent scheduling* (Alidaee and Womer [8], Cheng et al. [9], Gawiejnowicz [10]), a rapidly developing branch of modern scheduling theory. In time-dependent scheduling the processing time of a job depends on the starting time of the job. This assumption considerably extends the area of applicability of scheduling theory, since numerous real-life problems can be modelled as time-dependent scheduling problems. For example, the problems of repayment of multiple loans (Gupta et al. [11]), producing ingots in a steel mill (Kunnathur and Gupta [12]), recognizing aerial threats (Ho et al. [13]), maintenance assignments (Mosheiov [14]), assignment of divisible loads in a multiprocessor environment (Drozdowski [15]), fire fighting (Rachaniotis and Pappis [16]) and scheduling derusting operations (Gawiejnowicz et al. [17]) can be formulated as time-dependent scheduling problems.

In general, the functions that describe job processing times in a time-dependent scheduling problem can be arbitrary non-negative functions of time. In time-dependent scheduling literature, however, most intensively studied are non-decreasing

* Corresponding author. Tel.: +48 61 829 5334; fax: +48 61 829 5315.
E-mail addresses: stgawiej@amu.edu.pl (S. Gawiejnowicz), tclai@ccms.ntu.edu.tw (T.-C. Lai), cmh@ccms.ntu.edu.tw (M.-H. Chiang).

0307-904X/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.apm.2010.11.012

and non-increasing functions. If the functions are non-decreasing, we deal with *deteriorating* processing times of jobs; if the functions are non-increasing, we consider *shortening* job processing times.

Though the huge majority of time-dependent scheduling literature concerns the case when jobs are independent (i.e. when job precedence constraints are empty), there are known some results concern dependent deteriorating jobs. Tanaev et al. [18], using *priority-generating functions*, have formulated $O(n\log n)$ algorithms for a set of linearly deteriorating jobs executed on a single machine and job precedence constraints in the form of a tree or a series–parallel graph. A similar result for series–parallel precedence constraints was obtained by Wang et al. [19]. For a detailed discussion of scheduling deteriorating jobs with different forms of polynomially solvable job precedence constraints, we refer the reader to monograph [10, Chapter 13].

Time-dependent scheduling with dependent shortening jobs was considered by Gordon et al. [20]. Applying priority-generating functions, the authors proved polynomial solvability of series–parallel precedence constraints.

Throughout this paper, we consider the following time-dependent scheduling problem. There is given a set $\mathcal{J}$ of jobs $J_1,J_2,\ldots,J_n$ to be processed on a single machine. The machine is available for processing at time $t_0 \geqslant 0$. We assume that job processing times are shortening, i.e. the processing time of job J_j is equal to $p_j = a_j - b_jS_j$, where S_j is the starting time of the job, the job basic processing time $a_j > 0$ and the job deterioration rate $0 < b_j < 1$ for $1 \leqslant j \leqslant n$. We also assume that for $1 \leqslant j \leqslant n$ there hold inequalities

$$b_j\left(\sum_{i=1}^{n} a_i - a_j\right) < a_j. \tag{1}$$

We restrict our further considerations only to schedules without idle times between jobs. This, together with inequalities (1), causes that job processing times remain positive in all possible schedules (see Ho et al. [13] for details).

In the set $\mathcal{J}$ there are also defined non-empty job precedence constraints, i.e. a reflexive, antisymmetric and transitive relation on the Cartesian product $\mathcal{J}\times\mathcal{J}$. These precedence constraints can be given in the form of a set of chains, a tree, a forest or a series–parallel digraph.

Given the set of jobs defined as above, our aim is to find such a schedule that minimizes the maximum completion time, $C_{\max} := \max_{1\leqslant j\leqslant n}\{C_j\}$, where C_j denotes the completion time of job J_j.

In this paper, applying a uniform approach other than priority-generating functions, we show that for the mentioned forms of job precedence constraints the above problem can be solved in $O(n\log n)$ time. For each case of job precedence constraints, we also propose a polynomial algorithm and prove its optimality. Hence, our results complement the result by Gordon et al. [20], since the authors have not presented details concerning algorithms for the case of series–parallel precedence constraints and its subcases.

The remainder of the paper is organized as follows. In Section 2, we recall some graph definitions used in the paper. In Section 3, we present basic properties of the problem. In subsequent sections, we consider different forms of job precedence constraints: a set of chains (Section 4), a tree or a forest (Section 5), a series–parallel digraph (Section 6). Conclusions are given in Section 7.

2. Preliminaries

In the section, we recall some graph definitions used throughout the paper. We start with definitions concerning directed graphs.

A directed graph (a digraph) is an ordered pair $G = (V,A)$, where $V \neq \emptyset$ is a finite set of vertices and $A \subseteq \{(v_1,v_2) \in V\times V : v_1 \neq v_2\}$ is a set of arcs.

A directed path in a digraph $G = (V,A)$ is a sequence $(v_1, v_2,\ldots, v_k)$ of distinct vertices from V such that $(v_i, v_{i+1}) \in A$ for each $1 \leqslant i \leqslant k-1$. The number k is called the length of the path $(v_1, v_2,\ldots, v_k)$.

A directed cycle in a digraph $G = (V,A)$ is a directed path $(v_1, v_2,\ldots, v_k)$ such that $v_k = v_1$. A digraph $G = (V,A)$ is an acyclic digraph if it contains no directed cycle.

A digraph $G = (V,A)$ is connected if for every $v_1, v_2 \in V$ there exists in G a directed path starting with v_1 and ending with v_2; otherwise, it is disconnected.

A digraph $G' = (V',A')$ is called a subdigraph of a digraph $G = (V,A)$, if $V' \subseteq V$ and $(v_1,v_2) \in A'$ implies $(v_1,v_2) \in A$.

A vertex $v_1 \in V$ of a digraph $G = (V,A)$ is called a predecessor (successor) of a vertex $v_2 \in V$, if there exists a directed path from v_1 to v_2 (from v_2 to v_1). For a given digraph $G = (V,A)$ and $v \in V$, we denote the set of all predecessors and successors of v by $Pred(v)$ and $Succ(v)$, respectively.

If vertices $v_1, v_2 \in V$ of a digraph $G = (V,A)$ and the directed path from v_1 to v_2 is of unit length, then v_1 is called a direct predecessor (successor) of v_2. A vertex $v \in V$ of a digraph $G = (V,A)$ that has no direct predecessor (successor) is called an initial (a terminal) vertex in the digraph. A vertex $v \in V$ of a digraph $G = (V,A)$ that is neither initial nor terminal is called an internal vertex in the digraph.

A chain $(v_1, v_2,\ldots, v_k)$ is a digraph $G = (V,A)$, where $V = \{v_i : 1 \leqslant i \leqslant k\}$ and $A = \{(v_i, v_{i+1}) : 1 \leqslant i \leqslant k-1\}$.

A special case of a chain is an independent chain.

Definition 1. A chain $(v_1, v_2,\ldots, v_k)$ in a digraph $G = (V,A)$ is said to be an *independent chain*, if for any $v_i \in V\backslash\{v_1, v_2,\ldots, v_k\}$ the vertex v_i is neither predecessor nor successor of any vertex from the chain $(v_1, v_2,\ldots, v_k)$, or v_i precedes (follows) all vertices of the chain $(v_1, v_2,\ldots, v_k)$.