A Technique to Obtain Hardness Results for Randomized Online Algorithms – A Survey

Hans-Joachim Böckenhauer, Juraj Hromkovič$^{(\boxtimes)}$, and Dennis Komm

Department of Computer Science, ETH Zürich,
Universtitätsstrasse 6, 8092 Zürich, Switzerland
{hjb,juraj.hromkovic,dennis.komm}@inf.ethz.ch

Abstract. We survey how the advice complexity of online algorithms can be used to obtain lower bounds on the performance of randomized online algorithms. Online algorithms with advice may query an oracle that knows the whole input from the start to solve some instance of an online problem. This is done by reading a finite prefix of some infinite binary *advice tape*, which is created by the oracle before the first piece of input is processed. Similarly, a randomized online algorithm may use a binary tape where every bit is chosen uniformly at random.

In this survey, we review a technique, similar to Yao's principle, which allows statements on the advice complexity of some given online problem to translate to results on the power of randomization for this problem in terms of lower bounds. We give some examples where this technique works and how it is applied, and show its limitations and that it is tight in a very general sense.

1 Introduction

In online computation, an adversary produces some hard input of an optimization problem that is fed to an online algorithm, denoted by `A`, piece by piece over a number of discrete time steps. In every such time step, `A` needs to produce a corresponding piece of output which cannot be changed afterwards. In this paper, we focus on a class of problems where the objective is to minimize some cost function which is associated with the given online problem.

Many real-world problems such as the paging problem or many routing and scheduling problems are modeled this way; for instance, suppose you want to assign jobs to a fixed number of resources, e. g., processors. Such a service is offered to a large number of customers and processor time can be booked at any given point in time. But this means that assignments should be made long before all requests are known. Still, the objective is to minimize the waiting time for, say, the customer that waits the longest. Such a situation is a typical online scenario.

At first, we formally define the term *online minimization problem*, where the input consists of *requests* that arrive in consecutive time steps and the output is created piecewise as a sequence of *answers* to these requests.

C.S. Calude et al. (Eds.): Gruska Festschrift, LNCS 8808, pp. 264–276, 2014.
DOI: 10.1007/978-3-319-13350-8_20

Definition 1 (Online Minimization Problem). *An* online minimization problem *consists of a set $\mathcal{I}$ of inputs and a cost function. Every input $I \in \mathcal{I}$ is a sequence $I = (x_1, \ldots, x_n)$ of* requests*. Furthermore, a set of feasible outputs (or solutions) is associated with every I; every output is a sequence $O = (y_1, \ldots, y_n)$ of* answers*. The* cost function *assigns a positive real value* $\text{cost}(I, O)$ *to every input I and any feasible output O. For every input I, we call any feasible output O for I that has smallest possible cost (i. e., that minimizes the cost function) an* optimal solution *for I.*

Classically, one searches for online algorithms that perform well in the sense that they produce output which has a cost that is as small as possible compared to the cost of an optimal solution. The study of such online algorithms is coined "competitive analysis" [11]. Note that the optimal solution can usually not be computed with full accuracy in an online manner. We therefore speak of an optimal "offline" solution. In a recent model considered in this paper one asks an advanced question that is beyond pure competitive analysis. In particular, we are interested in the (amount of) information that is both needed and sufficient to outperform purely deterministic or even randomized online algorithms. In a sense, we want to know which information about the yet unknown parts of the input is crucial to obtain a low competitive ratio or even an optimal solution. Let us first give a formal framework to study *online algorithms with advice.* To this end, we use the standard definition as first given in [9, 26]. The intuitive idea is that, after the adversary created an input for a given online problem, an oracle inspects this input and writes binary information about it on a tape (the *advice tape*). Then, an online algorithm starts processing the input as usual, but it may, with every request, read some part of the tape (sequentially) to get additional information about the input. The minimum length of the prefix that the algorithm needs to read while guaranteeing some quality on every input, is then the advice complexity of this algorithm.

Definition 2 (Online Algorithm with Advice). *Consider an input I of an online minimization problem. An* online algorithm $\mathtt{A}$ with advice *computes the output sequence $\mathtt{A}^{\phi}(I) = (y_1, \ldots, y_n)$ 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. $\mathtt{A}$ is* c-competitive with advice complexity $b(n)$ *if there exists a non-negative constant α such that, for every n and for any input sequence I of length at most n, there exists some advice string ϕ such that*

$$\text{cost}(\mathtt{A}^{\phi}(I)) \leq c \cdot \text{cost}(\textsc{Opt}(I)) + \alpha$$

and at most the first $b(n)$ bits of ϕ have been accessed during the computation of the solution $\mathtt{A}^{\phi}(I)$. If the above inequality holds with $\alpha = 0$, we call $\mathtt{A}$ strictly c-competitive with advice complexity $b(n)$. $\mathtt{A}$ *is called* optimal *if it is strictly 1-competitive.*

A first model of online computation with advice was introduced by Dobrev et al. [18]. As this model was not precise enough to measure the number of advice bits needed, Hromkovič et al. proposed the general model used here, and discussed its relation to the general notion of the *information content* of a problem [26]. The fruitfulness of this model was for the first time explored by Böckenhauer et al. [9], where it was applied to paging, disjoint path allocation, and job shop scheduling. At the same time, Emek et al. [20] proposed a similar model and studied the k-server problem and metrical task systems. Since then, new results on job shop scheduling [28], the k-server problem [8,23,32], and disjoint path allocation [2] were obtained. Additionally, many other online problems were studied including buffer management [14], online set cover [29], string guessing [7], graph exploration [17], online independent set [15], online knapsack [10], online makespan scheduling [19,33], online bin packing [12,33], online Steiner tree [1], list update [13] and online graph coloring [3,4,22,34]. Online algorithms using both advice and randomization were investigated by Böckenhauer et al. [6]. Further connections between computing online with advice and randomized online computation where, e. g., observed by Komm and Královič [28]. Our main observation on the topic of this paper was first made by Böckenhauer et al. [8], it establishes a non-trivial relationship between randomized online algorithms and online algorithms with advice for a given online minimization problem.

Note the resemblance between Definition 2 and the definition of the expected competitive ratio of a randomized online algorithm, which we give in what follows to fix our notation.

Definition 3. *Randomized Online Algorithm] Consider an input I of an online minimization problem. A* randomized online algorithm $\mathtt{R}$ *computes the output sequence $\mathtt{R}^{\phi}(I) = (y_1, \ldots, y_n)$ such that y_i is computed from $\phi, x_1, \ldots, x_i$, where ϕ is the content of a* random tape*, i. e., an infinite binary sequence, where every bit is chosen uniformly at random and independently of all the others. By* $\mathrm{cost}(\mathtt{R}^{\phi}(I))$*, we denote the random variable expressing the cost of the solution computed by $\mathtt{R}$ on I. $\mathtt{R}$ is c-*competitive in expectation *if there exists a non-negative constant α such that, for every I,*

$$\mathbb{E}\big[\mathrm{cost}(\mathtt{R}^{\phi}(I))\big] \le c \cdot \mathrm{cost}(\mathtt{Opt}(I)) + \alpha,$$

where, as above, $\mathtt{Opt}$ *is an optimal offline algorithm for the problem.*

Throughout this paper, since ϕ is always clear from context, we omit it and simply write, e. g., $\mathtt{R}$ instead of $\mathtt{R}^{\phi}$. We observe that the basic change is from speaking of "one best solution" (i. e., there always is one binary sequence ϕ that guarantees some success) to speaking of "all solutions *on average*" (i. e., in expectation, we can guarantee some success).

Let us take another point of view which will come in handy later. We consider a function $b\colon \mathbb{N} \to \mathbb{N}$ that measures the number of advice bits some randomized online algorithm uses on inputs of size n, for any $n \in \mathbb{N}$. For the ease of presentation, we will assume that $\mathtt{R}$ uses *exactly* $b(n)$ random bits on every input of size n. For any fixed bit string on $\mathtt{R}$'s random tape, the algorithm's decisions are fully determined by the input. As a result, we can think of $\mathtt{R}$ as a probability distribution over a set of $2^{b(n)}$ deterministic strategies. We denote this set by

Alg($\mathtt{R}$) $= \{\mathtt{A}_1, \ldots, \mathtt{A}_{2^{b(n)}}\}$. We further assume from now on that $\mathtt{R}$ picks a deterministic strategy uniformly at random from Alg($\mathtt{R}$). We are now ready to present the key theorem for proving lower bounds on the expected competitive ratio of randomized online algorithms.

2 The Main Theorem

As already mentioned, we want to focus on the relationship between advice and randomization. Before revisiting the main theorem [8], we make the following two observations, which are immediate.

1. If there is a randomized online algorithm $\mathtt{R}$ for some online problem Π such that $\mathtt{R}$ uses b random bits and achieves an expected competitive ratio of c, then there also is an online algorithm with advice for Π that is c-competitive and uses b advice bits.
2. Conversely, if there is provably no online algorithm with advice for Π that is c-competitive while using b advice bits, then there also is no randomized online algorithm using b random bits while being c-competitive in expectation.

If we follow our intuition, advice bits seem to be a lot more powerful than random bits. After all, we compare a situation where we always pick a best strategy for any instance, to a situation where we pick strategies with a fixed distribution; in essence, we compare the best to the average. We therefore ask whether there exists a scenario in which it is possible to save some bits if they are supplied by an oracle and not a random source. In what follows, we give a positive answer to this question. More specifically, we show that, if there is some randomized online algorithm $\mathtt{R}$ for some online minimization problem Π, then there also is some online algorithm with advice that is almost as good while using a number of advice bits (and that is the interesting part) which does not depend on the number of random bits $\mathtt{R}$ uses. However, the bound does depend on the number of possible instances of Π of given length. The proof uses some ideas that are similar to the proof of Yao's theorem [35].

Theorem 1 (Böckenhauer et al. [8]). *Let Π be an online minimization problem for which we have $m(n)$ different inputs of length n. Moreover, suppose there is a randomized online algorithm $\mathtt{R}$ for Π, which achieves an expected competitive ratio of c. Then there is an online algorithm $\mathtt{A}$ with advice for Π, which achieves a competitive ratio of $(1+\varepsilon)c$, for any $\varepsilon > 0$, while using at most*

$$\lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil + \log\left(\left\lfloor \frac{\log(m(n))}{\log(1+\varepsilon)} \right\rfloor + 1 \right)$$

advice bits.

$$\begin{array}{c|cccc} & \mathtt{A}_1 & \mathtt{A}_2 & \mathtt{A}_3 & \cdots \\ \hline I_1 & c_{1,1} & c_{1,2} & c_{1,3} & \cdots \\ I_2 & c_{2,1} & c_{2,2} & c_{2,3} & \\ I_3 & c_{3,1} & c_{3,2} & c_{3,3} & \\ \vdots & \vdots & & & \ddots \end{array}$$

Fig. 1. An example matrix $\mathcal{M}$ as used in the proof

Proof. We suppose that, for any input length $n \in \mathbb{N}$, $\mathtt{R}$ uses $b(n)$ random bits. As observed above, this is equivalent to choosing uniformly at random a deterministic strategy from a set $\mathrm{Alg}(\mathtt{R}) = \{\mathtt{A}_1, \ldots, \mathtt{A}_{2^{b(n)}}\}$. Since $\mathtt{R}$ is c-competitive in expectation, there is a constant α such that for every instance I, we have

$$\mathbb{E}[\mathrm{cost}(\mathtt{R}(I))] \leq c \cdot \mathrm{cost}(\mathtt{Opt}(I)) + \alpha$$

or, equivalently,

$$\frac{\mathbb{E}[\mathrm{cost}(\mathtt{R}(I))] - \alpha}{\mathrm{cost}(\mathtt{Opt}(I))} \leq c.$$

Now, for each deterministic strategy $\mathtt{A}_j$ and for each instance I_i of length n, $1 \leq j \leq 2^{b(n)}$ and $1 \leq i \leq m(n)$, we set

$$c_{i,j} := \max\left\{1, \frac{\mathrm{cost}(\mathtt{A}_j(I_i)) - \alpha}{\mathrm{cost}(\mathtt{Opt}(I_i))}\right\},$$

and we call $c_{i,j}$ the *performance* of $\mathtt{A}_j$ on I_i. Next, we construct an $(m(n) \times 2^{b(n)})$-matrix $\mathcal{M}$ that we fill with these entries as shown in Fig. 1. As a result, the entry in the ith row and the jth column gives the performance of $\mathtt{R}$ on the input I_i if $\mathtt{R}$ chooses the deterministic strategy $\mathtt{A}_j$. The central idea of the proof is to show that we are able to cleverly choose a small number of columns of $\mathcal{M}$ such that the performances of the corresponding deterministic strategies are small for many instances, and the sets of the chosen strategies together cover all input instances. We collect these algorithms in a set $\mathcal{S}$ and $\mathtt{A}$ gets as advice the index of the algorithm from $\mathcal{S}$ that should be used for the input at hand (and some additional information we describe later).

One row i of $\mathcal{M}$ corresponds to exactly one input I_i. Thus, by the definition of $c_{i,j}$ and the expected competitive ratio of $\mathtt{R}$, for every i, $1 \leq i \leq m(n)$, we get

$$\begin{aligned} \frac{1}{2^{b(n)}} \sum_{j=1}^{2^{b(n)}} c_{i,j} &= \frac{1}{2^{b(n)}} \sum_{j=1}^{2^{b(n)}} \frac{\mathrm{cost}(\mathtt{A}_j(I_i)) - \alpha}{\mathrm{cost}(\mathtt{Opt}(I_i))} \\ &= \frac{\frac{1}{2^{b(n)}} \sum_{j=1}^{2^{b(n)}} \mathrm{cost}(\mathtt{A}_j(I_i)) - \alpha}{\mathrm{cost}(\mathtt{Opt}(I_i))} \\ &= \frac{\mathbb{E}[\mathrm{cost}(\mathtt{R}(I_i))] - \alpha}{\mathrm{cost}(\mathtt{Opt}(I_i))} \\ &\leq c \end{aligned}$$

or, equivalently,

$$\sum_{j=1}^{2^{b(n)}} c_{i,j} \leq c \cdot 2^{b(n)},$$

and for the sum of all entries in all cells of $\mathcal{M}$, we get

$$\sum_{i=1}^{m(n)} \sum_{j=1}^{2^{b(n)}} c_{i,j} \leq \sum_{i=1}^{m(n)} c \cdot 2^{b(n)} \leq c \cdot 2^{b(n)} \cdot m(n).$$

Since there are $2^{b(n)}$ columns in $\mathcal{M}$, there is one column (deterministic strategy) j' such that

$$\sum_{i=1}^{m(n)} c_{i,j'} \leq c \cdot m(n).$$

The online algorithm $\mathtt{A}_{j'}$ is then included in $\mathcal{S}$ and it is used for every instance I_i, for which $c_{i,j'} \leq (1+\varepsilon)c$. Let s denote the number of these instances. In what follows, we want to estimate how large s is, i. e., for how many instances $\mathtt{A}$ can use $\mathtt{A}_{j'}$. Clearly, the performance of $\mathtt{A}_{j'}$ is larger than $(1+\varepsilon)c$ on $m(n)-s$ instances.

Summing up, this gives a total of $(m(n)-s)(1+\varepsilon)c$ for the corresponding rows and we have

$$(m(n)-s)(1+\varepsilon)c < \sum_{i=1}^{m(n)} c_{i,j'}.$$

From this, it follows that $(m(n)-s)(1+\varepsilon)c < m(n) \cdot c$ and therefore $s > \varepsilon/(1+\varepsilon) \cdot m(n)$, which means we can use the deterministic strategy $\mathtt{A}_{j'}$ for a fraction of $\varepsilon/(1+\varepsilon)$ of the instances as we know that on these its performance is not larger than $(1+\varepsilon)c$.

After $\mathtt{A}_{j'}$ is put into the set $\mathcal{S}$, we delete the column j' from $\mathcal{M}$ together with all rows that correspond to inputs on which $\mathtt{A}_{j'}$ achieves a sufficiently small performance. There remain

$$\left(1-\frac{\varepsilon}{1+\varepsilon}\right) m(n) = \left(\frac{1}{1+\varepsilon}\right) m(n)$$

rows for which we need to find another algorithm from Alg($\mathtt{R}$). For every remaining row, the deleted entry in column j' was larger than c. It follows that, after removing this column, the average over all entries of remaining rows is still not larger than c. Therefore, we can repeat the aforementioned method with the remaining

$$\left(\frac{1}{1+\varepsilon}\right) m(n)$$

rows of $\mathcal{M}$. This way, we find another deterministic online algorithm $\mathtt{A}_{j''}$, which has a sufficiently small performance on a fraction of $\varepsilon/(1+\varepsilon)$ of the remaining instances.

Now we compute how often we have to iterate this strategy at most until we have found an algorithm for every input. This means that we want to find a natural number r such that

$$\left(\frac{1}{1+\varepsilon}\right)^r m(n) < 1.$$

We get

$$\left(\frac{1}{1+\varepsilon}\right)^r < \frac{1}{m(n)} \iff (1+\varepsilon)^r > m(n) \iff r > \log_{1+\varepsilon}(m(n)),$$

which means that we have to make at most

$$\left\lfloor \frac{\log(m(n))}{\log(1+\varepsilon)} \right\rfloor + 1$$

iterations, i. e., we need that many deterministic algorithms from Alg($\mathtt{R}$). This immediately gives an upper bound on the size of $\mathcal{S}$.

Finally, we calculate the number of advice bits needed for this approach.

1. First, $\mathtt{A}$ needs to know the input length n, which can be encoded on the advice tape using $\log n$ bits. However, this must be done in a self-delimiting fashion, using, e. g., Elias encoding [21], summing up to a total of $2\lceil\log\lceil\log n\rceil\rceil + \lceil\log n\rceil$ advice bits at most.
2. Knowing n, $\mathtt{A}$ constructs $\mathcal{M}$ by simulating the randomized online algorithm $\mathtt{R}$ on any possible input. Then, $\mathtt{A}$ constructs $\mathcal{S}$ and enumerates all algorithms from $\mathcal{S}$ in, e. g., canonical order. After reading another

$$\log\left(\left\lfloor \frac{\log(m(n))}{\log(1+\varepsilon)} \right\rfloor + 1\right)$$

advice bits, $\mathtt{A}$ can pick one algorithm from $\mathcal{S}$, which is then simulated for the input at hand.

It follows that the competitive ratio of $\mathtt{A}$ on any instance is at most $(1+\varepsilon)c$ and $\mathtt{A}$ uses as much as advice as claimed by the theorem. □

The contraposition of Theorem 1 is particularly interesting. Suppose we can show that any online algorithm with advice needs an amount of advice that is asymptotically larger than the value from the theorem statement to be c-competitive. The it follows that no randomized online algorithm can be c-competitive in expectation. We will apply this approach to $L(2,1)$-coloring and to the k-server problem in Section 4.

3 Limits of this Approach

As we will see in the next section, the above mentioned technique is widely applicable. However, before giving examples, we want to point out some limitations and drawbacks.

First, note that the online algorithm A with advice does not run in polynomial time (even if R is efficient) if $m(n)$ is large with respect to the input length n, because A needs to construct the whole matrix $\mathcal{M}$. Clearly, $\mathcal{M}$ cannot be a part of A as it depends on n, which, of course, is not known by A, but is part of the advice.

Second, the online algorithm A is worse than the original randomized online algorithm R, even if the difference is very small. A natural question is whether it is possible to improve Theorem 1 such that A obtains the same competitive ratio as R. Intriguingly, this is not possible, so we really need this small gap. Consider the following online problem. The input $I = (x_1, \ldots, x_n)$ starts with a request $x_1 = 0$. All other requests are bits, i. e., $x_i \in \{0, 1\}$, for $2 \leq i \leq n$. Moreover, all answers must be bits, i. e., $y_i \in \{0, 1\}$, for $1 \leq i \leq n-1$. If $y_i = x_{i+1}$, for all i, $1 \leq i \leq n-1$, the total cost of the corresponding solution is 1, else it is 2. Thus, an optimal algorithm pays 1 and every other solution pays 2. Obviously, a best randomized online algorithm chooses every answer such that it is either 0 or 1 with a probability of 1/2 each. This algorithm uses $n-1$ random bits and its expected competitive ratio is not larger than

$$\frac{\frac{2^{n-1}-1}{2^{n-1}} \cdot 2 + \frac{1}{2^{n-1}} \cdot 1}{1} = 2 - \frac{1}{2^{n-1}}.$$

Conversely, every online algorithm with advice that uses less than $n-1$ advice bits is at most 2-competitive. It follows that there are problems such that any online algorithm with advice needs as many advice bits as a randomized online algorithm needs random bits, or it is worse off.

Third, in this paper as well as in [8], we only considered online minimization problems. With a similar argument, however, it can be shown that an analogous statement for online maximization problems is possible [16].

4 Applications

In this section, we show how to apply Theorem 1 both directly and indirectly, thus creating both online algorithms with advice and lower bounds for randomized online algorithms for a selection of online problems.

4.1 Job Shop Scheduling

First, we study the online job shop scheduling problem with l machines and k jobs that consist of l unit length tasks each, denoted by (k, l)-JSS. More precisely, we are given k different jobs that need to use l different machines in some fixed order. A machine can only process one task at a time. Since every job asks for every machine exactly once, we can view a job as a permutation of the machine indices. Thus, an example input for $(4, 8)$-JSS is

$$\text{Job}_1 = (1, 2, 3, 4, 5, 6, 7, 8), \quad \text{Job}_2 = (3, 8, 4, 1, 2, 7, 6, 5),$$
$$\text{Job}_3 = (7, 1, 5, 6, 8, 3, 4, 2), \quad \text{Job}_4 = (8, 2, 4, 5, 3, 1, 6, 7),$$

which means that, e. g., the second job first needs the third machine, then the eighth, and so on. An algorithm for (k, l)-JSS must assign the machines to the jobs in the given order. In an online framework, these permutations arrive in consecutive time steps such that the $(i+1)$th machine index of a job is revealed after the ith request is satisfied (i. e., assigned to a machine). In the example above, all four machines ask for four different machines in the first time step. Therefore, machine 1 can be assigned to Job_1, machine 3 is assigned to Job_2, and so on. In time step 2, however, two jobs, namely Job_1 and Job_4, ask for the same machine 2. In such a situation, an online algorithm needs to delay one of the two. Obviously, an optimal choice depends on future time steps.

The advice complexity of $(2, l)$-JSS was studied before by Böckenhauer et al. [9] and Komm and Královič [28]. Hromkovič et al. [27] constructed a randomized online algorithm which achieves a competitive ratio of $1 + 2k/\sqrt{l}$ in expectation. Now let us apply Theorem 1. There are $(l!)^k$ distinct instances of (k, l)-JSS of length $n = kl$, i. e., k-tuples of permutations of length l each. Using Stirling's approximation [24], we therefore get

$$m(n) = (l!)^k \leq \left(\left(1 + \frac{1}{11l}\right)\left(\frac{l}{\mathrm{e}}\right)^l \sqrt{2l\pi}\right)^k$$

different inputs. Applying Theorem 1, it follows that, for any $\varepsilon > 0$, there is a $((1+2k/\sqrt{l}) \cdot (1+\varepsilon))$-competitive online algorithm with advice that uses at most

$$\begin{aligned}
&\lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil + \log\left(\left\lfloor \frac{\log(m(n))}{\log(1+\varepsilon)} \right\rfloor + 1\right) \\
&\leq 2\lceil \log kl \rceil + \log\left(dk \log\left(\left(\frac{l}{\mathrm{e}}\right)^l \sqrt{2l\pi}\right)\right) \\
&\leq 2\lceil \log kl \rceil + \log(d' kl \log l) \\
&\leq d'' \log(d''' kl) \\
&\leq d'' \log(kl) + d'''
\end{aligned}$$

advice bits, for some constants d, d', d'', d'''. Thus, the advice complexity grows merely logarithmically in k and l. Note that, since the input length n is fully determined by these two parameters, the advice complexity can even be reduced further by a constant factor since we do not need a self-delimiting encoding of n.

Theorem 2. *There is a $(1+2k/\sqrt{l})$-competitive online algorithm* A *with advice for (k, l)-JSS which reads $\mathcal{O}(\log kl)$ advice bits.*

This is the first time, an upper bound of the general version of the problem (i. e., for k different jobs) was studied in terms of advice complexity. So far, the case for $k = 2$ was investigated by Böckenhauer et al. [9] and Komm and Královič [28].

4.2 $L(2,1)$-Coloring

Another application of Theorem 1 is a version of graph coloring that arises in the context of assigning frequencies to transmitters in a multihop radio network. The difference between the frequencies that are used by the transmitters should be anti-proportional to their proximity to avoid interference. A simple graph-theoretic model of the frequency assignment problem has been introduced by Griggs and Yeh [25]. Here, the transmitters are the vertices of a graph and the frequencies are modeled by colors from a finite, ordered set, usually $\{0, 1, \ldots, \lambda\}$, for some natural number λ. In the easiest case, two levels of proximity are considered, neighboring vertices have to be assigned colors with a distance of at least 2 in the given order, and vertices at distance 2 in the graph still have to get different colors. The resulting problem of finding a coloring minimizing the color range λ is called $L(2,1)$-coloring. The advice complexity of the online version of the $L(2,1)$-coloring problem was studied by Bianchi et al. [4]. Here, a graph is given online, one vertex after another, and together with every vertex, exactly those edges are uncovered that are adjacent to vertices that are already known. If a vertex is revealed in some time step, an online algorithm must immediately assign it a color.

Among other results, Bianchi et al. [4] showed that every online algorithm with advice for $L(2,1)$-coloring with a competitive ratio of $5/4$ needs to read at least $3.9402 \cdot 10^{-10}n$ advice bits, even if the online graph has a maximum degree of 2, i. e., is a collection of paths and cycles. Although the constant factor in this linear lower bound is very small, the following lower bound on the expected competitive ratio of any randomized online algorithm was proven using Theorem 1.

Theorem 3 (Bianchi et al. [4]). *For arbitrarily small $\delta > 0$, every randomized algorithm for the online $L(2,1)$-coloring problem on graphs with maximum degree 2 has a worst-case expected competitive ratio of at least $\frac{5}{4}(1-\delta)$ on sufficiently large instances.*

Proof sketch. By an easy counting argument, there are $m(n) \leq 2^{\binom{n}{2}}n!$ online graphs on n vertices. It is easy to see that there exists a threshold n_0 on the input length such that the bound from Theorem 1 with this value of $m(n)$ plugged in exceeds $3.9402 \cdot 10^{-10}n$ for all $n \geq n_0$ (see the original paper [4] for the details of the calculation).

Since we already know that (even when restricting the considered online graphs to paths), with this number of advice bits, no online algorithm with advice can be better than $5/4$, the result follows immediately by Theorem 1. □

4.3 The k-Server Problem

The k-server problem is one of the most prominent online minimization problems. In this setup, we are given a metric space and k so-called servers that can be moved through this space. In every time step, a request is made that is given by some point. An answer is given by a server that is moved to this point, incurring a cost that is given by the distance between the original position of the server and the requested point.

Introduced in 1988 by Manasse et al. [31], the k-server problem is still not fully understood. There are, so far, two conjectures about the best possible deterministic and randomized online algorithms for the problem. Here, we want to focus on the *randomized k-server conjecture* which claims that there is a randomized online algorithm which is $\Theta(\log k)$-competitive in expectation. The following theorem shows how our technique could possibly be used to disprove the conjecture.

Theorem 4 (Böckenhauer et al. [8]). *If every online algorithm with advice for the k-server problem needs to use at least $\omega(\log n)$ advice bits to be $\mathcal{O}(\log k)$-competitive, the randomized k-server conjecture does not hold.*

Proof. Let us only consider inputs for k-server such that the size of the metric space is bounded from above by 2^n, where n is the input length. As above, let $m(n)$ denote the number of inputs of length n. In every time step, a point is requested, thus

$$m(n) = (2^n)^n.$$

Consider a randomized online algorithm R that is $\mathcal{O}(\log k)$-competitive in expectation on all of these instances. Then, there exists a constant $c > 0$ such that R is $(c \cdot \log k)$-competitive. Let $\varepsilon = 1$. Following Theorem 1, there also is a $(2 \cdot c \cdot \log k)$-competitive online algorithm with advice, which uses at most

$$\lceil \log n \rceil + 2\lceil \log \lceil \log n \rceil \rceil + \log(\lfloor \log((2^n)^n) \rfloor + 1) \in \mathcal{O}(\log n)$$

advice bits.

If we could prove that any online algorithm with advice needs asymptotically more advice, this is a contradiction to the existence of R. □

Note that, so far, the best known randomized online algorithm for k-server for arbitrary metric spaces is the $(2k-1)$-competitive algorithm of Koutsoupias and Papadimitriou [30]. Considering the advice complexity, a lower bound is only known for optimality (approximately $n \log k$ advice bits are necessary [8]). Furthermore, Renault and Rosén [32] constructed a $\lceil \lceil \log k \rceil / (b-2) \rceil$-competitive online algorithm with advice which reads b advice bits per request (this improves a previous result by Böckenhauer et al. [8] by a factor of 2). To be $\mathcal{O}(\log k)$-competitive, this algorithm thus needs a number of advice bits that is linear in n. As shown in the proof of Theorem 4, if the randomized k-server conjecture holds, there is an $\mathcal{O}(\log k)$-competitive online algorithm with advice that uses $\mathcal{O}(\log n)$ advice bits as long as the size of the metric space is at most 2^n. Thus, there remains an interesting exponential gap. A step towards proving the conjecture was made by Bansal et al. who constructed a randomized online algorithm with an expected competitive ratio of $\mathcal{O}((\log l)^3 (\log k)^2 \log \log n)$ in expectation [5], where l is the number of points of the underlying metric space. This algorithm improves over the one by Koutsoupias and Papadimitriou if $l \in o(2^{(2k)^{-(3+\varepsilon)}})$ and its competitive ratio is polylogarithmic in k if l is polynomial in k.

References

1. Barhum, K.: Tight Bounds for the Advice Complexity of the Online Minimum Steiner Tree Problem. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 77–88. Springer, Heidelberg (2014)
2. Barhum, K., Böckenhauer, H.-J., Forišek, M., Gebauer, H., Hromkovič, J., Krug, S., Smula, J., Steffen, B.: On the Power of Advice and Randomization for the Disjoint Path Allocation Problem. In: Geffert, V., Preneel, B., Rovan, B., Štuller, J., Tjoa, A.M. (eds.) SOFSEM 2014. LNCS, vol. 8327, pp. 89–101. Springer, Heidelberg (2014)
3. Bianchi, M.P., Böckenhauer, H.-J., Hromkovič, J., Keller, L.: Online Coloring of Bipartite Graphs with and without Advice. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 519–530. Springer, Heidelberg (2012)
4. Bianchi, M.P., Böckenhauer, H.-J., Hromkovič, J., Krug, S., Steffen, B.: On the Advice Complexity of the Online $L(2,1)$-Coloring Problem on Paths and Cycles. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 53–64. Springer, Heidelberg (2013)
5. Bansal, N., Buchbinder, N., Madry, A., Naor, J.: A polylogarithmic-competitive algorithm for the k-server problem (extended abstract). In: Proc. of FOCS 2011, pp. 267–276 (2011)
6. Böckenhauer, H.-J., Hromkovič, J., Komm, D., Královič, R., Rossmanith, P.: On the Power of Randomness versus Advice in Online Computation. In: Bordihn, H., Kutrib, M., Truthe, B. (eds.) Languages Alive. LNCS, vol. 7300, pp. 30–43. Springer, Heidelberg (2012)
7. Böckenhauer, H.-J., Hromkovič, J., Komm, D., Krug, S., Smula, J., Sprock, A.: The String Guessing Problem as a Method to Prove Lower Bounds on the Advice Complexity. In: Du, D.-Z., Zhang, G. (eds.) COCOON 2013. LNCS, vol. 7936, pp. 493–505. Springer, Heidelberg (2013)
8. Böckenhauer, H.J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.): ICALP 2011, Part I. LNCS, vol. 6755, pp. 207–218. Springer, Heidelberg (2011)
9. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the Advice Complexity of Online Problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 331–340. Springer, Heidelberg (2009)
10. Böckenhauer, H.-J., Komm, D., Královič, R., Rossmanith, P.: On the Advice Complexity of the Knapsack Problem. In: Fernández-Baca, D. (ed.) LATIN 2012. LNCS, vol. 7256, pp. 61–72. Springer, Heidelberg (2012)
11. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
12. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: Online bin packing with advice. In: Proc. of STACS 2014. LIPIcs 25, pp. 174–186. Schloss Dagstuhl (2014)
13. Boyar, J., Kamali, S., Larsen, K.S., López-Ortiz, A.: On the List Update Problem with Advice. In: Dediu, A.-H., Martín-Vide, C., Sierra-Rodríguez, J.-L., Truthe, B. (eds.) LATA 2014. LNCS, vol. 8370, pp. 210–221. Springer, Heidelberg (2014)
14. Dorrigiv, R., He, M., Zeh, N.: On the Advice Complexity of Buffer Management. In: Chao, K.-M., Hsu, T., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 136–145. Springer, Heidelberg (2012)
15. Dobrev, S., Královič, R., Královič, R.: Independent Set with Advice: The Impact of Graph Knowledge. In: Erlebach, T., Persiano, G. (eds.) WAOA 2012. LNCS, vol. 7846, pp. 2–15. Springer, Heidelberg (2013)

16. I. Seleceniova. Personal communication.
17. Dobrev, S., Královič, R., Markou, E.: Online Graph Exploration with Advice. In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 267–278. Springer, Heidelberg (2012)
18. Dobrev, S., Královič, R., Pardubská, D.: How Much Information about the Future Is Needed? In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 247–258. Springer, Heidelberg (2008)
19. Dohrau, J.: Online makespan scheduling with sublinear advice. Technical Report, ETH Zurich (2013)
20. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online Computation with Advice. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 427–438. Springer, Heidelberg (2009)
21. Elias, P.: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory **21**(2), 194–203 (1975)
22. Forišek, M., Keller, L., Steinová, M.: Advice Complexity of Online Coloring for Paths. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 228–239. Springer, Heidelberg (2012)
23. Gupta, S., Kamali, S., López-Ortiz, A.: On Advice Complexity of the k-server Problem under Sparse Metrics. In: Moscibroda, T., Rescigno, A.A. (eds.) SIROCCO 2013. LNCS, vol. 8179, pp. 55–67. Springer, Heidelberg (2013)
24. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics, 2nd ed. Addison-Wesley (1994)
25. Griggs, J.R., Yeh, R.K.: Labelling graphs with a condition at distance 2. SIAM Journal on Discrete Mathemtics **5**(4), 586–595 (1992)
26. Hromkovič, J., Královič, R., Královič, R.: Information Complexity of Online Problems. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 24–36. Springer, Heidelberg (2010)
27. Hromkovič, J., Mömke, T., Steinhöfel, K., Widmayer, P.: Job shop scheduling with unit length tasks: Bounds and algorithms. Algorithmic Operations Research **2**(1), 1–14 (2007)
28. Komm, D., Královič, R.: Advice complexity and barely random algorithms. Theoretical Informatics and Applications (RAIRO) 45(2), 249–267 (2011)
29. Komm, D., Královič, R., Mömke, T.: On the Advice Complexity of the Set Cover Problem. In: Hirsch, E.A., Karhumäki, J., Lepistö, A., Prilutskii, M. (eds.) CSR 2012. LNCS, vol. 7353, pp. 241–252. Springer, Heidelberg (2012)
30. Koutsoupias, E., Papadimitriou, C.H.: On the k-server conjecture. Journal of the ACM **42**(5), 971–983 (1995)
31. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for on-line problems. In: Proc. of STOC 1998, pp. 322–333 (1988)
32. Renault, M.P., Rosén, A.: On Online Algorithms with Advice for the k-Server Problem. In: Solis-Oba, R., Persiano, G. (eds.) WAOA 2011. LNCS, vol. 7164, pp. 198–210. Springer, Heidelberg (2012)
33. Renault, M.P., Rosén, A., van Stee, R.: Online algorithms with advice for bin packing and scheduling problems. CoRR abs/1311.7589 (2013)
34. Seibert, S., Sprock, A., Unger, W.: Advice Complexity of the Online Coloring Problem. In: Spirakis, P.G., Serna, M. (eds.) CIAC 2013. LNCS, vol. 7878, pp. 345–357. Springer, Heidelberg (2013)
35. Yao, A.C.-C.: Probabilistic computations: Toward a unified measure of complexity (extended abstract). In: Proc. of FOCS 1977, pp. 222–227. IEEE Computer Society (1977)