Theoretical Computer Science 560 (2014) 190–206

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Online optimization of busy time on parallel machines ☆

Mordechai Shalom [a], Ariella Voloshin [b], Prudence W.H. Wong [c,*], Fencol C.C. Yung [c], Shmuel Zaks [b]

[a] TelHai College, Upper Galilee, 12210, Israel
[b] Department of Computer Science, Technion, Haifa, Israel
[c] Department of Computer Science, University of Liverpool, Liverpool, UK

ARTICLE INFO

Article history:
Received 6 November 2012
Accepted 15 July 2014
Available online 22 July 2014

Keywords:
Online algorithms
Competitive analysis
Busy time minimization
Parallel machines scheduling

ABSTRACT

We consider the following online scheduling problem in which the input consists of n jobs to be scheduled on identical machines of bounded capacity g (the maximum number of jobs that can be processed simultaneously on a single machine). Each job is associated with a release time and a completion time between which it is supposed to be processed. When a job is released, the online algorithm has to make decision without changing it afterwards. A machine is said to be busy at a certain time if there is at least one job processing on the machine at that time. We consider two versions of the problem. In the minimization version, the goal is to minimize the total busy time of machines used to schedule all jobs. In the resource allocation maximization version, the goal is to maximize the number of jobs that are scheduled under a budget constraint given in terms of busy time. This is the first study on online algorithms for these problems. We show a rather large lower bound on the competitive ratio for general instances. This motivates us to consider special families of input instances for which we show constant competitive algorithms. Our study has applications in power-aware scheduling, cloud computing and optimizing switching cost of optical networks.

1. Introduction

The problem. Job scheduling on parallel machines has been widely studied (see, e.g., the surveys in [5,21]). In particular, much attention was given to *interval scheduling* [20], where jobs are given as intervals on the real line, each representing the time interval during which a job should be processed; each job has to be processed on some machine, and it is commonly assumed that a machine can process a *single* job at any given time.

In this paper we consider online interval scheduling with *bounded parallelism*. Formally, the input is a set $\mathcal{J}$ of n jobs. Each job, $J \in \mathcal{J}$, is associated with an interval $[r_J, c_J]$ during which it should be processed. We are also given the parallelism parameter $g \geq 1$, which is the maximum number of jobs that can be processed simultaneously by a single machine. At any given time t a machine M_i is said to be busy if there is at least one job J scheduled on it such that $t \in [r_J, c_J]$, otherwise

☆ This work was supported in part by the Israel Science Foundation grant No. 1249/08 and British Council Grant UKTELHAI09. A preliminary version of the paper appeared in Proceedings of the 9th Annual Conference on Theory and Applications of Models of Computation (TAMC), Beijing, 16–21 May 2012, pp. 448–460.

* Corresponding author.

E-mail addresses: cmshalom@telhai.ac.il (M. Shalom), variella@cs.technion.ac.il (A. Voloshin), pwong@liverpool.ac.uk (P.W.H. Wong), ccyung@graduate.hku.hk (F.C.C. Yung), zaks@cs.technion.ac.il (S. Zaks).

http://dx.doi.org/10.1016/j.tcs.2014.07.017
0304-3975/© 2014 Elsevier B.V. All rights reserved.

M_i is said to be idle at time t. We call the time period in which a machine M_i is busy its *busy period*. In this work we study two optimization problems MinBusy and MaxThroughput. In MinBusy we focus on minimizing the total busy time over all machines. Note that a solution minimizing the total busy time may not be optimal in terms of the number of machines used. In Section 5.3 we discuss the relation between MinBusy and minimization of the number of machines. In MaxThroughput, the resource allocation version of the problem, we are given a budget T of total machine busy time and the objective is to maximize the number of scheduled jobs under this constraint.

The input to our scheduling problems can be viewed as an *interval graph*, which is the intersection graph of a set of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intersecting intervals. In our setting, each vertex corresponds to a job, and there is an edge between two jobs whose processing times overlap.

Applications. Our scheduling problems can be directly interpreted as **power-aware scheduling** problems in cluster systems. These problems focus on minimizing the power consumption of a set of machines (see, e.g., [31] and references therein) measured by the amount of time the machines are switched on and processing, i.e. the total busy time. It is common that a machine has a bound on the number of jobs that can be processed at any given time.

Another application of the studied problems comes from **cloud computing** (see, e.g., [27,30]). Commercial cloud computing provides computing resources with specified computing units. Clients with computation tasks require certain computing units of computing resources over a period of time. Clients are charged in a way proportional to the total amount of computing time of the computing resource. The clients would like to minimize the charges they have to pay (i.e. minimize the amount of computing time used) or maximize the amount of tasks they can compute with a budget on the charge. This is in analogy to our minimization and maximization problems, respectively.

Our study is also motivated by problems in **optical network design** (see, e.g., [9,11,12]). Optical wavelength-division multiplexing (WDM) is the leading technology that enables us to deal with the enormous growth of traffic in communication networks, like the Internet. In an optical network, communication between nodes is realized by *lightpaths*, each of which is assigned a certain color. As the energy of the signal along a lightpath decreases, *regenerators* are needed in order to regenerate the signal, thus the associated hardware cost is proportional to the length of the lightpaths. Furthermore, connections can be "groomed" so that a regenerator placed at some node v and operating at some color λ can be shared by at most g connections colored λ and traversing v. This is known as *traffic grooming*. The regenerator optimization problem on the path topology is in analogy to our scheduling problem in the sense that the regenerator cost measured in terms of length of lightpaths corresponds to the busy time while grooming corresponds to the machine capacity.

In the above three applications, it is natural to consider online version of the problem where jobs arrive at arbitrary time and decisions have to be made straightaway (see e.g., [25,27,30]).

Related work. Some of the earlier work on interval scheduling considers the problem of scheduling a feasible subset of jobs with maximum total weight, i.e., a *maximum weight independent set* (see, e.g., [2] and surveys in [17,18]). There is wide literature on *real-time scheduling*, where each job has to be processed on some machine during a time interval between its release time and due date. There are also studies on real-time scheduling, where each machine has some capacity and each job has a demand of a certain machine capacity; however, to the best of our knowledge, all of this prior work (see, e.g., [2,6,8,28]) refers to different flavor of the model than the one presented here. Interval scheduling has been studied in the context of online algorithms and competitive analysis [19,22]. It is also common to consider both minimization and maximization versions of the same scheduling problem, see e.g., [3] but in that model the machines have unit capacity.

Our study also relates to *batch scheduling* of conflicting jobs, where the conflicts are given as an interval graph. In p-batch scheduling model (see, e.g., Chapter 8 in [5]) a set of jobs can be processed jointly. All the jobs in the batch start simultaneously, and the completion time of a batch is the last completion time of any job in the batch. (For known results on batch scheduling, see, e.g., [5].) Our scheduling problem differs from batch scheduling in several aspects. In our problems, each machine can process g jobs simultaneously, for some $g \geq 1$, the jobs need not be partitioned to batches, i.e., each job can start at different time. Also, while in known batch scheduling problems the set of machines is given, we assume that *any* number of machines can be used for the solution. Finally, while common measures in batch scheduling refer to the maximum completion time of a batch, or a function of the completion times of the jobs, we consider the total busy times of the machines.

Previous work on busy time scheduling. The complexity of MinBusy was studied in [32], which showed that the problem is NP-hard already for $g = 2$. The work [13] considered the problem where jobs are given as intervals on the line with unit demand. For this version of the problem it gives a 4-approximation algorithm for general inputs, and better bounds for some subclasses of inputs. In particular, 2-approximation algorithms were given for instances where no job interval is properly contained in another interval (called "proper" instance), and "clique" instances where any two job intervals intersect, i.e., the input forms a clique (see same approximation but different algorithm and analysis in [14]). The work [16] extends the results of [13], considering the case where each job has a different demand on machine capacity and possibly has some slack time. The work [26] improves upon [13] on some subclasses of inputs and initiates the study of MaxThroughput. A 6-approximation is proposed for clique instances, and a polynomial time algorithm is proposed for proper clique instances, i.e. instances that are both "clique" and "proper". These special instances have been considered in [16,26]. The study so far has been focused on the offline setting [13,14,16,26].