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], Tobias Mömke[d]

[a] *Slovak Academy of Sciences, Slovakia*
[b] *Department of Computer Science and Engineering, 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*

Abstract

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 input 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. [SIAMJ 39(2), 2009]: 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 bound 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.

Keywords: Online computation, advice complexity, set cover problem

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 standard literature [1, 9, 16, 19]. Similar to the concept of the approximation ratio of algorithms 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 run time 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

☆Some of the results presented in this paper were already published in an extended abstract [22].
☆☆This work was partially supported by grant BL511/10-1 of Deutsche Forschungsgemeinschaft, SNF grant 200021-146372, VEGA 2/0136/12, and VEGA 1/0671/11.

Email addresses: stefan.dobrev@savba.sk (Stefan Dobrev), jeff@cse.yorku.ca (Jeff Edmonds), dennis.komm@inf.ethz.ch (Dennis Komm), kralovic@dcs.fmph.uniba.sk (Rastislav Královič), richard.kralovic@dcs.fmph.uniba.sk (Richard Královič), moemke@cs.uni-saarland.de (Tobias Mömke)
† Deceased 20 November 2015.

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 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 restriction 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 $I = (x_1, \ldots, x_n)$ be an input sequence for some online problem. An *online algorithm* ALG *with advice* computes the output sequence $\text{ALG}^\phi(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $\phi, x_1, \ldots, x_i$, 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 $\text{cost}(\text{ALG}^\phi(I)) \leq c \cdot \text{cost}(\text{OPT}(I)) + \alpha$ and at most the first $b(n)$ bits of ϕ have been accessed during the computation of $\text{ALG}^\phi(I)$. If $\alpha = 0$, ALG is called *strictly* c-competitive. An algorithm is *optimal* if it is strictly 1-competitive.

Advice complexity was introduced by Dobrev et al. [13, 14] in 2008 and since then revised versions [7, 15, 17] were used to study various online problems. For a detailed survey on this topic, we refer to Hromkovič et al. [17], Boyar et al. [10], and Komm [20]. Previous to this paper, the advice complexity of various problems was analyzed, 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]. 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 [11, 25].

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 $\mathrm{e} = 2.718\ldots$. By $\exp(x) = \mathrm{e}^x$ we denote the natural exponential function. To bound binomial coefficients,

we will sometimes use [12] that

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{ne}{k}\right)^k . \tag{1}$$

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

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

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

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

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* $X = \{1, \ldots, n\}$, 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 $\{S_1, \ldots, S_k\}$ of $\mathcal{S}$ such that $\bigcup_{i=1}^{k} S_i \supseteq X'$. The aim is to minimize k, i. e., to use as few sets as possible. X and $\mathcal{S}$ are known in advance, but the elements of X' arrive successively one by one in consecutive time steps. An online algorithm Alg is feasible for the online set cover problem if, immediately after each yet uncovered request j, it specifies a set $S_i \in \mathcal{S}$ such that $j \in S_i$.

Note that formally the term "online set cover problem" refers to a class of problems parametrized by two parameters n and m. It is crucial to note that $n = |X|$ is an upper bound on the actual input length. Therefore, an input for this problem does not have an arbitrary length, and since n is a parameter of the problem that is known in advance, we may say that every online algorithm is 1-competitive according to Definition 1 if we set the additive constant α to $n - 1$. In this paper, we therefore consider *strictly* competitive algorithms only when talking about both lower and upper bounds.

Alon et al. [2] designed a deterministic $\mathcal{O}(\log m \log n)$-competitive algorithm that even works for the weighted version of the set cover problem.[2] In the following, we give bounds on the advice complexity with respect to both the size n of the ground set X and the size m of the set family $\mathcal{S}$. Note that we may assume that $\mathcal{S}$ is subset-free (if not, sets that are contained in other sets may be removed from the input by a preprocessing step; since the sets are unweighted, it is never worse to prefer supersets). From this, it directly follows due to (2) that we have $|\mathcal{S}| \leq \binom{n}{\lfloor n/2 \rfloor}$.

Sometimes, when we need to communicate a non-negative number x in a self-delimiting way, we use a well-known technique that enables us to use at most $2\lceil \log \lceil \log x \rceil \rceil + \lceil \log x \rceil$ advice bits; see, e. g., Komm [20].

1.1. Organization and Contribution

We give almost tight upper and lower bounds that on the number of advice bits needed, which are linear in both the size of the ground set, i. e., $n = |X|$, and in the size m of $\mathcal{S}$. Section 2 considers the situation when an optimal solution is needed. Sections 3 and 4 then give upper and lower bounds on the number of advice bits necessary to obtain c-competitive solutions. These results can be summarized as follows.

[2] Here, every member of $\mathcal{S}$ has a non-negative weight, and the objective is to minimize the sum of the weights of all sets chosen.

Theorem 1. *With a ground set of size $n = |X|$ and a set family of size $m = |\mathcal{S}|$, the number b of advice bits needed by any online algorithm is bounded by*

- $\max\left(n - \log n, \frac{\log 3}{3}m\right) \leq b \leq \min(n, m)$ *to be optimal (Theorems 3 to 6) and*
- $\max\left(\frac{n}{\mathcal{O}(c)}, \frac{m}{c^{\mathcal{O}(c)}}\right) \leq b \leq \min\left(\frac{\log c}{c}n, \frac{\log m}{c}m\right)$ *to be c-competitive (Theorems 7, 9 and 10).*

Note that the fact that a linear number of advice is necessary to be optimal also follows from Böckenhauer et al. [4], which deals with near-optimal solutions. Here, we improve both the lower bound given in the preliminary version of the paper [22] (which is $(n - c^2 - 2c - 1)(\log((c+1)/c))/(c^2 + c)$) and the result from Böckenhauer et al. [4].

It is usually assumed that the oracle has unlimited computational power. The following observation considers the case when the oracle is more limited.

Observation 1. *If the oracle has to work in polynomial time, then our lower bounds still hold since the problem is only harder. For the upper bounds, note that the simple greedy algorithm is an $\ln n$-approximation. Hence, any upper bound on c-competitiveness implies a $(c \ln n)$-competitive online algorithm with the same number of advice bits and an oracle that works in polynomial time.* □

Section 5 considers the advice complexity of the problem with respect to a number of natural parameters, namely the largest set contained in $\mathcal{S}$, the size of an optimal solution, and the maximum number of times that an item occurs in the sets in $\mathcal{S}$. These results can be summarized as follows.

Theorem 2. *When ℓ is the size of the largest set from $\mathcal{S}$, k is the size of an optimal solution, and p is the maximum number of times that an item occurs in the sets in $\mathcal{S}$, the number b of advice bits needed by online algorithm is bounded by*

- $\ell \log(n/\ell) \leq b \leq k\ell \log n$ *to be optimal (Theorems 11 and 12),*
- $k \log p \leq b \leq k \log p$ *to be optimal, and*
- $1 \leq b \leq m - (c-1) \cdot m/(\log p + c)$ *to be c-competitive.*

Concluding remarks are given in Section 6.

2. Bounds to Obtain Optimality

First, let us analyze the *information complexity* of the problem, i. e., the advice complexity of optimal solutions [17]. In the following, we show that a number of advice bits linear in $|X|$ is sufficient to create an optimal output by a very easy argument.

Theorem 3. *There is an optimal online algorithm with advice that uses $n - 1$ advice bits.*

Proof. The proof is straightforward. The oracle simply writes the characteristic function of X' on the advice tape, i. e., a one at position i if and only if the ith element from X is contained in X'. However, the online algorithm knows one requested element from X before it has to take any action (and this is known by the oracle). Thus, if the first element requested is j, the jth position is simply skipped on the tape (not written down by the oracle). □

It is interesting to note that we are able to complement this result with an almost matching lower bound to show that a linear number of advice bits is also necessary to be optimal.

Theorem 4. *Every optimal online algorithm with advice needs to use at least* $n - (1/2)\log(n-1) - 3$ *advice bits.*

PROOF. Let ALG be an optimal online algorithm with advice, let n be even, and let $N(x) := \binom{x}{\lfloor x/2 \rfloor}$, for any $x \in \mathbb{N}$. Consider the set $\mathcal{S}$ that contains all subsets of X of size exactly $n/2$. Clearly, there are exactly $N(n)$ such sets. Now the adversary requests $n/2$ items, starting with one fixed item x_1 (after which ALG has to start with choosing a set from $\mathcal{S}$); one single set from $\mathcal{S}$ is sufficient to cover all requests. Since ALG knows that x_1 is included in the set it has to choose, there are $\binom{n-1}{n/2-1}$ remaining candidate sets. Observe that, since n is even, we have $n/2 - 1 = \lfloor (n-1)/2 \rfloor$. Consequently, there are exactly $N(n-1)$ sets left from which ALG has to select one.

Therefore, ALG has to distinguish between $N(n-1)$ different advices to distinguish this many sets and if it reads strictly less than $\log(N(n-1))$ bits, this merely enables ALG to choose among strictly less than $N(n-1)$ strategies, which is equivalent to choosing among this many deterministic algorithms. By the pigeonhole principle, there are two distinct sets S_1 and S_2 (i. e., inputs which both contain x_1, but differ in at least one element) for which ALG chooses the same deterministic strategy $A \in \mathrm{Strat}(\mathrm{ALG})$. Clearly, A cannot be optimal in both cases. Using Stirling's approximation, we get

$$\log(N(n-1)) = \log\left(\binom{n-1}{\lfloor (n-1)/2 \rfloor}\right) \geq \log\left(\frac{4^{(n-1)/2}}{2\sqrt{(n-1)\pi/2}}\right) > n - \frac{1}{2}\log(n-1) - 3\,,$$

which finishes the proof. □

Now we look at the online set cover problem from a different perspective, by bounding the advice in the number m of sets that are given. Encoding the characteristic vectors of the sets that are used by an optimal solution immediately gives the following result.

Theorem 5. *There is an optimal online algorithm with advice that uses* m *advice bits.* □

Again, this very naive approach is asymptotically the best we can hope for.

Theorem 6. *Every optimal online algorithm with advice needs to use at least* $m/3 \cdot \log 3$ *advice bits.*

PROOF. Let $n := ((q+1)/q)m$, for some $q \in \mathbb{N}^{\geq 2}$. The adversary chooses X such that there are $n/(q+1)$ items $y_1, \ldots, y_{n/(q+1)}$ and $(q/(q+1))n$ items $x_{i,j}$, $i \in \{1, \ldots, q\}$, $j \in \{1, \ldots, n/(q+1)\}$. After that, the adversary defines $\mathcal{S}$ to contain exactly the following sets: for every $k \in \{1, \ldots, n/(q+1)\}$, there are q sets $\{y_k, x_{1,k}\}, \ldots, \{y_k, x_{q,k}\}$.

Next, the adversary requests all items y_i; hence, since each request has to be satisfied, ALG has to fix $n/(q+1)$ sets to cover all items. After that, for every i, the adversary requests one of the q items $x_{1,i}, \ldots, x_{q,i}$. Note that any combination leads to a distinct optimal solution that consists of exactly m/q sets. This way, ALG has to correctly choose one of $q^{m/q}$ possible families as answers for the first m/q requests.

If, however, ALG uses less than $\log(q^{m/q})$ advice bits, then, by applying the pigeonhole principle, there must be two identical advices for two different sequences of correct answers. Thus, the adversary can choose the one not chosen by the algorithm. Consequently, a lower bound on the advice complexity is $\log(q^{m/q}) = m/q \cdot \log q$. The claim follows from observing that, for a fixed m, this expression is maximized for $q = 3$. □

Compare these results to those measured in n. On the one hand, in the proof of Theorem 6 we had $n = 4m/3$, and thus $m = 3n/4$ which yields a lower bound of $(n \log 3)/4$; i. e., it is worse than the lower bound provided by Theorem 4. On the other hand, in the proof of Theorem 4, we had $m = N(n)$ which means that it merely gives a logarithmic lower bound in m, whereas Theorem 6 gives a linear lower bound.

3. Upper Bounds to Obtain c-Competitiveness

The next question we are dealing with is how many bits of advice are necessary and sufficient to achieve c-competitiveness. This section gives two upper bounds. First, we introduce an online algorithm MaxCover that first tries to cover as many elements of X as possible together with every yet uncovered request. This is, however, not done too often (depending on the competitive ratio we want to achieve). For the part of X that is not covered by this procedure, the algorithm can compute an optimal solution using the advice.

Theorem 7. *There is a c-competitive online algorithm* MaxCover *with advice that uses $\mathcal{O}((n \log c)/c)$ advice bits.*

Proof. Consider a fixed n, and let k denote the number of sets taken by an optimal solution; let $d := \lfloor c \rfloor$. In order to be c-competitive, MaxCover can take $kd \leq kc$ sets to cover the requested elements. As sketched above, the high-level idea of the algorithm is as follows. At first, MaxCover reads k from the advice tape, which can be encoded using $\lceil \log n \rceil$ bits since $1 \leq k \leq n$. Afterwards, MaxCover identifies, without using any further advice or any information about the requested elements, $k(d-1)$ sets from $\mathcal{S}$ that cover a large part of X; we call these the "greedy sets." Additionally, the advice will encode all requested elements that are not covered by the greedy sets. Before processing any requests, MaxCover will use this advice to calculate an optimal solution to cover all such elements. By the definition of k, this is possible with at most k sets.

Afterwards, MaxCover processes the requests. Each requested element is covered by some greedy set whenever possible and by some set from the optimal solution for requested elements outside of the greedy sets otherwise. In that way, the total number of sets used by MaxCover is at most $k(d-1) + k = dk$. The idea is depicted in Figure 1.

First, we describe how the greedy sets are computed. For any $(X, \mathcal{S})$, let $\mathcal{S}_0 := \mathcal{S}$. Then we obtain a sequence $(\mathcal{S}_i)_{i \geq 0}$ as follows. Given a set family $\mathcal{S}_{i-1}$, we identify a set $T_i \in \mathcal{S}_{i-1}$ such that $|T_i|$ is maximal (ties are broken arbitrarily); T_i is the ith "shrunk greedy set." We obtain $\mathcal{S}_i$ from $\mathcal{S}_{i-1}$ by removing all elements of T_i from all sets $T' \in \mathcal{S}_{i-1}$. This way, we virtually shrink the instance such that it only contains elements that are not covered up to this point. Note that $\mathcal{S}_i = \{\emptyset\}$ for all $i \geq n$. We define

$$X_i := X \setminus \bigcup_{1 \leq j \leq i} T_j = \bigcup_{T' \in \mathcal{S}_i} T' \,,$$

i. e., X_i is the set of elements of X not covered after considering the first i shrunk greedy sets. By construction, each shrunk greedy T_i corresponds to some member of $\mathcal{S}$; all members of $\mathcal{S}$ that correspond to some shrunk greedy set T_i, for $1 \leq i \leq k(d-1)$, form the $k(d-1)$ greedy sets of MaxCover.

What needs to be done is to describe how to use the advice to specify which elements $W \subseteq X_{k(d-1)}$ are actually requested in the input X'. MaxCover is then able to use the advice to compute the optimal covering of these elements using at most k sets.

Let $Q_1, \ldots, Q_k$ be the sets of some fixed optimal solution and let $Q = \bigcup_{1 \leq i \leq k} Q_i$. Let $Q' := Q \cap X_{k(d-1)}$, and let a be the average size of the sets $T_1, \ldots, T_{k(d-1)}$, i. e.,

$$a = \sum_{i=1}^{k(d-1)} \frac{|T_i|}{k(d-1)} \,.$$

Then we can compute the number of elements of X not covered by greedy sets as

$$|X_{k(d-1)}| = n - k(d-1)a \,.$$

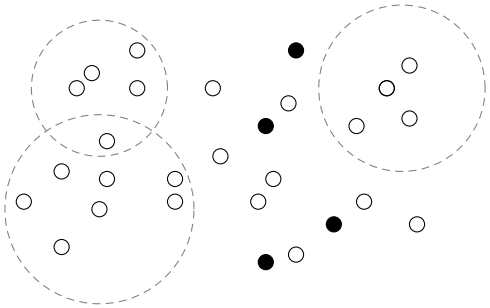

Figure 1: Example of how MaxCover processes an instance. The small circles represent the elements of X. The dashed circles represent the sets from $\mathcal{S}$ that correspond to the greedy sets T_i, where, $1 \leq i \leq k(d-1)$. The elements that remain and that are actually requested are filled.

Since the size $|T_i|$ decreases with i, we have $|T'| \leq a$, for all $T' \in \mathcal{S}_{k(d-1)}$, and we conclude that $|Q'| \leq ka$ since the sets $Q_1, \ldots, Q_k$ restricted to the corresponding sets in $\mathcal{S}_{k(d-1)}$ cover Q'. Then again, since Q' covers the actual requests W, we also have that $|W| \leq ka$.

The advice contains the encoding of $|W|$ using $\lceil \log n \rceil$ bits (which is possible since $|W| \leq n$). There are

$$\binom{|X_{k(d-1)}|}{|W|} = \binom{n - k(d-1)a}{|W|}$$

different subsets of $X_{k(d-1)}$ of size $|W|$; all of them are candidates for W. Since MaxCover knows $X_{k(d-1)}$ and $|W|$, it can compute all such candidates in some fixed ordering. The position of the actual W with respect to this ordering can then be encoded using another

$$\log\left(\binom{n - k(d-1)a}{|W|}\right) \tag{4}$$

advice bits.

We still have to bound (4) from above. Note that, by the definition of a and the T_is, it holds that $n - k(d-1)a \geq 0$ and therefore

$$ka \leq \frac{n}{d-1} \tag{5}$$

is always true.

Now we distinguish two cases depending on the size of the upper bound ka on the cardinality of W.

Case 1. First, suppose $ka > (n - k(d-1)a)/2$. Using (5) and $\binom{n}{l} \leq \binom{n}{\lfloor n/2 \rfloor}$, we immediately obtain

$$\begin{aligned}
\log\left(\binom{n - k(d-1)a}{|W|}\right) &< \log\left(\binom{2ka}{|W|}\right) \\
&\leq \log\left(\binom{2n/(d-1)}{|W|}\right) \\
&\leq \log\left(\binom{2n/(d-1)}{n/(d-1)}\right) \\
&\leq \frac{n}{d-1} \log 4 \,,
\end{aligned}$$

which is in $\mathcal{O}(n/d)$.

Case 2. Suppose $ka \leq (n - k(d-1)a)/2$. From $|W| \leq ka$ and (3), it follows that

$$\log\left(\binom{n-k(d-1)a}{|W|}\right) \leq \log\left(\binom{n-k(d-1)a}{ka}\right) \leq (n-k(d-1)a) \cdot \mathcal{H}_2\left(\frac{ka}{n-k(d-1)a}\right). \tag{6}$$

Note that $\mathcal{H}_2(x) \leq -2x \log x$, for $x \leq 1/2$, which holds due to the assumption of this case. Using this we get from (6) that

$$\log\left(\binom{n-k(d-1)a}{|W|}\right) \leq 2(n-k(d-1)a) \cdot \frac{ka}{n-k(d-1)a} \cdot \log\left(\frac{n-k(d-1)a}{ka}\right). \tag{7}$$

To further bound the expression, we define γ such that

$$ka = \frac{n}{\gamma(d-1)},$$

and thus

$$n - k(d-1)a = n\left(1 - \frac{1}{\gamma}\right).$$

Note that due to the assumptions of this case we have

$$\frac{n}{\gamma(d-1)} \leq \frac{n - n/\gamma}{2},$$

and hence $\gamma > 1$.

We use this together with the fact that, for any $x > 0$, we have $\log x/x \leq 0.54$, and finally obtain from (7) that

$$\begin{aligned}
\log\left(\binom{n-k(d-1)a}{|W|}\right) &\leq 2\left(1-\frac{1}{\gamma}\right)n \cdot \frac{ka}{n-k(d-1)a} \cdot \log\left(\frac{n-k(d-1)a}{ka}\right) \\
&\leq 2\left(1-\frac{1}{\gamma}\right)n \cdot \frac{\log((\gamma-1)(d-1))}{(\gamma-1)(d-1)} \\
&\leq 2\left(1-\frac{1}{\gamma}\right)n \cdot \left(\frac{\log(\gamma-1)}{(\gamma-1)(d-1)} + \frac{\log(d-1)}{(\gamma-1)(d-1)}\right) \\
&\leq 2\left(1-\frac{1}{\gamma}\right)n \cdot \left(\frac{0.54}{d-1} + \frac{\log(d-1)}{(\gamma-1)(d-1)}\right) \\
&\leq n\left(\frac{1.08}{d-1} + \left(\frac{\gamma-1}{\gamma}\right) \cdot \frac{2\log(d-1)}{(\gamma-1)(d-1)}\right) \\
&= n\left(\frac{1.08}{d-1} + \frac{2\log(d-1)}{\gamma(d-1)}\right) \\
&\leq n\left(\frac{1.08 + 2\log(d-1)}{d-1}\right),
\end{aligned}$$

which is in $\mathcal{O}((n \log d)/d)$.

To sum up, MaxCover needs to read $\lceil \log n \rceil$ bits to know k, another $\log n$ bits to know $|W|$, and at most $\mathcal{O}((n \log d)/d)$ additional bits to compute an optimal solution for the part of the input that is not covered by greedy sets. Since $(\log d)/d \leq (\log c)/(c-1)$, the algorithm is c-competitive using $\mathcal{O}((n \log c)/c)$ bits of advice. □

To obtain results with respect to the number m of sets in $\mathcal{S}$, we can make use of the previous results we gained with respect to n. More specifically, consider Theorem 7, which states that the online algorithm MaxCover with advice uses $\mathcal{O}((n \log c)/c)$ advice bits to be c-competitive. If $m \in \Theta(n)$, this implies that MaxCover uses $\mathcal{O}((m \log c)/c)$ advice bits; if $m \in \omega(n)$, an even sublinear bound in m follows. We now describe an online algorithm ChFunc with advice that uses the characteristic function of $\mathcal{S}$.

Theorem 8. *There is a c-competitive online algorithm* ChFunc *that uses $(m\lceil \log m \rceil)/(\lceil \log m \rceil + c - 1)$ advice bits.*

Proof. Again, suppose that an optimal solution uses exactly k sets. Let $t := m/(\lceil \log m \rceil + c - 1)$. First, observe that if $k \leq t$, it clearly suffices to communicate

$$\frac{m\lceil \log m \rceil}{\lceil \log m \rceil + c - 1}$$

bits to ChFunc to be optimal by writing the indices of all sets taken by an optimal solution on the advice tape. Clearly, we need one additional bit to indicate that this is the case. Note that we do not need to encode k into the advice, but we can pad the advice by duplicating certain sets instead.

We may therefore assume the other case, i. e., $k > t$. Suppose that, in this case, the oracle again writes the characteristic function of $\mathcal{S}$ on the advice tape. Clearly, ChFunc may use $(c-1)k \geq (c-1)t$ more sets than an optimal solution to guarantee c-competitiveness. This means that if ChFunc knows the first $m - (c-1)t$ bits of the advice tape, it acted optimal to this point and may safely take one set for each remaining uncovered element. We get

$$m - (c-1)t = m - \frac{(c-1)m}{\lceil \log m \rceil + c - 1} = \frac{m\lceil \log m \rceil}{\lceil \log m \rceil + c - 1} .$$

As above, the number k does not need to be communicated to ChFunc.

Since m and c are known to ChFunc by construction, no further advice is necessary. □

4. Lower Bounds to Obtain c-Competitiveness

Now we give a lower bound on the number of advice bits required for any online algorithm that matches the previous upper bound up to a factor of $\log c$. To this end, we need the Azuma-Hoeffding inequality for submartingales. Recall that a sequence $Z_1, Z_2, \ldots, Z_n$ of discrete random variables forms a *submartingale* if $\mathrm{E}[Z_i \mid Z_{i-1}, \ldots, Z_1] \geq Z_{i-1}$.

Lemma 1 (Azuma, Hoeffding [3, 18]). *Let $Z_0, \ldots, Z_n$ be a submartingale, such that $|Z_i - Z_{i-1}| \leq c_i$. Then, for any positive real t,*

$$\Pr[Z_n - Z_0 \leq -t] \leq \exp\left(-\frac{t^2}{2\sum_{i=1}^{n} c_i^2}\right).$$ □

Theorem 9. *Let $c \geq 1$, and let* Alg *be a c-competitive online algorithm with advice. For any integers n and a such that $a > 2c$ and n is divisible by $2ac$,* Alg *needs to use at least*

$$\frac{n}{4c \ln 2} \cdot \frac{(a/c - 2)^2}{(a/c)^2}$$

advice bits on instances for which every set in $\mathcal{S}$ has size a.

PROOF. Let $\mathcal{S}$ contain all possible subsets of size a of X. The input X' consists of $n/(2c)$ requests; thus there are also $n/(2c)$ time steps. As a consequence, an optimal solution chooses $n/(2ac)$ sets from $\mathcal{S}$, and every c-competitive online algorithm is allowed to pick at most $n/(2a)$ sets.

We use the probabilistic method. In the first time step, the adversary chooses some element x_1 from X with probability $1/n$; in the second time step, it chooses another element x_2 from the remaining ones with probability $1/(n-1)$, and so on. In general, in time step i, some element x_i is chosen with probability $1/(n-i+1)$, uniformly at random from the not-yet-chosen elements.

In what follows, let A denote some fixed deterministic online algorithm. Furthermore, let Y_i, with $1 \leq i \leq n/(2c)$, be a random variable that is defined by

$$Y_i := \begin{cases} 1 & \text{if } A \text{ has chosen more than } n/(2a) \text{ sets in time steps } 1, \ldots, i-1, \\ 1 & \text{if } A \text{ has chosen at most } n/(2a) \text{ sets in time steps } 1, \ldots, i-1 \text{ and } x_i \text{ is uncovered}, \\ 0 & \text{else .} \end{cases}$$

The intuition behind this definition is that if, for any n', $\sum_{i=1}^{n'} Y_i$ is at most $n/(2a)$, this sum counts the number of sets from $\mathcal{S}$ that A takes up to time step n'. Conversely, if this sum is larger than $n/(2a)$, this means that A already took too many sets in the first n' time steps.

We bound the expected value of Y_i conditioned by $Y_{i-1}, \ldots, Y_1$ from below. First, suppose A took more than $n/(2a)$ sets in the first $i-1$ time steps. Then $\mathrm{E}[Y_i \mid Y_{i-1}, \ldots, Y_1] = 1$. In the second case, A took k sets and $k \leq n/(2a)$. On the one hand, since every set in $\mathcal{S}$ has size a, we know that A covers at most $n/2$ elements when an element is requested in time step i. On the other hand, the adversary has requested $i-1$ elements so far, thus there remain $n-i+1$ candidate elements out of which one is chosen uniformly at random. Thus, there are at least $n/2$ uncovered elements and each of them belongs to the $n-i+1$ candidates. Since $(n/2)/(n-i+1) \geq 1/2$, we have that

$$\mathrm{E}[Y_i \mid Y_{i-1}, \ldots, Y_1] \geq \frac{1}{2} \tag{8}$$

in any case.

Next, let $Z_0 := 0$ and Z_i, with $1 \leq i \leq n/(2c)$, be defined as $Z_i := -\frac{i}{2} + \sum_{j=1}^{i} Y_i$, which implies $Z_i - Z_{i-1} = Y_i - 1/2$. Note that since $0 \leq Y_i \leq 1$, for every i with $1 \leq i \leq n/(2c)$, we have

$$|Z_i - Z_{i-1}| = |Y_i - \frac{1}{2}| \leq \frac{1}{2} =: c_i \; . \tag{9}$$

Using linearity of expectation, we get

$$\begin{aligned} \mathrm{E}[Z_i \mid Z_{i-1}, \ldots, Z_1] &= \mathrm{E}\left[Y_i - \frac{1}{2} + Z_{i-1} \mid Z_{i-1}, \ldots, Z_1\right] \\ &= \mathrm{E}[Z_{i-1} \mid Z_{i-1}, \ldots, Z_1] + \mathrm{E}[Y_i \mid Z_{i-1}, \ldots, Z_1] - \frac{1}{2} \\ &\geq Z_{i-1} \; . \end{aligned}$$

For the last inequality we used $\mathrm{E}[Z_{i-1} \mid Z_{i-1}, \ldots, Z_1] = Z_{i-1}$, (8), and the fact that $Y_{i-1}, \ldots, Y_1$ uniquely define $Z_{i-1}, \ldots, Z_1$ and vice versa. Consequently, $Z_1, \ldots, Z_{n/(2c)}$ form a submartingale.

We are interested in the probability that A is c-competitive, i. e., chooses at most $n/(2a)$ sets from $\mathcal{S}$. Choosing $t := n/(4c) - n/(2a)$, we can express this probability as

$$\Pr\left[\sum_{i=1}^{n/(2c)} Y_i \leq \frac{n}{2a}\right] = \Pr\left[\sum_{i=1}^{n/(2c)} Y_i - \frac{n}{4c} \leq -t\right] = \Pr\left[Z_{n/(2c)} \leq -t\right] = \Pr\left[Z_{n/(2c)} - Z_0 \leq -t\right] \; ,$$

and we can use Lemma 1 with the above definition of t and $c_i = 1/2$ due to (9), yielding

$$\begin{aligned}\Pr\left[\sum_{i=1}^{n/(2c)} Y_i \leq \frac{n}{2a}\right] &\leq \exp\left(-\frac{(n(1/(4c) - 1/(2a)))^2}{2\sum_{j=1}^{n/(2c)} 1/4}\right) \\ &= \exp\left(-\frac{n(1/(4c) - 1/(2a))^2}{1/(4c)}\right) \\ &= \exp\left(-n\frac{(a-2c)^2}{4a^2c}\right) .\end{aligned}$$

As a result, the deterministic online algorithm A covers at most this fraction of the inputs that are defined as above.

Now consider an online algorithm ALG with advice that reads $b(n)$ bits in total for inputs of length n. We can view ALG as selecting the best of $2^{b(n)}$ deterministic algorithms from Strat(ALG) for every given input. If ALG is c-competitive, for any given input it has to use one deterministic online algorithm that is c-competitive on this input. In other words, it must hold that

$$2^{b(n)} \cdot \exp\left(-n\frac{(a-2c)^2}{4a^2c}\right) \geq 1 .$$

Taking the natural logarithm, we get

$$b(n) \geq n\frac{(a-2c)^2}{4a^2c\ln 2} = \frac{n}{4c\ln 2} \cdot \frac{(a-2c)^2}{a^2} \tag{10}$$

as an upper bound on the number of advice bits necessary, and the claim of the theorem follows. □

To get a more tangible result, we formulate the following corollary.

Corollary 1. *Let $c \geq 1$, and let* ALG *be a c-competitive online algorithm with advice. Then* ALG *needs to use at least $n/(4\ln(2)c) - o(1)$ advice bits.*

PROOF. Consider any integer $n > 4c^2$ divisible by $4c^2$. We apply Theorem 9 for $a := n/(2c)$. Note that the instances used in the proof of the theorem are such that the optimal solution uses one set from $\mathcal{S}$ to cover all requests. We get

$$b(n) \geq \frac{n}{4c\ln 2} \cdot \frac{(a/c-2)^2}{(a/c)^2} = \frac{n}{4c\ln 2} \cdot \left(1 - \frac{4}{a/c} + \frac{4}{(a/c)^2}\right) = \frac{n}{4c\ln 2} \cdot \left(1 - \frac{8}{n} + \frac{16}{n^2}\right) ,$$

which proves the claim. □

Next, we complement this result by a lower bound that is linear in m. To this end, we use an approach somewhat similar to an iterated application of the proof of Theorem 9.

Theorem 10. *Let $c \geq 1$, and let* ALG *be a c-competitive online algorithm with advice. Then* ALG *needs to use at least $m/c^{\mathcal{O}(c)}$ advice bits.*

PROOF. We again use the probabilistic method. Basically, in what follows, we concatenate a number of disjoint subinstances, in which the optimal cost is one, while any given deterministic online algorithm has to use an additional set for every given request with probability at least $1/2$. We first describe a single subinstance I'. For a given c, I' contains $a := 4c$ requests that are chosen uniformly at random among all not yet requested objects. There are $2a^2$ objects in total and every set has size a.

Now we concatenate q such instances resulting in an instance $I = (x_1, x_2, \ldots, x_{aq})$ with aq requests, $|X| = 2a^2q$, and $|\mathcal{S}| = \binom{2a^2}{a}q$; clearly, $\text{cost}(\text{OPT}(I)) = q$. Let A denote a deterministic online algorithm. Similarly to the proof of Theorem 9, we define random variables Y_i with $1 \leq i \leq aq$ by

$$Y_i := \begin{cases} 1 & \text{if } A \text{ has not covered } x_i \text{ when it is offered,} \\ 0 & \text{else .} \end{cases}$$

Since, for every subinstance, no algorithm covers more than $(a-1)a$ objects (except possibly after the last time step) and there are $2a^2$ objects in total, the current request is not covered with a probability of at least $1/2$, i. e., $\text{E}[Y_i \mid Y_{i-1}, \ldots, Y_1] \geq 1/2$. Again, let $Z_0 := 0$ and $Z_i := -\frac{i}{2} + \sum_{j=1}^{i} Y_i$; hence, $Z_i - Z_{i-1} = Y_i - 1/2$ and $Z_1, Z_2, \ldots, Z_{aq}$ again form a submartingale. Note that A is c-competitive as long as $\text{cost}(A(I)) \leq qc$. Let $t := q(a/2 - c)$; the probability that A is c-competitive is

$$\Pr\left[\sum_{i=1}^{aq} Y_i \leq qc\right] = \Pr\left[\sum_{i=1}^{aq} Y_i - \frac{aq}{2} \leq -t\right] = \Pr[Z_{aq} - Z_0 \leq -t] \,.$$

Following Lemma 1 with $c_i = 1/2$, we get

$$\Pr[Z_{aq} - Z_0 \leq -t] \leq \exp\left(-\frac{2t^2}{aq}\right) = \exp\left(-\frac{2q(a/2-c)^2}{a}\right) = \exp\left(-\frac{2qc^2}{4c}\right) = \exp\left(-\frac{qc}{2}\right) .$$

Now again consider ALG that uses $b(n)$ bits; we get

$$2^{b(n)} \cdot \exp\left(-\frac{qc}{2}\right) ,$$

and thus

$$b(n) \geq \frac{qc}{2\ln 2} = \frac{mc}{\binom{2a^2}{a} 2\ln 2} = \frac{mc}{\binom{32c^2}{4c} 2\ln 2} \geq \frac{mc!}{(32c)^{8c}} \geq \frac{m}{(32c)^{8c}}$$

advice bits are necessary to be c-competitive, which proves the claim. □

5. Parameters

In this section, we analyze the set cover problem with respect to other parameters, namely, the size ℓ of the largest set in $\mathcal{S}$, the size k of an optimal solution, and the maximum number p of times an item occurs in the sets of $\mathcal{S}$.

5.1. The Size of the Largest Set in $\mathcal{S}$

In this subsection, we study the advice complexity with respect to the parameters k and ℓ. First, since $\text{cost}(\text{ALG}(I))/\text{cost}(\text{OPT}(I)) \leq |X'|/(|X'|/\ell) = \ell$, every deterministic online algorithm without advice is ℓ-competitive. To see that this bound is tight, consider the instance given by $X := \{1, \ldots, \ell^2\}$ and $\mathcal{S} := \{S \subseteq X \mid |S| = \ell\}$. The adversary requests ℓ items; one set from $\mathcal{S}$ is thus sufficient to cover all requests. However, in each round, the adversary requests an item not covered by the sets previously picked by ALG. Thus ALG has to choose ℓ different sets and is ℓ-competitive.

For the next result, we need the following generalization of Sperner's Theorem [27] (see (2)).

Observation 2. *For any set X of size n, the maximum size of an antichain of X, i. e., a set family of incomparable subsets of X, consisting of sets of size at most ℓ is equal to $\binom{n}{\ell}$ if $\ell \leq n/2$ and $\binom{n}{n/2}$ if $\ell \geq n/2$.* □

We use this to obtain an upper bound on the number of advice bits sufficient to produce an optimal output with respect to the parameters n, ℓ, and k.

Theorem 11. *There is an optimal online algorithm with advice that uses at most $k\ell \log n$ advice bits, for $\ell \geq 4$ and $k \geq 5$.*

Proof. Consider an algorithm Alg that works as follows. First, it reads $\lceil \log k \rceil + 2\lceil \log \lceil \log k \rceil \rceil$ advice bits from the tape that encode k in a self-delimiting way. Then it reads, for every requested item that is not yet covered by a set previously specified by Alg, the index of the set to pick. Using Observation 2, this is possible with at most

$$\binom{n}{\min\{\ell, n/2\}}$$

advice bits. Hence, the algorithm reads at most

$$k\left\lceil \log\left(\binom{n}{\min\{\ell, n/2\}}\right)\right\rceil + \lceil \log k \rceil + 2\lceil \log \lceil \log k \rceil \rceil$$

advice bits in total.

In the following, we use the estimates $\lceil \log k \rceil + 2\lceil \log \lceil \log k \rceil \rceil \leq 1.4k$, for $k \geq 5$, (1), and $\log(\binom{n}{n/2}) \leq \log(n^{n/2}) - 2.4$. We distinguish two cases.

Case 1. If $\ell \geq n/2$, we can simply provide the index of the set to pick in every step. This needs at most $\log(\binom{n}{n/2})$ advice bits, thus in total, we get $k\left(\log(\binom{n}{n/2}) + 2.4\right) \leq k \log(n^{n/2}) = k(n \log n)/2 \leq k\ell \log n$.

Case 2. If $\ell \leq n/2$, we get

$$k\left(\log\left(\binom{n}{\ell}\right) + 2.4\right) \overset{(1)}{\leq} k\left(\ell \log\left(\frac{n\mathrm{e}}{\ell}\right) + 2.4\right) = k(\ell \log n - \ell \log \ell + \ell \log \mathrm{e} + 2.4) \leq k\ell \log n\,,$$

where the last inequality holds for $\ell \geq 4$.

The claim follows. □

A corresponding lower bound can be established in a straightforward fashion.

Theorem 12. *Every optimal online algorithm with advice has to use at least $(\ell - 1)\log((n-1)/(\ell-1))$ advice bits.*

Proof. Again, $\mathcal{S}$ consists of all $\binom{n}{l}$ subsets of X of size exactly ℓ. The adversary requests exactly ℓ elements, so the optimal solution has size 1. After the first element is requested, Alg has to select the unique set that covers all requested elements among all $\binom{n-1}{\ell-1}$ possible sets. Hence, Alg needs to read at least

$$\log\left(\binom{n-1}{\ell-1}\right) \overset{(1)}{\geq} \log\left(\left(\frac{n-1}{\ell-1}\right)^{\ell-1}\right) = (\ell-1)\log\left(\frac{n-1}{\ell-1}\right)$$

advice bits. □

Furthermore, one easily obtains a $(1 + 1/k)$-competitive online algorithm that uses at most $n - l$ advice bits. Let S be the first set in $\mathcal{S}$ of size ℓ with respect to the canonical order. The algorithm reads $n - l$ advice bits and interprets them as the characteristic function of X' with respect to $X \setminus S$. Then, it computes an optimal solution for this set and specifies only sets from this optimal solution or S. Thus, its computed solution has size at most $k+1$, and the competitive ratio of the algorithm is $(k+1)/k = 1+1/k$.

5.2. The Largest Number of Occurrences of an Item

In this subsection, we investigate the advice complexity with respect to the parameters k and the maximum number p of times an item occurs in the sets of $\mathcal{S}$. We show that $k\lceil \log p \rceil$ advice bits are sufficient and $\lfloor k \log p \rfloor$ advice bits are necessary to be optimal.

For the upper bound, an algorithm reads, in every step, from the tape the index of the set to take. Since this index can be given with respect to the number of sets in which the currently requested element occurs, it has size $\lceil \log p \rceil$. For the lower bound, let n be a multiple of $p+1$, and let X consist of items y_i and $x_{i,j}$, for $1 \leq i \leq n/(p+1)$, $1 \leq j \leq p$. Furthermore, $\mathcal{S}$ contains all sets $\{y_i, x_{i,j}\}$. First, the adversary requests all items y_i. Thus, ALG has to specify $n/(p+1)$ sets to cover these items. For $1 \leq i \leq n/(p+1)$, the adversary then requests one of the p possible items $x_{i,j}$. This determines a unique optimal solution among all the $p^{n/(p+1)}$ possible ones. Thus, ALG has to read at least $\lfloor \log\left(p^{n/(p+1)}\right) \rfloor = \lfloor (n \log p)/(p+1) \rfloor = \lfloor k \log p \rfloor$ advice bits.

Finally, the proof of Theorem 8 can also be used to prove that there is a c-competitive online algorithm ALG with advice that uses at most $(m\lceil \log p \rceil)/(\lceil \log p \rceil + c - 1)$ advice bits.

6. Conclusion

In this paper, we studied online algorithms with advice for the online set cover problem. We showed that linear advice is both sufficient and necessary in order to compute an optimal solution, with respect to both the size of the ground set and the set family. The main contribution is the c-competitive online algorithm MAXCOVER with advice that uses $\mathcal{O}((n \log c)/c)$ advice bits, and the lower bounds of $n/(4\ln(2)c)$ and $m/c^{\mathcal{O}(c)}$. Furthermore, we studied advice with respect to different parameters. We showed that there is an optimal online algorithm with advice that uses $(k\ell \log n)$ advice bits, where ℓ is the size of the largest set in $\mathcal{S}$, and k is the size of an optimal solution. This is complemented by a lower bound of roughly $\ell \log(n\ell)$. Additionally, we showed that $k \log p$ advice bits are both necessary and sufficient to be optimal, where p denotes the maximum number of times that an item occurs in the sets of $\mathcal{S}$.

As usual in the context of advice complexity, we ignored the running time of the algorithms we designed (see also Observation 1). If we consider optimal online algorithms with advice, either the algorithm (see Theorem 3, where information about the instance is encoded) or the oracle (see Theorem 5, where information about the optimal solution is encoded) need to solve an NP-hard problem. For c-competitiveness, our lower bounds are only strengthened by the fact that we do not impose any time restrictions on both the algorithm and oracle. As for upper bounds, MAXCOVER (and also its oracle) needs to calculate an optimal solution, which cannot be done in polynomial time unless $\mathsf{P} \neq \mathsf{NP}$. In general, it makes sense to analyze the advice complexity of efficient online algorithms. Almost nothing is known about the comparison of online algorithms with advice with unrestricted and restricted runtime, opening an interesting field for further research.

Moreover, it would certainly be interesting to get more insight about how the advice complexity of this and other problems behaves with respect to different natural parameters that might be known (at least in parts) in a practical scenario. Using such an approach, we touch other branches of models of online computation, such as *semi-online algorithms*.

Acknowledgment

The authors would like to thank Juraj Hromkovič for his technical comments on some of the proofs of the paper that improved the presentation.

References

[1] S. Albers. Online algorithms: a survey. *Mathematical Programming*, 97(1):3–26, 2003.

[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.

[3] K. Azuma. Weighted sums of certain dependent random variables. *Tôhoku Mathematical Journal*, 19(3):357–367, 1967.

[4] H.-J. Böckenhauer, J. Hromkovič, D. Komm, S. Krug, J. Smula, and A. Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. In: *Proc. of the 19th Annual International Computing and Combinatorics Conference (COCOON 2013)*, LNCS 7936, pages 493–505, Springer, 2013.

[5] H.-J. Böckenhauer, D. Komm, R. Královič, and R. Královič. On the advice complexity of the k-server problem. In *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, LNCS 6755, pages 207–218. Springer, 2011.

[6] H.-J. Böckenhauer, D. Komm, R. Královič, and P. Rossmanith. On the advice complexity of the knapsack problem. In *Proc. of the 10th Latin American Symposium on Theoretical Informatics (LATIN 2012)*, LNCS 7256, pages 61–72. Springer, 2012.

[7] H.-J. Böckenhauer, D. Komm, R. Královič, R. Královič, and T. Mömke. On the advice complexity of online problems. In *Proc. of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, LNCS 5878, pages 331–340. Springer, 2011.

[8] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, Springer, 1994.

[9] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[10] J. Boyar, L.M. Favrholdt, C. Kudahl, K.S. Larsen, and J.W. Mikkelsen: Online algorithms with advice: a survey. *SIGACT News*, 47(3):93–129, 2016.

[11] A. Chattopadhyay, J. Edmonds, F. Ellen, and T. Pitassi. A little advice can be very helpful. In *Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 615–625, 2012.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2009.

[13] S. Dobrev, R. Královič, and D. Pardubská. How much information about the future is needed? In *Proc. of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008)*, LNCS 4910, pages 247–258. Springer, 2008.

[14] S. Dobrev, R. Královič, and D. Pardubská. Measuring the problem-relevant information in input. *Theoretical Informatics and Applications (RAIRO)*, 43(3):585–613, 2009.

[15] Y. Emek, P. Fraigniaud, A. Korman, and A. Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.

[16] A. Fiat, G. J. Woeginger (eds.). *Online Algorithms – The State of the Art*, Lecture Notes in Computer Science 1442. Springer, 1998.

[17] J. Hromkovič, R. Královič, and R. Královič. Information complexity of online problems. In *Proc. of the 35th Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, Lecture Notes in Computer Science 6281, pages 24–36. Springer, 2010.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[19] S. Irani and A. R. Karlin. On online computation. In *Approximation Algorithms for $\mathcal{NP}$-Hard Problems, chapter 13*, pages 521–564. 1997.

[20] D. Komm. *An Introduction to Online Computation: Determinism, Randomization, Advice.* Texts in Theoretical Computer Science. An EATCS Series, Springer 2016.

[21] D. Komm and R. Královič. Advice complexity and barely random algorithms. *Theoretical Informatics and Applications (RAIRO)* 45(2):249–267, 2011.

[22] D. Komm, R. Královič, and T. Mömke. On the advice complexity of the set cover problem. In *Proc. of the 7th International Computer Science Symposium in Russia (CSR 2012)*, Lecture Notes in Computer Science 7353, pages 241–252. Springer, 2012.

[23] J. W. Mikkelsen: Randomization can be as helpful as a glimpse of the future in online computation. In *Proc. of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Leibniz International Proceedings in Informatics 55, pages 39:1–39:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik 2016.

[24] M. Mitzenmacher and E. Upfal. *Probability and Computing.* Cambridge University Press, 2005.

[25] M. Pătraşcu. Towards polynomial lower bounds for dynamic problems. *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC 2010)* pages 603–610. 2010.

[26] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[27] E. Sperner. Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.