Discrete Applied Mathematics 157 (2009) 3625–3630

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

A graph coloring approach to scheduling of multiprocessor tasks on dedicated machines with availability constraints

K. Giaro, M. Kubale, P. Obszarski *

Department of Algorithms and System Modeling, Gdańsk University of Technology, Poland

ARTICLE INFO

Article history:
Received 29 November 2007
Received in revised form 3 February 2009
Accepted 13 February 2009
Available online 3 April 2009

Keywords:
Hypergraph coloring
Dedicated machines
Multiprocessor scheduling
NP-completeness

ABSTRACT

We address a generalization of the classical 1- and 2-processor unit execution time scheduling problem on dedicated machines. In our chromatic model of scheduling machines have non-simultaneous availability times and tasks have arbitrary release times and due dates. Also, the versatility of our approach makes it possible to generalize all known classical criteria of optimality. Under these stipulations we show that the problem of optimal scheduling of sparse tree-like instances can be solved in polynomial time. However, if we admit dense instances then the problem becomes NP-hard, even if there are only two machines.

1. Introduction

In recent years a number of new approaches to the problem of parallel computer systems has been proposed. One of them is scheduling of multiprocessor task systems [4]. According to this model any task may require for its processing more than one processor at a time. There are two main classes of problems in multiprocessor task scheduling. In the first class of problems, it is assumed that the *number* of simultaneously required processors is important [2,11] and each task requires any fixed subset of processors whose cardinality is equal to the prescribed number. In the second class of multiprocessor scheduling problems, the *set* of simultaneously required processors is assumed to be important [1,3,9,10]. In this case either a certain fixed set of processors, or a family of processor sets on which the task can be executed is given. This paper deals with the problem of scheduling tasks assigned to a fixed set of processors.

Since the problem of scheduling 2-processor tasks is NP-hard subject to all classical optimality criteria, in this paper we are concerned with the special case in which the duration of every task is the same. We will call such tasks unit execution time (UET) tasks. Consequently, all data are assumed to be positive integers. Later on we will see that such a restricted version, although remaining NP-hard for some instances in dense systems, does allow for polynomial-time algorithms in numerous special cases of sparse systems.

The third important assumption concerns availability constraints. Namely, we assume that the availability of tasks is restricted and, in addition, some machines are available only in certain intervals called time windows. Time windows may appear due to computer breakdowns or maintenance periods. Moreover, in any multitasking computer system and in hard real-time systems in particular, urgent tasks have high priority and are pre-scheduled in certain time intervals, thus creating multiple time windows of availability. Scheduling in time windows was considered in e.g. [1,2,5,6].

* Corresponding author. Tel.: +48 58 347 17 41; fax: +48 58 347 17 66.
E-mail addresses: giaro@eti.pg.gda.pl (K. Giaro), kubale@eti.pg.gda.pl (M. Kubale), pawel.obszarski@eti.pg.gda.pl (P. Obszarski).

0166-218X/$ – see front matter
doi:10.1016/j.dam.2009.02.024

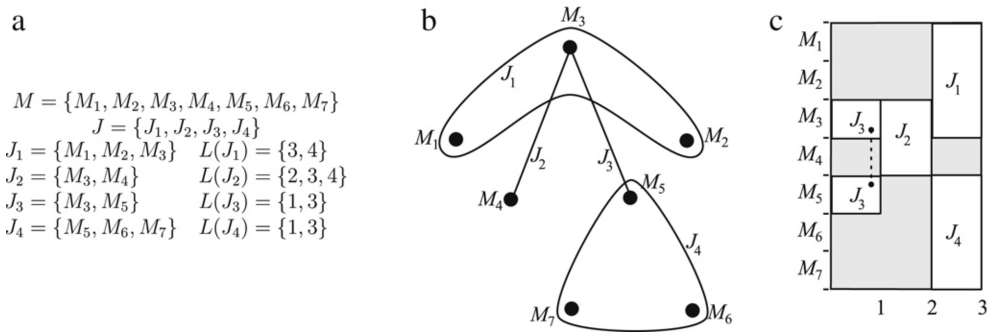

Fig. 1. Example (a) problem instance (b) scheduling hypergraph (c) Gantt diagram.

The rest of the paper is organized as follows. In Section 2 we set up the problem more formally and model it as a cost list edge-coloring (CLEC) problem. Section 3 is devoted to the case when two tasks sharing pairwise at most one machine constitute a hypertree. We show that our problem can be solved efficiently by a tree coloring technique. In Section 4 we generalize this approach to the case when besides the previous tasks there are $O(1)$ additional multiprocessor tasks.

In the second part of the paper we consider the case where any pair of tasks can share an arbitrary number of processors. In particular, we show that the scheduling problem is NP-hard even if the system consists of 2 machines. Since the proof is rather complicated, we postpone it to Section 5 of the paper. The paper is concluded with a description of how the procedure developed in Section 4 can be used to meet the most common criteria of scheduling.

2. Mathematical model

Classical scheduling theory provides various criteria of scheduling like C_{max}, L_{max}, T_{max}, F, $\Sigma w_j C_j$ and many others. All of them can be reduced to the cost list scheduling model which is presented in this section. Let $J = \{J_1, J_2, \ldots, J_m\}$ be a set of tasks which can be executed on a set of processors $M = \{M_1, M_2, \ldots, M_n\}$. All processors are distinct, and they can execute only one specified task at a time. Each task J_i requires the simultaneous use of a nonempty set $fix_i \subseteq M$ of processors for its execution. All tasks are independent, nonpreemptable, and of the same length. For the sake of simplicity we may assume that execution time of J_i, i.e. $p_i = 1$, for each $i = 1, \ldots, m$. Every processor can work on not more than one task at the same time. Time is divided into unit length slots numbered with successive integers. Tasks have availability constraints prespecified by lists $L(J_i) \subseteq \mathbb{N}$ (where $\mathbb{N}$ is the set of nonnegative integers) of available time slots in which J_i can be executed. With each task we associate a function $f_{J_i}(x)$ which assigns to each $x \in L(J_i)$ the cost of executing J_i if it is executed in time slot x. Our aim is to find a schedule with minimum total cost.

Our problem can be described using three-field notation as $P|win, p_j = 1, fix_j|crit$, where win stands for the fact that availability constraints are imposed on tasks. By the word *crit* we mean any of the criteria commonly used in classical scheduling theory.

The cost list scheduling problem can be modeled as the cost list edge-coloring of a hypergraph. Let $H = (V, E)$ be a *hypergraph*. $V = V(H)$ is the set of *vertices* and $E = E(H)$ is a multiset of nonempty subsets of V called *edges*. A d-edge e is an edge that contains exactly d vertices. $\Psi(e) = |e| = d$ denotes the *dimension* of edge e and $\Psi(H) = \max_{e \in E} \Psi(e)$ is the *dimension* of hypergraph H. The *degree* $\Delta(v)$ of a vertex $v \in V$ is the number of edges in which v occurs. $\Delta(H) = \max_{v \in V} \Delta(v)$ is the *degree* of H. The *neighborhood* $N(e)$ of edge e is the set of all edges in H that share at least one vertex with e. $N(H)$ is the cardinality of the maximal neighborhood of an edge in H.

A *proper edge-coloring* of hypergraph H with k colors is a function $c : E(H) \to \{1, \ldots, k\}$ such that no two edges which share a vertex have the same color (number). A *proper list edge-coloring* of a hypergraph requires the function c to satisfy an additional condition $c(e) \in L(e)$ for each $e \in E$, where $L(e)$ is the list of colors available to edge e. Let us consider a function $f_e : L(e) \to \mathbb{N}$ for $e \in E$. If for all e, all elements $x \in L(e)$ are assigned cost $f_e(x)$ and the aim of the coloring is to minimize the total cost, then the problem becomes a *cost list edge-coloring* (CLEC) of the hypergraph.

There is a one-to-one correspondence between cost list edge-coloring of a hypergraph and the cost list scheduling model presented in this section. Vertices of a hypergraph correspond to processors, edges to jobs and colors to time slots. A hypergraph created for an instance of a scheduling problem is called a *scheduling hypergraph*. From now on the terms 'scheduling' and 'edge coloring' will be used interchangeably. In Fig. 1 we give an example of an instance of the cost list scheduling problem, and a solution to it with total value 9. We assume in the example that $f_e(x) = x$ for all e and x.

The hypergraph coloring problem is in the general case NP-hard. For this reason we consider some sparse and specific hypergraphs in this paper. A *hyperforest* H is a hypergraph such that there exists a spanning tree T (simple graph) for which each edge of H induces a connected subtree in T. A connected hyperforest is called a *hypertree*. We say that a hypergraph is *linear* if no two edges share more then one vertex.