Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Improved analysis of the online set cover problem with advice[☆,☆☆]

Stefan Dobrev[a], Jeff Edmonds[b], Dennis Komm[c,*], Rastislav Královič[e], Richard Královič[f], Sacha Krug[c,1], Tobias Mömke[d]

[a] *Slovak Academy of Sciences, Slovakia*
[b] *Department of Electrical Engineering and Computer Science, York University, Canada*
[c] *Department of Computer Science, ETH Zurich, Switzerland*
[d] *Department of Computer Science, Saarland University, Germany*
[e] *Department of Computer Science, Comenius University, Slovakia*
[f] *Google Inc., Switzerland*

A R T I C L E   I N F O

Article history:
Received 12 February 2016
Received in revised form 28 February 2017
Accepted 30 May 2017
Available online xxxx
Communicated by Giuseppe F. F. Italiano

Keywords:
Online computation
Advice complexity
Set cover problem

A B S T R A C T

We study the advice complexity of an online version of the set cover problem. The goal is to quantify the information that online algorithms for this problem need to be supplied with to compute high-quality solutions and to overcome the drawback of not knowing future requests. This concept was successfully applied to many prominent online problems in the past while trying to capture the essence of "what makes an online problem hard." The online set cover problem was introduced by Alon et al. (2009) [2]: for a ground set of size n and a set family of m subsets of the ground set, we obtain bounds in both n and m. We show that a linear number (with respect to both n and m) of advice bits is both sufficient and necessary to perform optimally. Furthermore, we prove that $\mathcal{O}((n \log c)/c)$ bits are sufficient to design a c-competitive online algorithm, and this bound is tight up to a factor of $\mathcal{O}(\log c)$. We further give upper and lower bounds for achieving c-competitiveness with respect to m. Finally, we analyze the advice complexity of the problem with respect to some natural parameters, i.e., measurable properties of the inputs.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction and preliminaries

The idea of online algorithms is both a natural concept in practical applications and of great theoretical interest. In this model, the algorithm designer faces the problem that an algorithm Alg has to read the input instance of some problem piecewise and has to create parts of the output before knowing the whole input string. Furthermore, Alg is not allowed to revoke any pieces of submitted output. Online scenarios and online algorithms have been studied extensively; for an overview, we refer the reader to the literature [1,9,16,19]. Similar to the concept of the approximation ratio of algorithms

[☆] Some of the results presented in this paper were already published in an extended abstract [22].
[☆☆] This work was partially supported by grants BL511/10-1, MO2889/1-1 of Deutsche Forschungsgemeinschaft, SNF grant 200021-146372, VEGA 2/0136/12, and VEGA 1/0671/11.
[*] Corresponding author.
E-mail addresses: stefan.dobrev@savba.sk (S. Dobrev), jeff@cse.yorku.ca (J. Edmonds), dennis.komm@inf.ethz.ch (D. Komm), kralovic@dcs.fmph.uniba.sk (R. Královič), richard.kralovic@dcs.fmph.uniba.sk (R. Královič), moemke@cs.uni-saarland.de (T. Mömke).
[1] Deceased 20 November 2015.

http://dx.doi.org/10.1016/j.tcs.2017.05.029
0304-3975/© 2017 Elsevier B.V. All rights reserved.

dealing with (hard) offline problems, *competitive analysis* (introduced in 1985 by Sleator and Tarjan [26]) is usually used to measure the quality of online algorithms (ignoring the time complexity of the algorithm). Let Alg be an online algorithm for an online minimization problem; Alg is called *c-competitive* if its cost is at most c times as large as that of an optimal solution for every instance of the problem. The *competitive ratio* of Alg is the infimum over all c for which Alg is c-competitive. In this context, we think of the optimal solution as being computed by an *offline* algorithm Opt that sees the whole input in advance, which is obviously a huge advantage compared to the knowledge Alg possesses.

Clearly, we may see the competitive ratio as a value that tells us what we forfeit for not knowing the future. When studying the *advice complexity* of an online problem, we are interested in getting a deeper understanding of *what* it is we pay for. A straightforward answer to this question is “the remainder of the instance,” but we want to measure the essence of what makes the problem hard with respect to online computation. Conversely, from an adversary’s point of view, we aim to investigate the amount of power lost due to leakage of information to the algorithm. The main motivation is to quantify the amount of information that helps to achieve some specific solution quality. Since we do not impose any restrictions on the nature of the information, our lower bounds are valid for any kind of information; thus they generalize concepts such as lookahead or semi-online algorithms. What we get are statements of the sort “whenever less than x bits of information are supplied, the online algorithm will always be worse than $c(x)$-competitive.”

In the model used throughout this paper, we consider online algorithms that are able to receive extra information from some *advice tape* that is thought of as being created by an oracle that sees the whole input in advance. In every time step, an online algorithm Alg may sequentially read some piece of information from this tape together with the input. More formally, we are dealing with the following model.

Definition 1 *(Online algorithm with advice).* Let $I = (x_1, \dots, x_n)$ be an input sequence for some online problem. An *online algorithm* Alg *with advice* computes the output sequence $\textsc{Alg}^{\phi}(I) = (y_1, \dots, y_n)$ such that y_i is computed from $\phi, x_1, \dots, x_i$, where ϕ is the content of the advice tape, i.e., an infinite binary sequence. Alg is *c-competitive with advice complexity* $b(n)$ if there is a constant α such that, for every n and for each input sequence I of length at most n, there is some ϕ such that $\mathrm{cost}(\textsc{Alg}^{\phi}(I)) \le c \cdot \mathrm{cost}(\textsc{Opt}(I)) + \alpha$ and at most the first $b(n)$ bits of ϕ have been accessed during the computation of $\textsc{Alg}^{\phi}(I)$. If $\alpha = 0$, Alg is called *strictly c-competitive*. An algorithm is *optimal* if it is strictly 1-competitive.

Advice complexity of online algorithms was introduced by Dobrev et al. [13,14] in 2008 and since then revised versions [7,15,17] have been used to study various online problems, e.g., the paging problem and the disjoint path allocation problem [7], the job shop scheduling problem [7,21], metrical task systems [15], the k-server problem [5,15], and the knapsack problem [6]. There are interesting connections between computing with advice and randomization [5,21,23]. For a detailed survey on this topic, we refer to Hromkovič et al. [17], Boyar et al. [10], and Komm [20]. The study of advice has an interesting connection to recent developments in the field of dynamic data structures, where the usage of advice bits was proposed as a lower bound technique [25,11].

As usual, in order to design hard instances for the online algorithms studied, we consider an adversary that constructs the input in a malicious way [8,9,16,19]. As mentioned, we add another player to the game that is classically played between an online algorithm Alg and the adversary; an oracle that collaborates with Alg by giving help using the advice tape. Note that, similar to randomized online algorithms, we can think of an online algorithm Alg with advice as an algorithm that chooses, depending on the advice it is given, from a set of deterministic strategies denoted by Strat(Alg). Another view we might impose on *computations with advice* is that an optimal random choice is supplied by the oracle for every input.

Throughout this paper, log denotes the logarithm with base 2, and ln is the logarithm with base $\mathrm{e} = 2.718\dots$; by $\exp(x) = \mathrm{e}^x$ we denote the natural exponential function. To bound binomial coefficients, we will sometimes use [12] that

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{n\mathrm{e}}{k}\right)^k . \tag{1}$$

In addition, we make use of Sperner’s Theorem [27], which states that

$$|\mathcal{S}| \le \binom{n}{\lfloor n/2 \rfloor} , \tag{2}$$

for any subset-free set family $\mathcal{S}$ over a ground set with n elements. Denoting the *binary entropy function* by $\mathcal{H}_2(x) := -x \log x - (1-x)\log(1-x)$, for any $n, k \in \mathbb{N}$, we have

$$\binom{n}{k} \le 2^{n \cdot \mathcal{H}_2(k/n)} , \tag{3}$$

which gives us another means to bound the binomial coefficient [24].

We are now ready to define the online version of the set cover problem, which was introduced by Alon et al. [2].

Definition 2 *(The online set cover problem).* Given a *ground set* $X = \{1, \dots, n\}$, a set of *requests* $X' \subseteq X$, and a set family $\mathcal{S} \subseteq \mathcal{P}(X)$ of size m, without loss of generality $\emptyset \notin \mathcal{S}$, a feasible solution for the set cover problem is any subset $\{S_1, \dots, S_k\}$