Discrete Applied Mathematics 196 (2015) 28–53

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Algorithms for scheduling with integer preemptions on parallel machines to minimize the maximum lateness

Y. Zinder *, S. Walker

The University of Technology, Sydney, P.O. Box 123, Broadway, NSW 2007, Australia

ARTICLE INFO

Article history:
Received 31 October 2013
Received in revised form 31 December 2014
Accepted 2 May 2015
Available online 27 May 2015

Keywords:
Scheduling
Maximum lateness
Parallel machines
Integer preemptions
Precedence constraints

ABSTRACT

The paper is concerned with the problem of scheduling on parallel identical machines partially ordered tasks that can be preempted at integer points in time. The objective function is the maximum lateness. The well-known algorithms for scheduling unit execution time tasks and for scheduling with arbitrary preemptions are not directly applicable to the considered problem. The paper presents a new method for scheduling with integer preemptions and the worst-case analysis for several polynomial-time algorithms which have this method as a core scheduling routine.

1. Introduction

This paper presents polynomial-time algorithms and analyses their worst-case performance for the following scheduling problem. A finite set of tasks (jobs, operations) $N = \{1, 2, \ldots, n\}$ is to be processed on $m > 1$ identical parallel machines (processors). The processing of tasks begins at time $t = 0$. Each machine can process only one task at a time, and each task can be processed by only one machine at a time. Since all machines are identical, it is not important what machine actually processes a task, provided that the number of tasks, processed simultaneously, does not exceed m.

The order in which tasks can be processed is restricted by precedence constraints — a transitive, antireflexive and antisymmetric relation on N. If task j precedes task g, denoted $j \to g$, then task g cannot be processed until task j has been completed. In this case, g is called a successor of j and j is a predecessor of g. Task j is an immediate predecessor of task g if $j \to g$ and there is no $q \in N$ such that $j \to q \to g$. Task g is an immediate successor of j if j is an immediate predecessor of g.

The processing of a task can commence only at an integer point in time. At any integer point in time, the processing of a task can be interrupted and resumed later at another integer point in time. The processing time of $j \in N$ will be denoted p_j and is an integer. In other words, p_j is the integer number of time units that j should accumulate in order to be completed.

The constraints on the points in time where preemptions can occur permit more accurate modeling of various practical situations. For example, it permits modeling of the discrete nature of CPU clock cycles. More accurate models with limited preemptions have been subject to increasing attention in recent years, for example [3].

A schedule σ specifies, for each task j, a sequence of time units where j is processed. The right end of the last of these time units is denoted by $C_j(\sigma)$ and is referred to as the completion time of j in schedule σ. The goal is to minimize the maximum lateness

$$L_{max}(\sigma) = \max_{j \in N}\{C_j(\sigma) - d_j\},$$

where d_j is the integer due time (due date) by which it is desired to complete j.

* Corresponding author.
E-mail address: yakov.zinder@uts.edu.au (Y. Zinder).

http://dx.doi.org/10.1016/j.dam.2015.05.005
0166-218X/

If all $p_j = 1$, then no preemptions are involved and the considered scheduling problem becomes the classical maximum lateness problem with unit execution time (UET) tasks. In the standard three-field notation (see for example [1] or [9]) the latter problem is denoted by $P|prec, p_j = 1|L_{max}$. It is well-known that $P|prec, p_j = 1|L_{max}$ is NP-hard in the strong sense [11].

The problem with integer preemptions can be reduced to $P|prec, p_j = 1|L_{max}$ by replacing each $j \in N$ by a chain of p_j UET tasks. Unfortunately, this reduction cannot be accomplished in an amount of time polynomial in n. Therefore, the known polynomial-time approximation algorithms for $P|prec, p_j = 1|L_{max}$ are not directly applicable.

The relaxation of the restriction that tasks can commence processing only at integer points in time and that the preemptions and resumptions of processing can occur only at integer points in time gives the well-known maximum lateness problem with preemptions. This scheduling problem is denoted in the three-field notation by $P|prmp, prec|L_{max}$.

The problems $P|prec, p_j = 1|L_{max}$ and $P|prmp, prec|L_{max}$ are closely related. Thus, the algorithms in [2,4] and [12], which were developed for the $P|prec, p_j = 1|L_{max}$ problem, were modified in [7,5] and [13] for the problem $P|prmp, prec|L_{max}$. The core of all these modifications is an algorithm which schedules tasks according to their priorities, whereas the idea of the method of calculating priorities was borrowed from the algorithms for $P|prec, p_j = 1|L_{max}$. Two version of this core scheduling algorithm for $P|prmp, prec|L_{max}$ can be found in [7] and [10].

The paper presents a new polynomial-time algorithm for $P|prec, prmp|L_{max}$ which can be viewed as an amalgamation of the algorithms in [7] and [10]. This new algorithm is an alternative to the algorithms in [7] and [10], although in this paper its role is limited to be a subroutine (first phase) in another polynomial-time algorithm for the problem with integer preemptions. The latter algorithm will be referred to as Algorithm IP. The paper also presents modifications, for the case of integer preemptions, of the algorithms originally described in [2,4] and [12] for $P|prec, p_j = 1|L_{max}$. All three modifications are based on Algorithm IP and are characterized by their worst-case performance guarantees.

The structure of the paper is as follows. Section 2 describes Algorithm IP, which constructs a schedule using the tasks' priorities. The computation of tasks' priorities requires that each task j should be assigned a certain parameter μ_j. Section 3 describes three methods of computing these μ's. These three methods are modifications of the ideas originally presented in [2,4] and [12] for $P|prec, p_j = 1|L_{max}$. Each of these three modifications, together with Algorithm IP, gives rise to a scheduling algorithm. A schedule, constructed by the algorithm based on the modification of the idea in [2], will be denoted σ^{BGJ}; a schedule, constructed by the algorithm based on the modification of the idea in [4], will be denoted σ^{GJ}; finally, a schedule, constructed by the algorithm based on the modification of the idea in [12], will be denoted σ^{ZR}.

Section 4 provides insight into the structure of these schedules and any other schedules which may be constructed by algorithms with certain similar properties. Section 4 also presents a performance guarantee for the algorithm, originating from the idea in [12],

$$L_{max}\left(\sigma^{ZR}\right) \leq \left(2 - \frac{2}{m}\right) L_{max}\left(\sigma^{*}\right) + \left(1 - \frac{2}{m}\right) \max_{j \in N} d_j,$$

where $L_{max}\left(\sigma^{ZR}\right)$ is estimated in terms of the optimal maximum lateness $L_{max}\left(\sigma^{*}\right)$, computed for an optimal schedule σ^{*}. It is worth noting that, although the preemptions are allowed only at integer points in time, the expression for the performance guarantee is exactly the same as in [13] where no restrictions on preemptions have been imposed.

Section 5 presents another two performance guarantees

$$L_{max}\left(\sigma^{BGJ}\right) - L_{max}\left(\sigma^{*}\right) \leq \frac{m-1}{m} l \tag{1}$$

and

$$L_{max}\left(\sigma^{GJ}\right) - L_{max}\left(\sigma^{*}\right) \leq \frac{m-2}{m} l, \tag{2}$$

where σ^{*} is an optimal schedule and l is the length of the longest chain in the partially ordered set of tasks, defined as follows. A chain is a set of tasks $S \subseteq N$ such that for all $j \in S$ and $g \in S$ either $j \rightarrow g$ or $g \rightarrow j$. The length of a chain S is $\sum_{j \in S} p_j$. It is worth noting that, although all three scheduling algorithms developed and analysed in this paper are polynomial-time algorithms, the derivations of the performance guarantees (1) and (2) are achieved by a non-polynomial conversion to $P|prec, p_j = 1|L_{max}$.

2. Algorithm IP

For any schedule σ, any task j and any point in time t, the remaining processing time of task j in schedule σ will be denoted by $p_j(t, \sigma)$. In other words, $p_j(t, \sigma)$ is the difference between p_j and the amount of processing time that j has received until time t in schedule σ. The algorithms in [7,5] and [13] are based on the following approach: each task j is assigned a constant μ_j, and then the tasks are scheduled in such a manner that, when tasks compete for machine time at a point in time t, a task with larger $p_j(t, \sigma) + \mu_j$ is considered to have a higher priority. It is important to note that these μ's are byproducts of the algorithms considered, and are not a part of the problem instance to be solved.

All the above mentioned algorithms are comprised of two components — the method of calculating μ's and the method of constructing a schedule using $p_j(t, \sigma) + \mu_j$ as a priority of task j in schedule σ at time t. Algorithm IP, described in this