On the power of randomness versus advice in online computation

Book Chapter**Author(s):**

Böckenhauer, Hans-Joachim; Hromkovič, Juraj; Komm, Dennis; Kráľovič, Richard; Rossmanith, Rossmanith

Publication date:

2012

Permanent link:

<https://doi.org/10.3929/ethz-a-010886804>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

Lecture Notes in Computer Science 7300, https://doi.org/10.1007/978-3-642-31644-9_2

Funding acknowledgement:

141089 - Measuring the Information Content of Online Problems (SNF)

On the Power of Randomness Versus Advice in Online Computation^{*}

Hans-Joachim Böckenhauer¹, Juraj Hromkovič¹, Dennis Komm¹,
Richard Kráľovič¹, and Peter Rossmanith^{2**}

¹ Department of Computer Science, ETH Zurich, Switzerland,
{hjb,juraj.hromkovic,dennis.komm,richard.kralovic}@inf.ethz.ch

² Department of Computer Science, RWTH Aachen, Germany,
rossmani@cs.rwth-aachen.de

Abstract. The recently introduced model of *advice complexity* of online problems tries to achieve a fine-grained analysis of the hardness of online problems by asking how many bits of advice about the still unknown parts of the input an oracle has to provide for an online algorithm to guarantee a specific competitive ratio. Until now, only deterministic online algorithms with advice were considered in the literature. In this paper, we consider, for the first time, online algorithms having access to both random bits and advice bits. For this, we introduce the online problem (n, k) -BOXES: Given a number of n closed boxes, an adversary hides $k < \sqrt{n}$ items, each of unit value, within k consecutive boxes. The goal is to open exactly k boxes and gain as many items as possible. In the classical online setting without advice, we show that, if $k(k+1) \leq n$, any deterministic algorithm is not competitive, because the adversary can ensure that not a single item is found. However, randomization drastically increases the gain in expectation. More precisely, we prove that the expected gain is in the order of k^3/n and show that this bound is tight up to some constant factor. A crucial result of our analysis is the proof of the existence of two thresholds for the amount of random bits used for solving (n, k) -BOXES. If the amount of random bits is below the first threshold, randomization does not help at all. If, on the other hand, the amount of randomness is above the second threshold of about $\log n - 2 \log k$ random bits, then any additional random bit does not help to improve the gain.

As our main result, we analyze the advice complexity of the boxes problem both for deterministic and randomized online algorithms and give a tight trade-off between the number of random bits and advice bits needed for achieving a specific competitive ratio.

1 Introduction

In algorithmics, we seek for algorithms that produce high-quality output for some specific problem within some given time or space bounds. In many real-world applications, however, we are facing an additional hurdle when working in

^{*} This work was partially supported by SNF grant 200021-132510/1.

^{**} Part of this work was done while this author was staying at ETH Zurich.

so-called *online environments*; here, the input arrives piecewise in consecutive discrete time steps while parts of the output have to be produced before the whole input is known (for an introduction and comprehensive discussion, we refer to the standard literature, e. g., [3, 7, 8, 10]). Formally, we are dealing with the following class of problems.

Definition 1. *A maximization online problem consists of a set $\mathcal{I}$ of inputs and a cost function. Every input $I \in \mathcal{I}$ is a sequence of requests $I = (x_1, \dots, x_n)$. Furthermore, a set of feasible outputs (or solutions) is associated with every I ; every output is a sequence of answers $O = (y_1, \dots, y_n)$. The cost function assigns a positive real value $\text{cost}(I, O)$ to every input I and any feasible output O . If the input is clear from the context, we omit I and denote the cost of O as $\text{cost}(O)$. For every input I , we call any output O that is feasible for I and has highest possible cost an optimal solution of I , denoted by $\text{Opt}(I)$.*

The established tool for measuring the performance of an online algorithm is the *competitive analysis* [3, 12] where one compares the cost of the solution computed by the online algorithms to the cost of an optimal offline solution computed by an algorithm knowing the whole input in advance.

Definition 2. *Consider an input $I = (x_1, \dots, x_n)$ of a maximization online problem. An online algorithm A computes the output sequence $A(I) = (y_1, \dots, y_n)$ such that y_i is computed from $x_1, \dots, x_i$. We denote the costs of the computed output by $\text{cost}(A(I))$.*

The online algorithm A is c -competitive if there exists a non-negative constant α such that, for every n and for each I of length at most n , $\text{cost}(A(I)) \geq 1/c \cdot \text{cost}(\text{Opt}(I)) - \alpha$. If $\alpha = 0$, A is called strictly c -competitive; A is called optimal if it is strictly 1-competitive.

Due to the definition of the problem we study in what follows, we only consider *strict* competitiveness. However, in many cases, competitive analysis does not seem very realistic because, by the nature of many real-world online scenarios, optimal results can never be reached. We want to get a better understanding of what online algorithms really lack. For a more fine-grained analysis of how much knowledge about the future parts of the input an online algorithm needs to compute a high-quality solution, we consider the *information content* of the given online problem as it was defined in [9]. This model can be viewed as a cooperation of an online algorithm A and an *oracle* O that has unlimited computational power, sees the whole input in advance, and writes binary information about it onto an *advice tape* before A reads any part of the input. Afterwards, A can access the bits from the advice tape in a sequential manner, just as a randomized algorithm would use its random tape. The *advice complexity* of the online problem is then defined as the minimum number of advice bits needed for achieving a good solution. The following definition formalizes this concept.

Definition 3. *Consider an input I of a maximization online problem. An online algorithm A with advice computes the output sequence $A^\phi(I) = (y_1, \dots, y_n)$ 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. We denote the costs of the computed output by $\text{cost}(\mathbf{A}^\phi(I))$. $\mathbf{A}$ has advice complexity $s(n)$ if at most the first $s(n)$ bits of ϕ have been accessed during the computation of $\mathbf{A}^\phi(I)$. The c -competitiveness can be defined analogously to Definition 2.

A first model of online algorithms with advice was introduced in [4] (see also [5]) and later refined in [2, 6]. For a survey of the concepts of advice complexity and the *information content* of online problems, see [9]. This model was applied to several classical online problems such as paging, job shop scheduling, the k -server problem, or metrical task systems in [1, 2, 5, 6, 11].

Another powerful tool for increasing the output quality of online algorithms is to allow randomized computation. One can easily extend Definition 2 for randomized algorithms as well. For the first time, we consider algorithms using both randomization *and* advice, and we give tight bounds on the trade-off between the amount of randomness and advice needed for achieving a specific competitive ratio.

We are now ready to introduce the online problem, denoted by (n, k) -BOXES, we deal with in the following.

Definition 4 ((n, k)-Boxes). *There are n boxes $b_1, \dots, b_n$ standing in a row and we know that all are empty except for $k < \sqrt{n}$ boxes that are standing next to each other and which contain some expensive item each. An online algorithm $\mathbf{A}$ is allowed to open exactly k boxes of its choice aiming at opening as many full ones as possible. After $\mathbf{A}$ has opened k boxes, the (remaining) positions of the non-empty boxes are revealed and $\mathbf{A}$'s gain is the number of non-empty boxes it has opened.*

Note that the optimal solution of (n, k) -BOXES has always gain k . We call the position of the first full box the *starting position* and we note that, for any instance of size n , there are exactly $n - k + 1$ possible starting positions. Another way to look at the analysis of online problems is to view it as a game between the online algorithm $\mathbf{A}$ against an oblivious adversary Adv . The adversary tries to hide the full boxes in such a way that $\mathbf{A}$'s gain is (in expectation and/or for any advice string) as small as possible.

At first, we can make the following straightforward observation about any deterministic online algorithm without advice for (n, k) -BOXES.

Theorem 1. *If $n \geq k(k + 1)$, Adv can ensure that no deterministic algorithm $\mathbf{A}$ has any gain.*

Proof. Adv knows $\mathbf{A}$'s deterministic strategy; however, for every box at position i that $\mathbf{A}$ opens, the number of starting positions to hide the k items is decreased by at most k . The removed positions are those between i to $i - k + 1$ where $i \geq k$ (see Fig. 1); not all of them exist if $i < k$. Hence, if $\mathbf{A}$ chooses boxes such that these intervals are disjoint, Adv is prevented to take at most k^2 starting positions. Accordingly, if $n \geq k^2 + k$, Adv may hide the items in such a way that $\mathbf{A}$ is not able to find any item at all. $\square$

Fig. 1: A inspecting box b_5 and the “forbidden” starting positions for Adv

Thus, any deterministic algorithm is not competitive. In the next section, we show the power of randomization for (n, k) -BOXES.

2 Randomization

Since we are interested in values for k and n that are in the above order and the situation seems desperate for any deterministic online algorithm, we consider randomized algorithms, i. e., algorithms that use a source of randomness to make their decisions. Which box is chosen next can depend on which boxes that are already open are full and on the randomness that is available to the algorithm. Usually, the source of randomness is just a stream of random bits. It is then possible to measure the amount of randomness as the number of random bits used as a function of the input size. However, the input of the (n, k) -BOXES problem has always constant size, since n and k are fixed. As we are interested (without loss of generality) in randomized algorithms with a fixed upper bound on the running time, for any randomized algorithm R solving (n, k) -BOXES, there is an upper bound r on the number of random bits used by R .

Measuring the amount of randomness as the number of random bits is, however, rather coarse; in the sequel, we aim for a more fine-grained analysis. If R is allowed to use r random bits, then, equivalently, it can use a random number between 1 and 2^r instead. We generalize this model by considering algorithms that get one random number drawn uniformly from the set $\{1, \dots, M\}$ where M is not restricted to be a power of two. Let $\mathcal{R}(M)$ denote the set of randomized algorithms with randomness restricted to the uniformly random choice of a number from $\{1, \dots, M\}$.

If we fix an algorithm R , then the random variable $B_s(R)$ denotes the number of full boxes opened by R for starting position s . Then $\mathbb{E}[B_s(R)]$ is the expected number of full boxes opened if Adv chooses the starting position equal to s , and $\min_s \mathbb{E}[B_s(R)]$ is the worst-case expected gain of R . If R is clear from the context, we abbreviate $B_s(R)$ by B_s .

The following theorem gives a worst-case lower bound on the performance of any randomized algorithm $R \in \mathcal{R}(M)$ solving (n, k) -BOXES with restricted randomness.

Theorem 2. *For any randomized online algorithm $R \in \mathcal{R}(M)$ for (n, k) -BOXES, there exists a starting position s (i. e., an input instance of (n, k) -BOXES), such that*

1. *if $M < (n - k + 1)/k^2$, then $\mathbb{E}[B_s] = 0$,*
2. *for any M , $\mathbb{E}[B_s] \leq \frac{k^2(k+1)}{2(n-k+1)}$,*
3. *if $M = c(n - k + 1)/k^2$ with $c \in [1, 2k/(k+3)]$, then*

$$(a) \mathbb{E}[B_s] \leq \frac{2(k+\frac{3}{2})}{M} - \frac{2(n-k+1)}{M^2k} \text{ and also}$$

$$(b) \mathbb{E}[B_s] \leq \frac{2(c-1)}{c^2} \cdot \frac{k^3}{n-k+1} + \frac{3k^2}{c(n-k+1)}.$$

Proof. R gets a random number from $\{1, \dots, M\}$, so it can behave in M different ways and each behavior occurs with probability $1/M$. This is equivalent to choosing uniformly at random one deterministic algorithm from a set $\{A_1, \dots, A_M\} = \text{Alg}(R)$.

Recall that the k full boxes start at some starting position, i. e., a position between 1 and $n - k + 1$.

1. Suppose that $M < (n - k + 1)/k^2$. Basically, we now extend the idea from the proof of Theorem 1. The number of starting positions that allow Adv to place the obstacles in a way that R does not find any item is $n - k + 1 - (Mk^2) > n - k + 1 - (n - k + 1) = 0$, which implies the claim.
2. At first, we can make a simple observation: Consider some specific (deterministic) algorithm $A_i \in \text{Alg}(R)$. There are at most k starting positions such that A_i opens a full box in the first step and gets at most k full boxes in total. Next, there are at most k starting positions such that A_i opens an empty box in the first step and a full box in the second step and gets at most $k - 1$ full boxes in total, and so on.

Assume that the starting position is p . Let $\overline{O}_{i,p}$ denote the number of empty boxes opened by A_i until the first full box is found (or k if none) and let $O_{i,p} := k - \overline{O}_{i,p}$ and $O_p = O_{1,p} + \dots + O_{M,p}$. Clearly, $O_{i,p}$ is an upper bound on the number of full boxes opened by A_i in total and O_p/M is an upper bound on the expected number of full boxes opened by R . Hence, taking s such that O_s is minimal, the expected number of full boxes opened by R for starting position s is

$$\mathbb{E}[B_s] = \frac{1}{M} O_s = \frac{1}{M} \min \{O_1, \dots, O_{n-k+1}\}. \quad (1)$$

From the observation above it follows that, for every A_i , we have

$$O_{i,1} + \dots + O_{i,n-k+1} \leq k^2 + k(k-1) + k(k-2) + \dots + k = \frac{k^2(k+1)}{2}$$

and accordingly, using (1) and the fact that the minimum of a set of numbers can never be larger than its average, we get

$$\begin{aligned}
\mathbb{E}[B_s] &\leq \frac{1}{M} \cdot \frac{\sum_p O_p}{n-k+1} = \frac{1}{M(n-k+1)} \sum_p \sum_{i=1}^M O_{i,p} \\
&= \frac{1}{M(n-k+1)} \sum_{i=1}^M \sum_p O_{i,p} = \frac{1}{M(n-k+1)} \sum_{i=1}^M \frac{k^2(k+1)}{2} \\
&= \frac{1}{M} \cdot \frac{Mk^2(k+1)}{2(n-k+1)} = \frac{k^2(k+1)}{2(n-k+1)}. \tag{2}
\end{aligned}$$

3. Let us consider values of M such that $\frac{n-k+1}{k^2} \leq M \leq \frac{2(n-k+1)}{k(k+3)}$. Choose any value g such that $g \leq k/M$ and $g \geq \frac{k+1}{M} - \frac{n-k+1}{kM^2}$. We call a starting position p *big* if $O_{i,p} \geq gM$ for some i (i.e., there is a single algorithm A_i that makes a large contribution to O_p). Otherwise we call it *small*. Due to the assumptions on g , we have that $Mk(k - \lfloor gM \rfloor) \in [0, n-k+1]$. For any A_i , p is made big by A_i if $O_{i,p}$ takes values between $\lceil gM \rceil$ and k , which leads to $k - \lceil gM \rceil + 1 \geq k - \lfloor gM \rfloor$ different values. For any such value x , there are at most k positions p such that $O_{i,p} = x$, and since there are M algorithms in total to consider, there are at most $Mk(k - \lfloor gM \rfloor)$ big positions and, accordingly, at least $n-k+1 - Mk(k - \lfloor gM \rfloor) \geq 0$ small positions.

Let S be the set of all small positions. Since no algorithm A_j is allowed to contribute strictly more than $\lfloor gM \rfloor$ to these positions, we get, for any A_i ,

$$\sum_{p \in S} O_{i,p} \leq k(1 + 2 + \dots + \lfloor gM \rfloor) = \frac{k \lfloor gM \rfloor (\lfloor gM \rfloor + 1)}{2}$$

and consequently

$$\sum_{p \in S} O_p \leq \frac{Mk \lfloor gM \rfloor (\lfloor gM \rfloor + 1)}{2}.$$

In particular, there is some small position p with

$$\begin{aligned}
O_p &\leq \frac{Mk \lfloor gM \rfloor (\lfloor gM \rfloor + 1)}{2|S|} \leq \frac{Mk \lfloor gM \rfloor (\lfloor gM \rfloor + 1)}{2(n-k+1 - Mk(k - \lfloor gM \rfloor))} \\
&\leq \frac{MkgM(gM+1)}{2(n-k+1 - Mk(k - gM+1))} \tag{3}
\end{aligned}$$

and, since $\mathbb{E}[B_p] = \frac{O_p}{M}$, there exists s such that

$$\mathbb{E}[B_s] \leq \frac{k g M (g M + 1)}{2(n-k+1 - Mk(k - gM+1))}. \tag{4}$$

We choose

$$g = \frac{2(k + \frac{3}{2})}{M} - \frac{2(n-k+1)}{M^2 k}.$$

It is straightforward to verify that g satisfies our assumptions. Plugging this value of g into (4), we get $\mathbb{E}[B_s] \leq g$. Finally, by setting $M = c(n-k+1)/k^2$, we get

$$\mathbb{E}[B_s] \leq \frac{2(c-1)}{c^2} \cdot \frac{k^3}{n-k+1} + \frac{3k^2}{c(n-k+1)}.$$

This concludes the proof. $\square$

We now complement this lower bound by an upper bound that is tight up to a small constant factor.

Theorem 3. *Let M be an even number. There is a randomized algorithm $\mathbf{R} \in \mathcal{R}(M)$ for (n, k) -BOXES such that, for every starting position s ,*

1. *if $M \geq \frac{2n}{k(k-1)}$, then $\mathbb{E}[B_s] \geq \frac{8}{9} \frac{k^3}{2n} - 1$,*
2. *if $\frac{n-k+1}{k^2} < M \leq \frac{2n}{k(k-1)}$, then $\mathbb{E}[B_s] \geq \frac{2k-2}{M} - \frac{2n}{M^2 k}$.*

Proof. As we mentioned, $\mathbf{R}$ is a probability distribution over $\text{Alg}(\mathbf{R})$ where each deterministic algorithm $\mathbf{A} \in \text{Alg}(\mathbf{R})$ gets chosen with the same probability $1/M$. Every algorithm $\mathbf{A}$ opens boxes within some interval of fixed length and performs a straightforward local search when a box is found, which enables $\mathbf{A}$ to find at least $k-i$ full boxes if a full box is discovered in step i . Moreover, consider an adversary Adv that tries to hide the boxes as good as possible from every algorithm in $\text{Alg}(\mathbf{R})$ at once.

In the following, we focus on an equivalent problem to analyze the algorithm. We shrink the instance to an instance of size $\lfloor n/k \rfloor$, that is, we compress k boxes into one hyper-box, thereby neglecting the last $d = k \lfloor n/k \rfloor < k$ original boxes. There is exactly one non-empty hyper-box that has a value of $k-1$ in the beginning and whose value is decreasing by one with every unsuccessful opening of a hyper-box (except in the last step). The algorithm can open up to k hyper-boxes. When it opens the full hyper-box in the j -th trial, it achieves gain $k-j$ if $j < k$, and gain 1 if $j = k$.

Now we show that it is possible to reduce (n, k) -BOXES to its shrunk version. Indeed, assume that we have an algorithm $\mathbf{A}'$ for the shrunk version. We can construct an algorithm $\mathbf{A}$ for (n, k) -BOXES as follows. Whenever $\mathbf{A}'$ opens some hyper-box, $\mathbf{A}$ opens the last box corresponding to this hyper-box. As soon as $\mathbf{A}$ finds some full box, it continues with a local search.

Consider any input instance for $\mathbf{A}$, which is specified by the starting position p . Then, $\mathbf{A}$ achieves at least the same gain as $\mathbf{A}'$ running on an instance where the hyper-box corresponding to p is full. Since p cannot be within the last $k-1$ boxes, the hyper-box corresponding to p exists. If $\mathbf{A}'$ opens a full hyper-box, the starting position is within the distance k to the left of the box opened by $\mathbf{A}$, therefore $\mathbf{A}$ opens a full box as well. If this happens in step j , the local search of $\mathbf{A}$ guarantees a gain of at least $k-j$ if $j < k$, and of 1 if $j = k$.

In the sequel, we provide a randomized algorithm $\mathbf{R}$ solving the shrunk version of (n, k) -BOXES, thereby proving that an equally good algorithm for the original problem exists. Consider some constant l such that $1 \leq l < 2k$. Starting with

Fig. 2: The worst-case gain within one interval assigned to the $\lfloor n/k \rfloor$ hyper-boxes

some hyper-box q , a deterministic algorithm $\mathbf{A} \in \text{Alg}(\mathbb{R})$ opens k consecutive hyper-boxes until k empty hyper-boxes are inspected or it finds a full hyper-box at the j -th trial, gaining $k - j$ if $j < k$, and 1 if $j = k$. Next, we define the symmetric algorithm to $\mathbf{A}$ denoted by $\bar{\mathbf{A}} \in \text{Alg}(\mathbb{R})$: $\bar{\mathbf{A}}$ initially opens hyper-box $q + l - 1$ and then continues to open $k - 1$ consecutive hyper-boxes left of $q + l - 1$ in reverse order until it arrives at $s + l - k$ or finds the item; $\mathbf{A}$ and $\bar{\mathbf{A}}$ are called an *algorithm pair* because they work on the same interval.

Let us now bound the minimum of the total gain of the two algorithms within one interval of length l . Clearly, if $l = k$, the gain is at least $k - 1$ for every hyper-box, and, more general, if $l = k + i$ for $-k < i < k$, we get a gain of at least $k - 1 - i = 2k - l - 1$ (see Fig. 2). Starting at the first hyper-box, we assign every of the $M/2$ algorithm pairs to one interval of length l in a way such that, by allowing wrap-arounds, every hyper-box is covered by exactly c intervals.

It follows that

$$\mathbb{E}[B_s] \geq \frac{c(2k - l - 1)}{M}$$

if we can guarantee a number of c wrap-arounds (see Fig. 3); to do so, it clearly has to hold that

$$\frac{M}{2} \cdot l \geq \left\lfloor \frac{n}{k} \right\rfloor \cdot c$$

which can be guaranteed by satisfying

$$l \geq \frac{2cn}{Mk} \tag{5}$$

Fig. 3: The algorithm R with wrap-around $c = 2$

and we obviously aim at minimizing l while satisfying (5) which means that we may set $l := \lceil 2cn/Mk \rceil$, yielding

$$\mathbb{E}[B_s] \geq \frac{c(2k - \lceil \frac{2cn}{Mk} \rceil - 1)}{M} \geq \frac{c(2k - \frac{2cn}{Mk} - 2)}{M} = \frac{2ck - 2c}{M} - \frac{2c^2n}{M^2k}. \quad (6)$$

We now distinguish two cases according to the size of M .

1. Suppose $M \geq \frac{2n}{k(k-1)}$. Let

$$\delta := \frac{k^3}{2n} - \frac{k(16k - 8 + k^2)}{18n}.$$

In the following, we want to guarantee that

$$\frac{2ck - 2c}{M} - \frac{2c^2n}{M^2k} \geq \delta$$

and therefore

$$0 \geq k\delta M^2 - (2ck^2 - 2ck)M + 2c^2n \quad (7)$$

to prove the bound we claimed, that is, we have to show that there exists a number of wrap-arounds c for any $M \geq \frac{2n}{k(k-1)}$ such that (7) holds. A simple calculation gives that (7) is satisfied if and only if

$$\frac{3}{2} \cdot \frac{nc}{k(k-1)} \leq M \leq 3 \cdot \frac{nc}{k(k-1)}.$$

To show that we can cover all possible values of M for the right choice of c , note that, for $c = 1$, we have

$$\frac{3}{2} \cdot \frac{n}{k(k-1)} \leq 2 \cdot \frac{n}{k(k-1)} \leq M$$

and, for two consecutive values c' and c'' (that is, $c'' = c' + 1$), we have

$$\frac{3c'n}{k(k-1)} \geq \frac{3nc''}{2k(k-1)} = \frac{3n(c'+1)}{2k(k-1)} \iff 2 \geq \frac{c'+1}{c'}$$

which obviously holds for any $c' \geq 1$. From (6) and (7), we immediately conclude

$$\mathbb{E}[B_s] \geq \frac{2ck - 2c}{M} - \frac{2c^2n}{M^2k} \geq \frac{k^3}{2n} - \frac{k(16k - 8 + k^2)}{18n} \geq \left(1 - \frac{1}{9}\right) \frac{k^3}{2n} - 1.$$

2. Now suppose $\frac{n-k+1}{k^2} < M \leq \frac{2n}{k(k-1)}$. Here, we do not have enough randomness to do wrap-arounds. Thus, fixing $c := 1$ in (6), we immediately get

$$\mathbb{E}[B_s] \geq \frac{2k - l - 1}{M} \geq \frac{2k - 2}{M} - \frac{2n}{M^2k}.$$

The claim follows immediately. $\square$

The intuitive idea behind the proof above is the following. With increasing M , as long as $M \leq \frac{2n}{k(k-1)}$, the gain of $\mathbf{R}$ increases, because we can choose from more and more deterministic strategies and shrink l . However, if M grows too much, the gain gets less and we start the wrap-around technique and thereby get a bound that does not depend on M anymore. If M increases over some threshold, another wrap-around is made and the intervals get decompressed a little. If M increases further, the intervals shrink until, for the some next threshold value, another wrap-around is made.

Up to this point, we restricted ourselves to even values for M . However, a simple observation resolves this issue.

Corollary 1. *For any M and any randomized algorithm $\mathbf{R} \in \mathcal{R}(M)$, the bounds from Theorem 3 hold up to a multiplicative factor of $1 - 1/M$.*

Proof. Theorem 3 holds for any even M . Now if M is odd, $\mathbf{R}$ acts as above for any random choice from 1 to $M - 1$. In any of the cases, the expected gain X is the same as in Theorem 3. If M is chosen, $\mathbf{R}$ may open some arbitrary boxes. In this case, we assume that the gain is zero. We therefore get a total expected gain of at least

$$\left(1 - \frac{1}{M}\right) \cdot X + \frac{1}{M} \cdot 0.$$

$\square$

Observe the two theorems above provide us with two sharp thresholds on the amount of randomness. If $M < (n + k - 1)/k^2$, this small amount of randomness does not help at all. On the other hand, if $M > 2n/(k(k-1))$, which corresponds to roughly $\log n - 2 \log k$ random bits, any further randomness does not help to improve the gain.

3 Randomized Algorithms with Advice

In this section, we present our main result by analyzing the trade-off between randomness and advice for (n, k) -BOXES. For this, we consider online algorithms that base their computation on both advice bits and randomness. In essence, we prove that, for the same amount of randomness, every additional advice bit allows us to find the same expected number of full boxes within a sequence of roughly double length. This implies that, for instance, the bound of $\frac{8}{9} \cdot \frac{k^3}{2n} - 1$ on the expected number of opened full boxes from the first claim of Theorem 3 can already be reached with a random number of roughly half the size.

To achieve this goal, we introduce the following notation. We denote by $F(n, k, M, b)$ the expected number of full boxes opened by the best algorithm that solves (n, k) -BOXES with randomness M and b bits of advice.

For $S \subseteq \{1, \dots, n - k + 1\}$, we generalize (n, k) -BOXES to (S, n, k) -BOXES. In (S, n, k) -BOXES, the starting position Adv chooses for the full boxes, has to be chosen from the set S , otherwise it is identical to (n, k) -BOXES. An algorithm that solves (S, n, k) -BOXES is called *faithful* if it opens only boxes whose positions are in S until the first full box is encountered.

Lemma 1. *For every algorithm $\mathbf{A}$ that solves (S, n, k) -BOXES with randomness M and b advice bits, there exists a faithful algorithm $\mathbf{A}'$ that also solves (S, n, k) -BOXES with randomness M and b advice bits such that $\mathbb{E}[B_s(\mathbf{A}')] \geq \mathbb{E}[B_s(\mathbf{A})]$, for all s .*

Proof. Suppose we are given $\mathbf{A}$ as stated by the lemma. The following strategy is carried out by $\mathbf{A}'$ until it finds the first full box: If $\mathbf{A}$ opens box b_i , then $\mathbf{A}'$ opens b_j with $j = \max\{s \in S \mid s \leq i\}$, i.e., the next smaller box that is in S , or it opens no box at all if this maximum does not exist. It is easy to see that $\mathbf{A}'$ opens its first full box not later than $\mathbf{A}$: Assume $\mathbf{A}$ opens the first full box b_i , then $b_p, b_{p+1}, \dots, b_i$ are all full, where p is the starting position of the sequence chosen by Adv . Of course, $p \in S$ and $p \leq j \leq i$. Afterwards, $\mathbf{A}'$ opens the same boxes as $\mathbf{A}$, except for the case where $\mathbf{A}$ wants to open the box b_j which was already opened by $\mathbf{A}'$. In this case, $\mathbf{A}'$ opens b_i .

In that way, $\mathbf{A}'$ opens at least the same number of full boxes as $\mathbf{A}$. $\square$

Lemma 2. *Let $\mathbf{A}$ be an algorithm that solves (S, n, k) -BOXES with randomness M and no advice. Then*

$$\min_s \mathbb{E}[B_s(\mathbf{A})] \leq F(|S| + k - 1, k, M, 0) + 1.$$

Proof. Due to Lemma 1, we can assume that $\mathbf{A}$ is faithful without loss of generality. We now convert $\mathbf{A}$ into an algorithm $\mathbf{A}'$ that solves $(|S| + k - 1, k)$ -BOXES with randomness M and 0 advice bits such that

$$\min_s \mathbb{E}[B_s(\mathbf{A}')] \geq \min_s \mathbb{E}[B_s(\mathbf{A})] - 1.$$

Let $S = \{p_1, \dots, p_{|S|}\}$ with $p_1 < p_2 < \dots < p_{|S|}$. Let us further assume that $\mathbf{A}$ would open boxes at positions $p_{i_1}, \dots, p_{i_k}$ if we would report them all as

empty. Then A' opens boxes at positions $i_1, \dots, i_r$ until it finds the first full box b_{i_r} ; afterwards A' continues with local search.

Let i be a worst-case starting position for A' . As we have already discussed, the last $k - 1$ boxes cannot be starting positions. We therefore have $i \leq |S| + k - 1 - (k - 1) = |S|$. We choose p_i as the starting position for A . If A' does not open a full box in the first r rounds, then neither does A , because A is faithful, and therefore only opens boxes from S . By construction, any full box it finds this way corresponds to a full box A' opens. $\square$

We are now ready to prove the main claims of this section.

Theorem 4. $F(n, k, M, b) \leq F(\lceil (n - k + 1)/2^b \rceil + k - 1, k, M, 0) + 1$.

Proof. If an algorithm A solves (n, k) -BOXES with b advice bits, then, by the pigeon-hole principle, there is at least one advice string that is used for at least $\lceil (n - k + 1)/2^b \rceil$ different starting positions. Let S be the set of those starting positions. Then the algorithm A can be used to solve (S, n, k) -BOXES without advice (but with randomness M). Hence, if A is optimal, then $F(n, k, M, b) \leq \min_s \mathbb{E}[B_s(A)] \leq F(|S| + k - 1, k, M, 0) + 1$ by Lemma 2. Clearly, $F(n, k, M, b)$ is anti-monotone in n (if k, M , and b are fixed, Adv obtains more positions to hide the boxes with growing n) and thus $F(|S| + k - 1, k, M, 0) \leq F(\lceil (n - k + 1)/2^b \rceil + k - 1, k, M, 0)$. $\square$

Theorem 5. $F(n, k, M, b) \geq F(\lceil (n - k + 1)/2^b \rceil + k - 1, k, M, 0)$.

Proof. Consider an algorithm A for (n, k) -BOXES that uses b bits of advice. Again, there are $n - k + 1$ possible starting positions for Adv . On the other hand, we may subdivide these boxes into 2^b groups of size

$$\left\lceil \frac{n - k + 1}{2^b} \right\rceil,$$

and encode the position of the group that contains the starting position using b bits. Easily, we can extend this interval by $k - 1$ boxes (which cannot contain a starting position). Then, A simulates an optimal algorithm A' for an instance of this size. It directly follows that A gains at least as much as A' . $\square$

Combining the results from Section 2 with Theorems 4 and 5 immediately yields the following upper and lower bounds on the expected number of opened full boxes for randomized algorithms with advice.

Corollary 2.

- (a) If $M < \lceil \frac{n-k+1}{2^b} \rceil / k^2$, then $F(n, k, M, b) \leq 1$.
- (b) For any M ,

$$F(n, k, M, b) \leq \frac{k^2(k+1)}{2^{\lceil \frac{n-k+1}{2^b} \rceil}} + 1.$$

Fig. 4: The expected gain using randomness M and b bits of advice for $n = 1\,000\,000$, $k = 300$, different values of M and up to two advice bits.

(c) If $M = c \lceil \frac{n-k+1}{2^b} \rceil / k^2$ with $c \in [1, 2k/(k+3)]$, then

$$F(n, k, M, b) \leq \frac{2(k + \frac{3}{2})}{M} - \frac{2 \lceil \frac{n-k+1}{2^b} \rceil}{M^2 k} + 1 \quad \text{and}$$

$$F(n, k, M, b) \leq \frac{2(c-1)}{c^2} \cdot \frac{k^3}{\lceil \frac{n-k+1}{2^b} \rceil} + \frac{3k^2}{c \cdot \lceil \frac{n-k+1}{2^b} \rceil} + 1.$$

Corollary 3. Let M be an even number.

(a) If $M \geq 2(\lceil \frac{n-k+1}{2^b} \rceil + k - 1)/(k(k-1))$, then

$$F(n, k, M, b) \geq \frac{8}{9} \cdot \frac{k^3}{2 \cdot \lceil \frac{n-k+1}{2^b} \rceil + k - 1} - 1.$$

(b) If $\lceil \frac{n-k+1}{2^b} \rceil \cdot \frac{1}{k^2} < M < 2(\lceil \frac{n-k+1}{2^b} \rceil + k - 1)/(k(k-1))$, then

$$F(n, k, M, b) \geq \frac{2k-2}{M} - \frac{2 \cdot \lceil \frac{n-k+1}{2^b} \rceil + 2k-2}{M^2 k}.$$

Note that these upper and lower bounds are almost tight. Obviously, Corollary 3 can easily be extended to the case of odd M using Corollary 1. A graphical illustration of the connection between the amount of randomness, the number of advice bits and the expected number of full boxes that are opened is given in Fig. 4.

4 Conclusion

In this paper, we have analyzed, for the first time, the trade-off between randomness and advice bits for online computation. We gave matching or almost matching upper and lower bounds on the competitive ratio for (n, k) -BOXES for any combination of randomness and advice bits. A goal for further research is to extend these results to a broader class of online problems.

References

1. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Kráľovič, and Richard Kráľovič. On the advice complexity of the k -server problem. In *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6755 of *Lecture Notes in Computer Science*, pages 207–218, Springer-Verlag, 2011.
2. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Kráľovič, Richard Kráľovič, and Tobias Mömke. On the advice complexity of online problems. In *Proc. of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, volume 5878 of *Lecture Notes in Computer Science*, pages 331–340. Springer-Verlag, 2009.
3. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
4. Stefan Dobrev, Rastislav Kráľovič, and Dana 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)*, volume 4910 of *Lecture Notes in Computer Science*, pages 247–258. Springer-Verlag, 2008.
5. Stefan Dobrev, Rastislav Kráľovič, and Dana Pardubská. Measuring the problem-relevant information in input. *Theoretical Informatics and Applications (RAIRO)*, 43(3):585–613, 2009.
6. Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theoretical Computer Science*, 412(24):2642–2656, 2011.
7. Amos Fiat and Gerhard J. Woeginger, editors. *Online Algorithms, The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
8. Juraj Hromkovič. *Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms*. Springer-Verlag, 2005.
9. Juraj Hromkovič, Rastislav Kráľovič, and Richard Kráľovič. Information complexity of online problems. In *Proc. of the 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010)*, volume 6281 of *Lecture Notes in Computer Science*, pages 24–36. Springer-Verlag, 2010.
10. Sandy Irani and Anna R. Karlin. On online computation. In *Approximation Algorithms for NP-Hard Problems*, chapter 13, pages 521–564, 1997.
11. Dennis Komm and Richard Kráľovič. Advice complexity and barely random algorithms. *Theoretical Informatics and Applications (RAIRO)*, 45(2):249–267, 2011.
12. Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.