Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Online algorithms with advice: The tape model[☆], [☆☆]

Hans-Joachim Böckenhauer[a], Dennis Komm[a,*], Rastislav Královič[b], Richard Královič[c], Tobias Mömke[d]

[a] *Department of Computer Science, ETH Zurich, Switzerland*
[b] *Department of Computer Science, Comenius University, Slovakia*
[c] *Google Inc., Switzerland*
[d] *Department of Computer Science, Saarland University, Germany*

ARTICLE INFO

Article history:
Received 21 July 2010
Received in revised form 27 September 2016
Available online xxxx

Keywords:
Advice complexity
Online computation
Competitive analysis
Paging
Disjoint path allocation
Job shop scheduling

ABSTRACT

We investigate to what extent the solution quality of online algorithms can be improved when supplying them with information about the input. To this end, we refine the recently introduced notion of *advice complexity* where the algorithm has access to an advice string that is communicated by an oracle with access to the complete input. The advice complexity is the length of this communication. We study the connections between advice complexity and both determinism and randomization using the paging problem, job shop scheduling, and a routing problem as sample problems. Our results for these problems show that very small advice (only three bits in the case of paging) suffices to significantly improve over the best deterministic algorithms. To be as good as randomized algorithms, a moderate number of advice bits is sufficient, but it varies with the specific problem considered. However, to obtain optimality, usually very large advice is necessary.

1. Introduction

Many problems such as paging and problems from the area of routing and scheduling work in so-called online environments and their algorithmic formulation and analysis demand a model in which an algorithm that deals with such a problem knows only a prefix of its input at any specific point during runtime. These problems are called *online problems* and the respective algorithms are called *online algorithms*. In such an online setting, an online algorithm Alg has a huge disadvantage compared to offline algorithms (i.e., algorithms that know the whole input at the beginning of their computation) since Alg has to make decisions at any time step i without knowing the further chunks of the input. As Alg has to produce a part of the final output in every step, it cannot revoke decisions it has already made. These decisions can only be made by merely taking input chunks from time steps 1 to i into account and, in the case of randomized algorithms, applying some randomization.

The output quality of such an online algorithm is often studied by the well-established concept of *competitive analysis* introduced by Sleator and Tarjan [28]; note that Karlin et al. [21] were the first to use the term “competitive” in this setting.

[☆] This work was partially supported by ETH grant TH 18 07-3, and APVV grant 0433-06.
[☆☆] An extended abstract of this paper has been presented at ISAAC 2009 [8].
[*] Corresponding author.
E-mail addresses: hjb@inf.ethz.ch (H.-J. Böckenhauer), dennis.komm@inf.ethz.ch (D. Komm), kralovic@dcs.fmph.uniba.sk (R. Královič), richard.kralovic@inf.ethz.ch (R. Královič), moemke@cs.uni-saarland.de (T. Mömke).

http://dx.doi.org/10.1016/j.ic.2017.03.001
0890-5401/© 2017 Elsevier Inc. All rights reserved.

Informally speaking, the output quality of an online algorithm Alg is measured by comparing it to that of an optimal offline algorithm. For an online problem P, let $\text{Opt}(I)$ denote an optimal offline solution for an input I of P. Suppose P is an online minimization problem. Then, for some $r \geq 1$, Alg is called *r-competitive* if there exists some constant $\alpha \geq 0$ such that, for any input sequence I, $\text{cost}(\text{Alg}(I)) \leq r \cdot \text{cost}(\text{Opt}(I)) + \alpha$. Conversely, if P is an online maximization problem, we demand $\text{gain}(\text{Opt}(I)) \leq r \cdot \text{gain}(\text{Alg}(I)) + \alpha$. The *competitive ratio* of Alg is given by the infimum over all r for which Alg is r-competitive.

As a tool that has often proved successful, randomness is often employed in the design of online algorithms. The computations (sometimes also called *decisions*) of a randomized online algorithm Rand depend on a sequence of random bits often viewed as the content of a random tape accessible by Rand. For a fixed content of the random tape, Rand then behaves deterministically and achieves a certain competitive ratio. The expected value of the competitive ratio over all possible contents of the random tape is commonly used to measure the quality of a randomized online algorithm.

On the downside, it seems rather unrealistic to assume no knowledge about the input at all. Indeed, in many applications, properties of typical inputs may be known in advance. Competitive analysis has therefore often been criticized for not being sufficiently realistic [4]. In this paper, we try to quantify the information that can help to obtain high-quality solutions. In particular, lower bounds are of interest as we do not make any assumption on the nature of the information supplied. This way, we generalize different models of refined competitive analysis such as *semi-online algorithms* or *lookahead*.

Hence, we are interested not only in comparing the output quality of Alg to that of an optimal offline algorithm Opt, but we want to investigate what amount of information Alg really lacks. Surprisingly (and as already discussed [13]), there are problems where only one straightforward piece of information (i.e., one bit) is needed to allow Alg to be as good as Opt, e.g., the ski rental problem [4,13]. Clearly, this does not hold in general and thus we are interested in a formal framework which allows us to classify online problems according to how much information about the future input parts is needed for solving them optimally or with a specific competitive ratio. For the latter, we are especially interested in the best competitive ratios that are known to be achievable by deterministic or random strategies without any advice. One way of measuring the amount of information needed for an online algorithm to be optimal is called *advice complexity* and was proposed by Dobrev et al. [13]. This model of advice complexity can informally be viewed as a cooperation of a deterministic online algorithm Alg and an *oracle* that may passively communicate with Alg.

In this paper, the oracle, which has unlimited computational power, sees the whole input of Alg in advance and writes binary information onto an *advice tape* before Alg reads any input. Then, Alg can access the bits from the advice tape in a sequential manner, just as a randomized algorithm would use its random tape. The *advice complexity* of Alg on an input I is now defined as the number of advice bits Alg reads while processing this input. As usual, we consider the advice complexity as a function of the input size n by taking the maximum value over all inputs of length at most n.

Besides asking for the amount of advice that is necessary to compute an optimal solution, we also deal with the question whether some small advice might help to significantly reduce the competitive ratio. We analyze the advice complexity of three different online problems, namely the well-known paging problem, a job shop scheduling problem, and the problem of routing communication requests in a line-topology network. Using these sample problems, we investigate the relation between advice complexity and the competitive ratio of deterministic as well as randomized online algorithms.

1.1. *Contribution and organization of the paper*

In the original model of Dobrev et al. [13], the oracle sends information to the online algorithm one message per time step. There are two modes of operation. In the first one, the oracle oversees the algorithm's work and sends the advice without being asked. In the second one, the algorithm needs to ask for advice explicitly. The criticism has been made that this information is considered binary, although there is no explicit end marker of any message. This way, the length of a message also carries some information. Indeed, if the oracle is allowed to send an empty message, the algorithm can, e.g., decide between the three cases "empty message," "message is the bit 0," and "message is the bit 1." In the analysis, this is still treated as one bit of advice. In the model proposed in this paper, no additional information is revealed. One of the main contributions of this paper is therefore a refinement of the original model, which allows us to measure the information content of the problem in a clean way. Indeed, by our definition, the oracle has no possibility to explicitly indicate the end of the advice. It must provide as much advice as asked by the algorithm. This eliminates the ability of the oracle to encode some information into the length of the advice string, as was the case in [13]. As a result, our model is cleaner and more consistent with other complexity measures, e.g., the number of random bits required for randomized algorithms.

We start with the formal definition of our model and some general observations in Section 2. Furthermore, we show how to compress an advice string under certain conditions.

As a first example, we consider the well-known *paging problem* in Section 3 and show that very little advice is needed to significantly improve over every deterministic online algorithm. Recall that, for the paging problem, we are given a fast memory, called *buffer*, with a capacity of k data units, called *pages*, and a large but slow secondary memory. An input consists of a sequence of n page requests. Whenever a requested page is not in the buffer, it has to be fetched from the secondary memory. This situation is called a *page fault*, and the goal is to minimize the number of page faults. It is well-known that any deterministic online algorithm for the paging problem cannot achieve a better competitive ratio than k, and even randomized algorithms can at best achieve a ratio of H_k, where H_k is the kth harmonic number. As for our