The Complexity of Paging Against a Probabilistic Adversary

Stefan Dobrev[1], Juraj Hromkovič[2], Dennis Komm[2](✉), Richard Královič[3], Rastislav Královič[4], and Tobias Mömke[5]

[1] Mathematical Institute, Slovak Academy of Sciences, Bratislava, Slovakia
stefan.dobrev@savba.sk
[2] Department of Computer Science, ETH Zürich, Zurich, Switzerland
{juraj.hromkovic,dennis.komm}@inf.ethz.ch
[3] Google Inc., Zurich, Switzerland
richard.kralovic@dcs.fmph.uniba.sk
[4] Department of Computer Science, Comenius University, Bratislava, Slovakia
kralovic@dcs.fmph.uniba.sk
[5] Saarland University, Saarbrücken, Germany
moemke@cs.uni-saarland.de

Abstract. We consider deterministic online algorithms for paging. The offline version of the paging problem, in which the whole input is given in advance, is known to be easily solvable. If the input is random, chosen according to some known probability distribution, an $\mathcal{O}(\log k)$-competitive algorithm exists. Moreover, there are distributions, where no algorithm can be better than $\Omega(\log k)$-competitive.

In this paper, we ask the question of what happens if it is known that the input is one from a set of ℓ potential candidates, chosen according to some probability distribution. We present an $\mathcal{O}(\log \ell)$-competitive algorithm, and show a matching lower bound.

1 Introduction

In algorithmics, that is, the "study of algorithms" [12], one is concerned with constructing and analyzing algorithms for given computing problems that perform well with respect to some given constraints, for instance, being efficient or obtaining some specific solution quality. In a classical setup, an input is given to an algorithm, and some particular information needs to be extracted. This is usually done while having full knowledge about the input. For instance, the information that needs to be extracted may be a cheapest Hamiltonian cycle that is "hidden" in an instance that corresponds to a complete weighted graph. Many computing problems, however, are what is called "intrinsically online" which means that the whole input is not known in advance, but arrives gradually in consecutive time steps while a part of the definite output already needs

The research is partially funded by SNF grant 200021–146372, VEGA grant 1/0979/12, and Deutsche Forschungsgemeinschaft grant BL511/10-1.

to be created. Typical members of this class are scheduling or packing problems, and various kinds of resource management problems.

Such problems are called "online problems" [1,6,15,22], and they are found in many real-world situations. One of the most prominent and well-understood online problems is the paging (caching) problem. Here, an online algorithm maintains a cache containing up to k logical pages out of m possible ones. The input is a sequence of n requests for logical pages, and the algorithm has to process each request: if the requested page is in the cache (this is called a cache hit), nothing happens; if not (which is called a cache miss, or page fault), the algorithm has to evict one page from the cache, and replace it with the requested one. The goal of the algorithm is to process the input while minimizing the number of page faults. Let us give a formal definition; for the ease of presentation, we identify pages with their indices.

Definition 1 (Paging Problem). *An instance of the paging problem is a sequence of integers representing requests to logical pages $I = (x_1, x_2, \dots, x_n)$, $x_i > 0$. An online algorithm* Alg *maintains a buffer (content of the physical memory) $B = \{b_1, b_2, \dots, b_k\}$ of k integers, where k is a fixed constant known to* Alg*. Before processing the first request, the buffer gets initialized as $B = \{1, 2, \dots, k\}$. Upon receiving a request x_i, if $x_i \in B$, then* Alg *creates the partial output $y_i = 0$. If $x_i \notin B$, then a* page fault *occurs, and* Alg *has to find some* victim b_j*, that is, $B := B \setminus \{b_j\} \cup \{x_i\}$, and $y_i = b_j$. The cost of the solution* Alg(I) *is the number of page faults, that is,* $\mathrm{cost}(\textsc{Alg}(I)) = |\{i \mid y_i > 0\}|$.

In this paper, we consider deterministic online algorithms for the paging problem. The performance of an algorithm is measured by the *competitive ratio* (which basically is the online counterpart of the *approximation ratio*[1] for offline problems) where the cost of the algorithm on a particular input I is compared to an optimal solution for I. In a very general setting, one may consider the inputs to come from a probability distribution ρ over all possible inputs. The quality of a solution depends on the *data model* that specifies two orthogonal aspects of the setting. First, the data model specifies the class of possible input distributions $\mathcal{P}$; an algorithm Alg is called *c-competitive* if there is a constant α, such that

$$\forall \rho \in \mathcal{P}\colon \mathbb{E}_\rho\left[\frac{\mathrm{cost}(\textsc{Alg}(I)) - \alpha}{\mathrm{cost}(\textsc{Opt}(I))}\right] \leq c \tag{1}$$

where the instance I is taken according to the distribution ρ. Second, the data model specifies what information about the distribution ρ is known to the algorithm.

When $\mathcal{P}$ is the class of all point mass distributions (that is, distributions where one particular input has probability 1), the impact of the algorithm's knowledge has been widely studied. One extreme case is when the whole point mass distribution (that is, the input) is known to the algorithm; this corresponds to the

[1] Note that unlike offline algorithms, in an online setting we usually ignore the running time of the algorithm.

offline deterministic case, which is easily solvable by a greedy algorithm (LFD, longest forward distance, called Min by Bélády [2] who first proved its optimality). The other extreme case, when the point mass distribution is unknown, corresponds to the online deterministic worst-case scenario, and it is known [21] that no deterministic online algorithm can be better than k-competitive. The spectrum between these two extremes has been studied by means of advice complexity [4,5,11,14], which was first studied for paging by Dobrev et al. [10].

Orthogonally, in what is known as *distributional* approach [6] to the analysis of online algorithms, it is supposed that the inputs come from a fixed distribution (that is, $\mathcal{P}$ is a singleton, in our setting), which usually is a uniform distribution. Franaszek and Wagner [13] studied the particular input distribution where each request is selected independently from a given distribution. Note that this was done before competitive analysis was introduced by Sleator and Tarjan [21]. In the *Markov paging* model by Shedler and Tung [20], the inputs are generated by a Markov chain. Pandurangan and Upfal [18] studied a setting where the page requests come from a stochastic process with given entropy.

There are also approaches where the worst case from several distributions is analyzed. Notably, in the *access graph* model by Borodin et al. [7], $\mathcal{P}$ is the set of point mass distributions corresponding to walks in a (known) graph G and the particular distribution ρ is, of course, unknown. The *statistical adversary* introduced by Raghavan [19] considers (in the role of $\mathcal{P}$ in our notation) the class of all point mass distributions that fulfill certain statistical properties, for instance, each page is requested the same number of times. This model was later sucessfully applied to two-way currency trading by Chou et al. [9].

Our setting follows the general notion of the *diffuse adversary* introduced by Koutsoupias and Papadimitriou [17] with the only distinction that in the diffuse adversary model, the particular distribution is always unknown. The diffuse adversary is a generalization of the statistical adversary (and other models that have been introduced in the past). In 1998, Young further studied the diffuse adversary [25]; he gave both tight bounds (up to a factor of 2) for deterministic and randomized online algorithms in this setting.

As pointed out before [1,3,8,15,17,23], we argue that both extreme cases, that is, either knowing *nothing* about the input, or knowing *everything* about it, are unrealistic. As the requested pages depend on the user's actions, it is clearly impossible for any operating system to completely foresee which pages will be accessed in the future. On the other hand, it seems equally unrealistic to assume that *every* input sequence is possible.

In Sect. 2, we briefly state our contribution, and put it into context. We formally state and prove our results in Sect. 3; we conclude in Sect. 4. Throughout this paper, log denotes the logarithm with base 2.

2 Our Contribution

On one hand, a known point mass distribution corresponds to the offline case, and it is easily solvable as already mentioned [6]. Also, it follows from Yao's principle [24] and the results on randomized paging, that, for any known point mass distribution, there is an $\mathcal{O}(\log k)$-competitive algorithm, and there are such distributions where any algorithm is at least $\Omega(\log k)$-competitive. Moreover, as shown by Komm and Královič [16], it follows that, for any point mass distribution, there is an $\mathcal{O}(\log k)$-bit long binary string from which the $\mathcal{O}(\log k)$-competitive algorithm can be efficiently decoded.

Motivated by this, we analyze known distributions that are somewhat "close" to being point mass: instead of the probability mass being concentrated in one input, it can be distributed arbitrarily among ℓ inputs.

Definition 2 (Class $\mathcal{P}_\ell$ of Distributions). *By $\mathcal{P}_\ell$ we denote the class of* ℓ-point distributions*, that is, probability distributions over inputs such that at most ℓ inputs have non-zero probability.*

An online algorithm for paging that only evicts pages in case of a page fault is called a "demand paging" (for k-server, a generalization of paging, the term "lazy" is more common) algorithm. The laziness requirement comes with no loss of generality [6], hence we shall consider only lazy algorithms to obtain easier arguments. Note that we incorporated this already in our formal definition of paging that only allows an algorithm to evict a page if a page fault occurs.

We analyze the expected competitive ratio of deterministic paging algorithms over a known ℓ-point distribution. We show that, if ℓ is constant, there is an "almost optimal" (that is, 1-competitive) algorithm. Basically, we prove that, since ℓ is fixed, the overhead the algorithm pays until it realizes which input it is working on, can be hidden in the additive constant α from the definition of the competitive ratio. For the *strict* competitive ratio (that is, when demanding that $\alpha = 0$) we show an $\mathcal{O}(\log \ell)$-competitive algorithm for $\ell < k$ (for $\ell \geq k$, one can use the $\mathcal{O}(\log k)$-competitive algorithm). We complement the result by a matching lower bound stating that no online algorithm can be better than $(\log \ell)/2$-competitive.

3 Results

We start with the non-strict case, that is, the case where the additive constant α from the definition of the competitive ratio may be strictly positive. From the order of the quantifiers in (1), it follows that α may depend on the cache size k, and the class of possible distributions (in our case parametrized by ℓ). In this case, we can prove the following theorem.

Theorem 1. *There is a* 1*-competitive paging algorithm for any known distribution from the class $\mathcal{P}_\ell$, for any constant ℓ.*

Before presenting an online algorithm that obtains this bound, let us make the following observation.

Lemma 1. *Consider two inputs I_1 and I_2. Let* $\textsc{Opt}_1$ *and* $\textsc{Opt}_2$ *be the optimal (LFD) algorithms for I_1 and I_2, respectively. Then* $\textsc{Opt}_1$ *and* $\textsc{Opt}_2$ *make the same number of page faults on the common prefix of I_1 and I_2.*

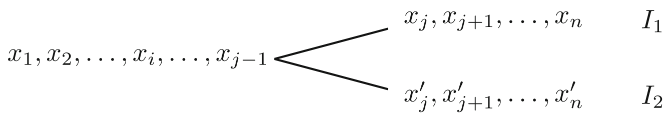
$x_1, x_2, \ldots, x_i, \ldots, x_{j-1}$
$x_j, x_{j+1}, \ldots, x_n$
I_1
$x'_j, x'_{j+1}, \ldots, x'_n$
I_2

Fig. 1. I_1, I_2, and the branching point

Proof. Let j be the first position where I_1, and I_2 differ (see Fig. 1). To prove the claim by contradiction, suppose that there is a page requested in the prefix $x_1, x_2, \ldots, x_{j-1}$ such that it causes a page fault for Opt_1 but not for Opt_2; let x_i be the first page with this property. Opt_1 evicted this page in some preceding time step as it was requested farthest in the future, namely in time step i. But since this happened on the common prefix of I_1 and I_2 (that is, $i \leq j-1$), Opt_2 would have taken the same action. □

We now prove Theorem 1 using Lemma 1.

Proof (of Theorem 1). Let $\rho \in \mathcal{P}_\ell$ be the input distribution, and let $I_1, I_2, \ldots, I_\ell$ denote the inputs that are chosen with non-zero probability sorted in non-increasing order $\rho(I_1) \geq \rho(I_2) \geq \cdots \geq \rho(I_\ell)$. Let Opt_i be the optimum (LFD) solution for I_i, $1 \leq i \leq \ell$.

The algorithm Alg starts by simulating Opt_1. If the actual input is I_1, Alg is optimal. If, at some point, Alg realizes that the instance is not I_1, Alg switches to Opt_2: it replaces the cache by the pages that would have been in the cache of Opt_2 at this moment, and continues as Opt_2 (actually, since we are considering lazy algorithms exclusively, Alg only remembers the state, and replaces the pages as needed). Let j be the time step where I_1 and I_2 differ for the first time; and thus, if Alg is not optimal on I_2, the actions of Opt_1 and Opt_2 differ in some time step $i < j$.

From Lemma 1 it follows that, after Alg switched from Opt_1 to Opt_2, the number of page faults so far was the same as in Opt_2 plus the at most k faults needed to change the cache content to be consistent with Opt_2.

This process can be iterated: Alg simulates Opt_2 until it sees a difference in the input, switches to Opt_3, and so on. After any switch to Opt_d, the cost of Alg so far is bounded by the cost of Opt_d plus at most kd page faults needed to replace the cache after each switch. Overall, since there are at most ℓ switches, we have

$$\text{cost}(\text{Alg}) \leq \text{cost}(\text{Opt}) + k\ell.$$

Since this inequality holds for any execution, (1) holds for $\alpha = k\ell$, which finishes the proof. □

The previous theorem asserts that the price of each switch of the algorithm Alg is at most k. If we consider the *strict* competitive ratio, in which $\alpha = 0$ in (1), this price is too high, since for any known distribution there is a $\mathcal{O}(\log k)$-competitive algorithm. In the following theorem, we present a more detailed accounting of the expected price of switching. In the proof, we again make use of Lemma 1.

Theorem 2. *For any constant ℓ, there is an online paging algorithm for any known distribution from the class $\mathcal{P}_\ell$ with an expected strict competitive ratio of at most* $\ln \ell + 1$.

Proof. Let $\rho \in \mathcal{P}_\ell$, and let $\mathcal{I} := \{I_1, I_2, \ldots, I_\ell\}$ be the instances that are chosen with non-zero probability. ALG computes the prefix tree T of $\mathcal{I}$ with root r. Note that in T there is exactly one node for each distinct prefix in any of the request strings in $\mathcal{I}$, and any instance is represented by a path from r to a distinct leaf of T. Each node of T that is not a leaf is labeled by its requested page, that is, the page that distinguishes its prefix from its predecessor's prefix. The root and the leaves obtain an empty label. We define $N^+(v)$ to be the out-neighborhood of a node v, that is, the children of v.

For each node v of T, we define a probability $p(v)$ inductively. If v is a leaf, then there is a exactly one instance $I_v \in \mathcal{I}$ that corresponds to a path from r to v and we set $p(v) = \rho(I_v)$. If v is no leaf but all vertices in $N^+(v)$ already have been assigned probabilities, we set

$$p(v) = \sum_{w \in N^+(v)} p(w).$$

Clearly, eventually all nodes are labeled and $p(r) = 1$.

We now identify a subgraph $\mathcal{T}$ of T as follows. Initially, $\mathcal{T}$ has all nodes of T but no arcs. Then for each node v, we introduce exactly one arc (v, w), where

$$w = \operatorname*{arg\,max}_{w' \in N^+(v)} p(w'),$$

breaking ties arbitrarily. Note that $\mathcal{T}$ is a collection of directed paths (possibly of length 0) that end in leaves of T. For each node v, denote by P_v the suffix of the path in $\mathcal{T}$ that contains v where v is the start vertex of P_v. Then the strategy of ALG is to move within T according to the requests and to follow the LFD strategy of the instance defined by the labels of P_v, where v is the currently visited node. Note that after requests the strategy may change. For a leaf w, OPT$_w$ is the optimal LFD solution for the instance defined by the path from r to w. To estimate the impact of strategy changes, we use the following claim that follows by applying Lemma 1 to all pairs of vertices that are reachable from a given vertex.

Claim. Let v be a node of T, and let S be the set of leaves reachable from v. Then the number of page faults on the prefix of instances defined by the path from r to v are identical for all solutions OPT$_w$ with $w \in S$.

The claim allows us to estimate the cost of changing strategies. For each pair of leaves w, w', there is a vertex v such that the paths from r to w and from r to w' fork at v. There is a *critical phase* from where OPT$_w$ and OPT$_{w'}$ differ first until v, and there is some number κ of differences between the two solutions within the critical phase. Therefore, changing the strategy from OPT$_w$ to OPT$_{w'}$ causes at most κ page faults and $\kappa \leq \text{cost}(\text{OPT}_{w'})$. As a consequence, the number of different strategies is an upper bound on the attained competitive ratio. In the remaining analysis, we give an upper bound on the expected number of strategy changes.

Let us fix an internal node v with $s := |N^+(v)| > 0$, that is, v is not a leaf. Furthermore, let w be the subsequent node in P_v. For each $w' \in N^+(v)$ we define

$$p'(w') = \frac{p(w')}{\sum_{w'' \in N^+(v)} p(w'')}.$$

In other words, $p'(w')$ is the probability that the given path continues with w', provided that it contains v. Then the probability to change the strategy after v is $1 - p'(w)$. We now give an upper bound on the fraction of reachable leaves left after leaving v.

For any node w', let leaves(w') be the set of leaves reachable from w'. Then, for each $w' \in N^+(v)$, we define the fraction of leaves reachable from v via w' by

$$\gamma(v, w') := \frac{|\text{leaves}(w')|}{\sum_{w'' \in N^+(v)} |\text{leaves}(w'')|}.$$

Then the expected fraction of leaves left after leaving v is

$$\sum_{w' \in N^+(v)} p'(w')\gamma(v, w').$$

This number is maximized if $\gamma(v, w) = 1$ and $\gamma(v, w') = 0$ for all $w' \neq w$. We obtain the upper bound $\gamma(v, w) \leq p'(w)$.

Therefore, we obtain an upper bound on the total number of strategy changes if we consider an integer t and a sequence $(p_i)_{i=1}^t$ of probabilities such that

$$\sum_{i=1}^{t} (1 - p_i)$$

is maximized subject to

$$\prod_i p_i \geq 1/\ell.$$

The meaning of the objective is that there are t nodes with probabilistic decisions. The probability of the selected strategy at the i-th node is p_i and thus there is a strategy change with probability $1 - p_i$. Intuitively, the maximum is attained when both the length t of the sequence is long and the probabilities to change strategies are large. However, for increasing values of t, the constraints enforce that most of the $1 - p_i$ are small. The constraints stem from the fact that there is no leaf left if the fraction of remaining leaves is smaller than $1/\ell$.

We claim that the maximum can be attained with $p_i = p_{i'}$ for all pairs of indices i, i'. Suppose towards contradiction that there is no maximal solution with this property. Let $p_1, p_2, \ldots, p_t$ be a solution attaining the maximum such that

$$\mu := \max_{i,i'} |p_i - p_{i'}|$$

is minimal and among these solutions one where

$$|\{(i,i') \mid |p_i - p_{i'}| = \mu\}|$$

is minimal. Let us fix two indices $\hat{\imath}, \hat{\imath}'$ such that $|p_{\hat{\imath}} - p_{\hat{\imath}'}| = \mu$. Now let us consider the solution $p'_1, p'_2, \ldots, p'_t$ where $p'_i = p_i$ for all indices except $\hat{\imath}$ and $\hat{\imath}'$ and where

$$p'_{\hat{\imath}} = p'_{\hat{\imath}'} = \frac{p_{\hat{\imath}} + p_{\hat{\imath}'}}{2}.$$

Clearly, still $\max_{i,i'} |p'_i - p'_{i'}| \le \mu$. Also, the value of the objective function did not change since $p_{\hat{\imath}} + p_{\hat{\imath}'} = p'_{\hat{\imath}} + p'_{\hat{\imath}'}$. To show that the constraints are satisfied, we claim that

$$p_{\hat{\imath}} p_{\hat{\imath}'} \le p'_{\hat{\imath}} p'_{\hat{\imath}'}.$$

By renaming the indices, we assume without loss of generality that $p_{\hat{\imath}} \le p_{\hat{\imath}'}$. Then we have

$$\begin{aligned} p'_{\hat{\imath}} p'_{\hat{\imath}'} &= \left(\frac{p_{\hat{\imath}} + p_{\hat{\imath}'}}{2}\right)^2 \\ &= \frac{p_{\hat{\imath}}^2 + 2p_{\hat{\imath}}p_{\hat{\imath}'} + p_{\hat{\imath}'}^2}{4} \\ &= p_{\hat{\imath}}p_{\hat{\imath}'} + \frac{p_{\hat{\imath}}^2}{4} - \frac{p_{\hat{\imath}}(p_{\hat{\imath}} + \delta)}{2} + \frac{(p_{\hat{\imath}} + \delta)^2}{4} \\ &= p_{\hat{\imath}}p_{\hat{\imath}'} + \frac{\delta^2}{4} \end{aligned}$$

where $\delta = p_{\hat{\imath}'} - p_{\hat{\imath}}$. Therefore, unless $\delta = 0$,

$$|\{(i,i') \mid |p'_i - p'_{i'}| = \mu\}| < |\{(i,i') \mid |p_i - p_{i'}| = \mu\}|,$$

which is a contradiction to the minimality of $|\{(i,i') \mid |p_i - p_{i'}| = \mu\}|$. Hence, from now on we may assume that $p_i = p$ for some probability p and all indices i. We obtain

$$t = \log_p p^t = \log_p(1/\ell) = \frac{\ln \ell}{\ln(1/p)}.$$

As a consequence, we have to find

$$\max_p \frac{(1-p)\ln \ell}{\ln(1/p)}.$$

We have

$$\frac{\partial}{\partial p} \frac{(1-p)\ln \ell}{\ln(1/p)} = \ln \ell \frac{\ln p - 1 + 1/p}{\ln^2 p}.$$

For $p \in (0,1)$, the derivative is always positive since $1/p > 1$. Thus,

$$\frac{(1-p)\ln \ell}{\ln(1/p)}$$

is monotonously increasing in p and the maximum is attained for $p \to 1$. On a high level this means that t is large whereas the probability of strategy changes, $1-p$, is small. Now the number of strategy changes is bounded from above by

$$\lim_{p\to 1} \frac{(1-p)\ln \ell}{\ln(1/p)} = \lim_{p\to 1} \frac{-\ln \ell}{-1/p} = \ln \ell$$

where we used l'Hôspital's rule. With our previous discussion, the statement of the theorem follows. $\square$

Finally, we argue that the algorithm from Theorem 2 is in a sense best possible by proving the following lower bound.

Theorem 3. *Any online algorithm* Alg *on the class $\mathcal{P}_\ell$ with $\ell \le k$ has an expected strict competitive ratio of at least* $(\log \ell)/2$.

Proof. First, assume that ℓ is a power of 2; without loss of generality, let Alg be a demand paging (that is, lazy) algorithm. We assume that the cache is always organized such that the page indices of all pages residing in the cache at any given point in time are in increasing order in every time step; we do this without loss of generality and to keep our arguments simple and not being forced to argue about permutations of the cache cells. We describe a class $\mathcal{I}$ of ℓ instances, and the probability distribution ρ will be a uniform distribution over $\mathcal{I}$. Let us assume that initially the cache contains pages $1, 2, \ldots, k$.

All instances in $\mathcal{I}$ start by introducing a page $k+1$, and particular instances can be described by a number i, such that in I_i, the optimal algorithm evicts page i from the cache in the first time step. For all I_i, the optimal cost will be one, so this is the only page fault the optimal algorithm incurs. In each instance, the first request is followed by $\log \ell$ rounds, and we show that any algorithm causes a page fault in every round with probability at least 1/2. Together with the one page fault in the first time step, this gives the expected number of faults, and also the expected competitive ratio.

Consider any algorithm Alg running on an instance I_i. Every round starts and ends by a sequence requesting pages $\ell+1, \ell+2, \ldots, k$; if these pages are not in the cache of Alg at the beginning of the round, Alg makes a page fault with probability 1, and we are done. Hence, we can assume that Alg never evicts pages from the range $\ell+1, \ell+2, \ldots, k$ from the cache. Let us call the pages $1, 2, \ldots, \ell$ *active*.

For round 2, the active pages are partitioned into two halves, and all pages of that half that does not contain i are requested consecutively (see Fig. 2). Clearly, Alg makes another page fault with probability at least 1/2. This procedure is applied recursively to the half that contains i until, in round $\log \ell + 1$, there are only two halves that contain single pages (out of which one is the page i). Moreover, in every round, all pages that were requested in the previous round are requested again.

Initial cache:		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Round 1:	17																
Round 2:	17									9	10	11	12	13	14	15	16
Round 3:	17	1	2	3	4					9	10	11	12	13	14	15	16
Round 4:	17	1	2	3	4			7	8	9	10	11	12	13	14	15	16
Round 5:	17	1	2	3	4		6	7	8	9	10	11	12	13	14	15	16

Fig. 2. Example for $k = \ell = 16$; the optimal solution removes page 5 in time step one, when page 17 is requested; after that, there are 4 rounds that each consist of requests that were in the cache of Opt at the beginning

It follows that, in each round r, $2 \leq r \leq \log \ell + 1$, Alg makes a page fault with probability at least 1/2 (note that the corresponding events are all independent) and an additional page fault in round 1 with probability 1. At the same time, Opt makes exactly 1 page fault in total. Summing up, the ratio of the costs is $(\log \ell)/2 + 1$.

Finally, assume that ℓ is not a power of 2. Let ℓ' denote the largest power of 2 that is smaller than ℓ. We follow the exact same strategy as above with ℓ' instead of ℓ; in particular, we request all pages $\ell' + 1, \ell' + 2, \ldots, k$ in every round. By definition, we have $\ell' > \ell/2$, thus there is at most one less round, and consequently the lower bound decreases by at most 1. □

4 Conclusion

We studied the case of paging against a known distribution. On one hand, there are distributions where no algorithm can perform better than being $\Omega(\log k)$-competitive; on the other hand, for a point mass distribution, an easy optimal algorithm exists. We addressed the general question of characterizing the distributions in terms of the complexity of paging algorithms for them. We showed that if the distribution has at most ℓ inputs that are chosen with non-zero probability each, there is an $(\ln \ell + 1)$-competitive online algorithm. Complementing, by constructing a class of hard instances, we showed that this bound is tight up to a small factor when $\ell \leq k$. In the case that the additive constant α from the competitive ratio is allowed to be positive, there is a simple 1-competitive online algorithm where $\alpha = k\ell$.

Acknowledgement. The authors would like to thank Hans-Joachim Böckenhauer for very valuable discussions.

References

1. Albers, S.: Online algorithms: a survey. Math. Program. **97**(1), 3–26 (2003)
2. Bélády, L.A.: A study of replacement algorithms for virtual-storage computer. IBM Syst. J. **5**(2), 78–101 (1966)
3. Ben-David, S., Borodin, A.: A new measure for the study of on-line algorithms. Algorithmica **11**(1), 73–91 (1994)
4. 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)
5. 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.) Automata, Languages and Programming. LNCS, vol. 6755, pp. 207–218. Springer, Heidelberg (2011)
6. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, New York (1998)
7. Borodin, A., Irani, S., Raghavan, P., Schieber, B.: Competitive paging with locality of reference. J. Comput. Syst. Sci. **50**(2), 244–258 (1995)
8. Boyar, J., Larsen, K.S., Nielsen, M.N.: The accommodating function: a generalization of the competitive ratio. SIAM J. Comput. **31**(1), 233–258 (2001)
9. Chou, A., Cooperstock, J., El-Yaniv, R., Klugerman, M., Leighton, T.: The statistical adversary allows optimal money-making trading strategies. In: Proceeding of SODA 1995, pp. 467–476. Society for Industrial and Applied Mathematics (1995)
10. 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)
11. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theor. Comput. Sci. **412**(24), 2642–2656 (2011)
12. Harel, D., Feldman, Y.: Algorithmics: The Spirit of Computing. Addison-Wesley, 3rd edn (2004)
13. Franaszek, P.A., Wagner, T.J.: Some distribution-free aspects of paging algorithm performance. J. ACM **21**(1), 31–39 (1974)
14. 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)
15. Irani, S., Karlin, A.R.: On online computation. In: Hochbaum, D.S. (ed.) Approximation Algorithms for NP-hard Problems, pp. 521–564. PWS Publishing Company (1997)
16. Komm, D., Královič, R.: Advice complexity and barely random algorithms. Theor. Inf. Appl. (RAIRO) **45**(2), 249–267 (2011)
17. Koutsoupias, E., Papadimitriou, C.H.: Beyond competitive analysis. SIAM J. Comput. **30**(1), 300–317 (2000)
18. Pandurangan, G., Upfal, E.: Entropy-based bounds for online algorithms. ACM Trans. Algorithms **3**(1), 1–19 (2007)
19. Raghavan, P.: A statistical adversary for on-line algorithms. DIMACS **7**, 79–83 (1991)
20. Shedler, G.S., Tung, C.: Locality in page reference strings. SIAM J. Comput. **1**, 218–241 (1972)

21. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985)
22. Fiat, A. (ed.): Online Algorithms 1996. LNCS, vol. 1442. Springer, Heidelberg (1998)
23. Fiat, A., Woeginger, G.J.: Competitive odds and ends. In: Fiat, A., Woeginger, G.J. (eds.) Online Algorithms 1996. LNCS, vol. 1442, pp. 385–394. Springer, Heidelberg (1998)
24. Yao, A.C.-C.: Probabilistic computations: Toward a unified measure of complexity (extended abstract). In: Proceeding of FOCS 1977, pp. 222–227. IEEE Computer Society (1977)
25. Young, N.E.: Bounding the diffuse adversary. In: Proceeding of SODA 1998, pp. 420–425. Society for Industrial and Applied Mathematics (1998)