European Journal of Operational Research 000 (2017) 1–14

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Competitive algorithms for multistage online scheduling

Michael Hopf[a,*], Clemens Thielen[a], Oliver Wendt[b]

[a] Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, Kaiserslautern D-67663, Germany
[b] Department of Economics, University of Kaiserslautern, Erwin-Schrödinger-Str. 42, Kaiserslautern D-67663, Germany

ARTICLE INFO

Article history:
Received 17 June 2015
Accepted 30 December 2016
Available online xxx

Keywords:
Scheduling
Online optimization
Competitive analysis

ABSTRACT

We study an online flow shop scheduling problem where each job consists of several tasks that have to be completed in t different stages and the goal is to maximize the total weight of accepted jobs. The set of tasks of a job contains one task for each stage and each stage has a dedicated set of identical parallel machines corresponding to it that can only process tasks of this stage. In order to gain the weight (profit) associated with a job j, each of its tasks has to be executed between a task-specific release date and deadline subject to the constraint that all tasks of job j from stages $1, \dots, i-1$ have to be completed before the task of the ith stage can be started. In the online version, jobs arrive over time and all information about the tasks of a job becomes available at the release date of its first task. This model can be used to describe production processes in supply chains when customer orders arrive online.

Even the basic version of the offline problem with a single machine in each stage, unit weights, unit processing times, and fixed execution times for all tasks (i.e., deadline minus release date equals processing time) is APX-hard and we show that the approximation ratio of any polynomial-time approximation algorithm for this basic version of the problem must depend on the number t of stages.

For the online version of the basic problem, we provide a $(2t-1)$-competitive deterministic online algorithm and a matching lower bound. Moreover, we provide several (sometimes tight) upper and lower bounds on the competitive ratios of online algorithms for several generalizations of the basic problem involving different weights, arbitrary release dates and deadlines, different processing times of tasks, and several identical machines per stage.

1. Introduction

Scheduling is concerned with the allocation of jobs to scarce resources (machines). In revenue management, each job has a certain weight (or revenue) and the goal is to output a feasible subset of the jobs with maximum total weight. For each job j, the scheduler has to decide whether to accept the job (occupying a machine) or reject it (losing potential revenue). Additionally, the scheduler must decide to which machine each accepted job should be assigned and when it should be executed within the time interval between its release date r_j and deadline d_j.

The special case where the processing requirement p_j of a job j is equal to $d_j - r_j$ is known as *interval scheduling*. Here, the scheduler does not have to decide when to execute a job since each accepted job has to start directly at its release date and will end at its deadline.

The more general case $p_j \leq d_j - r_j$ is considered as *job scheduling*, where the difference $d_j - r_j$ is called *interval length*.

In this paper, we analyze a scheduling problem where each job j consists of several tasks and each of them has to be scheduled in a certain *stage*. There is a dedicated set of parallel machines available for each stage $i \in \{1, \dots, t\}$ and these machines can only process tasks of stage i (where each machine can process one task at a time). Each job j has one task T_j^i in each stage i and each task T_j^i has a specific release date r_j^i, processing time p_j^i and deadline d_j^i, where we assume that $d_j^{i-1} \leq r_j^i$ for $i = 2, \dots, t$. Each job j has a nonnegative weight w_j that is obtained if job j is accepted, in which case all tasks of job j have to be completed by their deadlines on the machines of the corresponding stages. The objective is to maximize the total weight of accepted jobs.

In the online version of the problem, jobs arrive over time and all tasks of a job become known at the release date of the first task of the job. Here, the scheduler is allowed to abort previously accepted jobs in order to accept jobs arriving later (which might have larger weight).

The problem is motivated by scheduling processes in which several individual operations have to be performed in sequence

* Corresponding author.
E-mail addresses: hopf@mathematik.uni-kl.de (M. Hopf), thielen@mathematik.uni-kl.de (C. Thielen), wendt@wiwi.uni-kl.de (O. Wendt).

http://dx.doi.org/10.1016/j.ejor.2016.12.047
0377-2217/

and profit is only obtained after a job has been processed in all stages. A specific example is scheduling emergency patients in a hospital. Patients arrive online and each patient either has to be scheduled immediately for several treatments (stages) on a clinical pathway (if the job is accepted) or has to be sent to another hospital (if the job is rejected). The goal of the scheduler is to accept as many patients as possible, which means that the jobs have unit weights.[1]

Another motivation is the map-reduce paradigm (Dean & Ghemawat, 2008), which is a standard programming model used in industry for processing and generating large data sets. The idea is to portion the input into map tasks that can be run on map machines in the first stage outputting key-value pairs. In the second stage, these pairs serve as input for the reduce machines. A more detailed description of the map-reduce paradigm from a scheduling perspective can be found in Moseley, Dasgupta, Kumar, and Sarlós (2011), where the authors consider a variant of the two-stage flexible flow shop problem motivated by the map-reduce paradigm. Later, Fotakis, Milis, Papadigenopoulos, Zampetakis, and Zois (2015) extended this model to three stages by introducing an additional shuffle stage.

Further possible applications of our model include production processes in supply chains, where each step of the production process corresponds to a stage and profit is only obtained when the final product is delivered to the customer. In this context, aborting a job may correspond to outsourcing its remaining tasks to external contractors.

1.1. Previous work

Some of our results focus on interval scheduling, i.e., the case where the processing time of each job/task is equal to its deadline minus its release date. Single-stage interval scheduling problems are well-studied in literature. The single-stage offline version of our problem is known to be efficiently solvable in polynomial time, even in the case of arbitrary weights (Arkin & Silverberg, 1987; Bouzina & Emmons, 1996). For unit weights, even the online problem with a single stage can be solved optimally by a simple greedy algorithm (Carlisle & Lloyd, 1995; Faigle & Nawjin, 1995). When different weights are allowed, however, this problem does not admit any competitive online algorithms if no further restrictions are imposed (Canetti & Irani, 1998; Woeginger, 1994).

A model similar to ours was considered in Bar-Yehuda, Halldórsson, Naor, Shachnai, and Shapira (2006). Here, the authors provide (offline) approximation algorithms for a single-stage model in which each job (or *t-interval*) consists of a union of at most t half-open intervals. However, there is only one machine that processes all the intervals whereas, in our model, we have a separate set of machines in each stage, which implies that tasks of different stages cannot interfere with each other. In Bachmann, Halldórsson, and Shachnai (2013), the authors consider the online selection of t-intervals, which do not necessarily arrive in order of their left endpoints, and provide upper and lower bounds on the competitive ratio of randomized online algorithms.

Other researchers consider similar problems. Bafna, Narayanan, and Ravi (1995) and Berman and DasGupta (2002) analyze the problem of scheduling nonoverlapping local alignments, which corresponds to our basic problem in the offline case. Their work is motivated by applications in computational molecular biology. In Bafna et al. (1995), the authors analyze the *IR problem*, which consists of choosing a maximum independent subset of axis-parallel rectangles, where two rectangles are independent if both of their projections on the axes do not intersect. They show NP-completeness of the problem even for unit weights and provide a tight analysis of a natural local improvement heuristic for general weights. In Berman and DasGupta (2002), the authors provide a 3-approximation for the two-dimensional weighted version that runs in $\mathcal{O}(n \log n)$ time. In Chlebík and Chlebíková (2005), the authors provide inapproximability results for the independent set problem in d-box graphs, i.e., intersection graphs of axis-parallel rectangles in $\mathbb{R}^d$. Here, intersections between rectangles are defined by the intersection of sets in $\mathbb{R}^d$ rather than by considering projections to the axes.

The offline case of our problem is a special case of finding a maximum weight independent set in a d-claw free graph. More specifically, scheduling jobs in t stages can be thought of finding a maximum weight independent set in the corresponding intersection graph, which is $(2t+1)$-claw free. Approximation algorithms for the maximum weight independent set problem in d-claw free graphs can be found in Chandra and Halldórsson (1999). A $d/2$-approximation algorithm is given in Berman (2000). This corresponds to a $(t+\frac{1}{2})$-approximation algorithm for our scheduling problem. In Halldórsson (1995), the author presents an approximation algorithm for the unweighted maximum independent set problem in d-claw free graphs based on local improvement search and achieves an approximation ratio of $\frac{d-1}{2}+\epsilon$. This yields a $(t+\epsilon)$-approximation algorithm for the basic version of our scheduling problem.

There is also some work that directly faces the online independent set problem where nodes are presented one after each other and, at each step, the algorithm either has to accept or decline the incoming node. However, it is easy to see that an online algorithm cannot have a better competitive ratio than n for this problem, where n denotes the number of nodes in the graph. Therefore, researchers consider models in which the online algorithm has more power or additional information about the arriving nodes. For example, in Halldórsson, Iwama, Miyazaki, and Taketomi (2002), the authors consider settings where the online algorithm can maintain several solutions and in Göbel, Hoefer, Kesselheim, Schleiden, and Vöcking (2014), the online algorithm is given additional information about the graph or the distribution of the node weights a priori.

To the best of our knowledge, the online case of our scheduling setting with the objective of maximizing the total weight of accepted jobs has not been studied so far.

1.2. Our contribution

We present online algorithms for several cases of the problem, sometimes obtaining tight upper and lower bounds on the competitive ratio achievable by any deterministic online algorithm. Our competitiveness results are summarized in Table 1.

The basic problem considered in Section 3 is a restricted version with unit weights, unit processing times, unit interval lengths, and a single machine in each stage. We obtain upper and lower bounds on the competitiveness of online algorithms for this setting as well as for the generalizations to arbitrary weights (Section 4.1) and unit weights but arbitrary interval lengths (Section 4.2). Afterward, we consider the combination of different weights and arbitrary interval lengths (Section 4.3) as well as a general model combining different weights, arbitrary interval lengths, and arbitrary processing times (Section 4.4). In the section on parallel machines (Section 4.5), we analyze the problem with several identical parallel machines in each stage and show how several results from the single-machine case can be extended.

For the offline problem, we show that even the offline version of the basic problem considered in Section 3 is APX-hard. Moreover, we show that (unless P = NP) there does not exist a

[1] Note that aborting jobs is not feasible in this particular application. However, the online algorithms we present for the case of unit weights never abort jobs, so they apply to settings without the possibility to abort jobs as well.

Please cite this article as: M. Hopf et al., Competitive algorithms for multistage online scheduling, European Journal of Operational Research (2017), http://dx.doi.org/10.1016/j.ejor.2016.12.047