Paging with Succinct Predictions

Antonios Antoniadis [*1] **Joan Boyar** [*2] **Marek Eliáš** [*3] **Lene M. Favrholdt** [*2] **Ruben Hoeksma** [*1]
Kim S. Larsen [*2] **Adam Polak** [*4] **Bertrand Simon** [*5]

Abstract

Paging is a prototypical problem in the area of online algorithms. It has also played a central role in the development of learning-augmented algorithms. Previous work on learning-augmented paging has investigated predictions on (i) when the current page will be requested again (*re-occurrence predictions*), (ii) the current state of the cache in an optimal algorithm (*state predictions*), (iii) all requests until the current page gets requested again, and (iv) the relative order in which pages are requested. We study learning-augmented paging from the new perspective of requiring the least possible amount of predicted information. More specifically, the predictions obtained alongside each page request are limited to one bit only. We develop algorithms satisfy all three desirable properties of learning-augmented algorithms – that is, they are consistent, robust and smooth – despite being limited to a one-bit prediction per request. We also present lower bounds establishing that our algorithms are essentially best possible.

1 Introduction

Paging (also known as *caching*) is a classical online problem, and an important special case of several other online problems (Borodin & El-Yaniv, 1998), which can be motivated through resource management in operating systems. You are given a fast cache memory with capacity to simultaneously store at most a constant number, k, of pages. Requested pages, according to a sequence of *page requests*, have to be loaded into the cache to be served by the operating system. More specifically, pages are requested one by one in an online fashion, and each request needs to be immediately served upon its arrival. Serving a page is done at zero cost if the requested page currently resides in the cache. If this is not the case, then a *page fault* occurs and the page has to first be loaded into the cache (after potentially evicting some other page to make space). This incurs a fixed cost. The underlying algorithmic question is to decide which page to evict each time a page has to be loaded into the cache, with the goal to minimize the total incurred cost, i.e., the total number of page faults.

Paging has been extensively studied and is well-understood. There exists an optimal offline algorithm, LFD (*longest forward distance*), that simply follows the so-called *Belady's rule:* always evict the page that will be requested again the furthest in the future. Note that Belady's rule can only be applied to the offline variant of the problem, where all future page requests are known to the algorithm. With respect to online algorithms, no deterministic online algorithm can obtain a competitive ratio[1] smaller than k (Sleator & Tarjan, 1985). Two simple deterministic algorithms that are k-competitive (Sleator & Tarjan, 1985) exist: FIFO (evict the oldest page in the cache) and LRU (evict the least recently used/requested page from the cache). Fiat et al. (1991) developed a randomized algorithm called MARK that is $(2H_k - 1)$-competitive[2] (Achlioptas et al., 2000). Furthermore this is tight, up to a constant factor of 2, since no randomized algorithm can obtain a competitive ratio better than H_k (Fiat et al., 1991). Later, optimal H_k-competitive randomized algorithms were discovered (Achlioptas et al., 2000; McGeoch & Sleator, 1991).

The above results are tight in the worst case, although inputs encountered in many practical situations may allow for a better performance. The novel research area of *learning-augmented algorithms* attempts to take advantage of such

[*]Equal contribution [1]University of Twente, Enschede, Netherlands [2]University of Southern Denmark, Odense, Denmark [3]Bocconi University, Milan, Italy [4]Max Planck Institute for Informatics, Saarbrücken, Germany [5]IN2P3 Computing Center and CNRS, Villeurbanne, France. Correspondence to: Antonios Antoniadis <a.antoniadis@utwente.nl>, Joan Boyar <joan@imada.sdu.dk>, Marek Eliáš <elias@ba30.eu>, Lene M. Favrholdt <lenem@imada.sdu.dk>, Ruben Hoeksma <r.p.hoeksma@utwente.nl>, Kim S. Larsen <kslarsen@imada.sdu.dk>, Adam Polak <apolak@mpi-inf.mpg.de>, Bertrand Simon <bertrand.simon@cc.in2p3.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]Competitive ratio is the standard performance metric for online algorithms, see Section 1.1 for a definition.

[2]$H_k = \sum_{i=1}^{k} 1/i$ is the k'th harmonic number. Recall that $\ln k \leq H_k \leq 1 + \ln k$.

opportunities and ameliorate shortcomings of worst-case analysis by assuming that the algorithm has black-box access to a set of (e.g., machine-learned) predictions regarding the input. Naturally, the quality of these predictions is not known a priori, hence the goal is to design algorithms with a good performance on the following parameters: *robustness*, which is the worst-case performance guarantee that holds independently of the prediction accuracy; *consistency*, which is the competitive ratio under perfect predictions; and *smoothness*, which describes the rate at which the competitive ratio deteriorates with increasing prediction error.

Given the central role of paging within online algorithms, it is no surprise that learning-augmented paging has been extensively studied as well, and actually a significant number of papers in the area are either directly or indirectly linked to the paging problem. Examples include the seminal paper by Lykouris & Vassilvitskii (2021), who studied *reoccurrence predictions*, i.e., along with each page request the algorithm obtains a prediction on the timepoint of the next request of that page. Their results were later refined by Rohatgi (2020) and Wei (2020). Jiang et al. (2022) investigated the setting in which all requests until the next request of the currently requested page are predicted, whereas Bansal et al. (2022) considered predictions regarding the relative order in which the pages are requested. Antoniadis et al. (2020a) looked into so-called *state predictions* that predict the cache-contents of an optimal algorithm.

Although the above algorithms have been analyzed with respect to their consistency, robustness and smoothness, no consideration has been made regarding the total amount of predicted information. Given that the predicted information needs to be computed through a separate black-box algorithm and also communicated to the actual paging algorithm for each request, a learning-augmented algorithm that is based on a large amount of predicted information may be impractical in a real-world application.

In this paper we study learning-augmented paging while taking a new approach, requiring a minimal amount of predicted information. We assume that the predictions must be encoded in only one bit per request. This is indeed the least possible amount of predicted information (up to a constant) since any (deterministic or randomized) algorithm that receives perfect predictions that can be encoded in sublinearly many bits (in the length of the request sequence) cannot be better than H_k-competitive (Mikkelsen, 2016). Moreover, there are binary classifiers producing one-bit predictions for paging (Jain & Lin, 2016; Shi et al., 2019) which have great performance in practice (see Appendix A for more details) and it is desirable to use them in learning-augmented algorithms.

We study two natural such *setups*, with one-bit predictions, which we call *discard predictions* and *phase predictions*. The predicted bit in discard predictions denotes whether LFD would evict the current page before it gets requested again. In phase predictions, the bit denotes whether the current page will be requested in the following k-phase (the notion of a k-phase is based on marking algorithms, such as MARK and LRU, and it is formally defined in Section 2). Both of these new setups can be interpreted as condensing the relevant information from the previously existing setups into one bit per request.

We develop algorithms for each of the two setups that satisfy all three desirable properties of learning-augmented algorithms – that is, they are consistent, robust and smooth – despite being limited to a one-bit prediction per request. It may appear that straightforward modifications to pre-existing paging algorithms from the literature would directly lead to some of our results. However, we want to stress that new ideas were required in designing our marking algorithm for the discard-predictions setup, and novel ideas are employed in the analysis of our algorithms in both setups. We also present lower bounds establishing that our algorithms are essentially best possible.

1.1 Our Contribution

An important preliminary observation is that there is an asymmetry regarding prediction errors: Wrongly evicting a page will generally only lead to one page-fault once that page is requested again, however keeping a page which should be evicted in the cache can lead to multiple page-faults while the algorithm keeps evicting pages that will be requested again soon. For this reason we distinguish between two types of prediction errors. For a sequence of n page requests, let $p \in \{0,1\}^n$ be the vector of predictions and $p^* \in \{0,1\}^n$ be the ground truth, where, intuitively, a value of 0 means (in both setups) that, according to the prediction, the page requested should stay in cache. We define η_0 and η_1 as the number of incorrect predictions of 0 and 1, respectively, usually leaving out the parameters p and p^* when they are understood:

$$\eta_h(p,p^*) = |\{i \in [n] \mid p_i = h, p_i^* = 1-h\}|\,, \text{ for } h \in \{0,1\}.$$

In order to capture how different types of errors affect the cost of an algorithm, we generalize the notion of competitive ratio to what we call (α,β,γ)-*competitiveness*.

Definition 1.1. A learning-augmented online paging algorithm ALG is called (α,β,γ)-competitive if there exists a constant b (possibly depending on k) such that for any instance I with ground truth p^* and any predictions p,

$$\text{ALG}(I,p) \le \alpha \cdot \text{OPT}(I) + \beta \cdot \eta_0(p,p^*) + \gamma \cdot \eta_1(p,p^*) + b\,,$$

where $\text{ALG}(I,p)$ and $\text{OPT}(I)$ denote[3] costs incurred on this instance by the online algorithm and the offline optimal

[3]Following a standard practice in online algorithms literature,

algorithm, respectively, and η_0, η_1 denote the two types of error of predictions provided to the online algorithm.

Note that the notion of (α, β, γ)-competitiveness generalizes that of the (classical) competitive ratio: an algorithm is c-competitive if and only if it is $(c, 0, 0)$-competitive. Furthermore, it is easy to see that (α, β, γ)-competitiveness directly implies a consistency of α; it also quantifies the smoothness of the algorithm. We can achieve robustness as follows: any deterministic (α, β, γ)-competitive algorithm for paging can be combined with LRU or FIFO through the result of Fiat et al. (1991) to give a deterministic algorithm with a consistency of $(1+\epsilon)\alpha$ and a robustness[4] of $\frac{1+\epsilon}{\epsilon}k$, for any $\epsilon > 0$. Similarly, any randomized (α, β, γ)-competitive algorithm for paging can be combined (see Antoniadis et al. (2020a) and Blum & Burch (2000)) with an H_k-competitive algorithm (Achlioptas et al., 2000; McGeoch & Sleator, 1991) to give a $((1+\epsilon)\alpha)$-consistent and $((1+\epsilon)H_k)$-robust algorithm. Both of these combination approaches work independently of the considered prediction setup. We therefore focus the rest of the paper on giving upper and lower bounds for the (α, β, γ)-competitiveness.

As explained at the beginning of this section, the two types of prediction errors have significantly different impact: keeping a page in cache while it was safe to evict is potentially much more costly than evicting a page that should have been kept. Hence, β will intuitively be much larger than γ in our results. Our lower bounds also show that $\alpha + \beta$ cannot be smaller than the best classical competitive ratio.

We remark that previous papers on learning-augmented paging (e.g., Lykouris & Vassilvitskii (2021); Rohatgi (2020); Wei (2020)) analyze smoothness by expressing the classical competitive ratio as a function of the normalized prediction error $\frac{\eta}{\text{OPT}}$, and that our results could also be stated this way because every (α, β, γ)-competitive algorithm is also $(\alpha + \beta \cdot \frac{\eta_0}{\text{OPT}} + \gamma \cdot \frac{\eta_1}{\text{OPT}})$-competitive in the classical sense.

Discard-predictions setup upper bounds. In Section 3 we develop a deterministic and a randomized algorithm for the discard-predictions setup:

Theorem 1.2. *There is a deterministic $(1, k-1, 1)$-competitive algorithm for the discard-predictions setup.*

The algorithm realizing Theorem 1.2 is very simple and natural: On each page-fault, evict a page that is predicted as safe to evict, if such a page exists. If it does not exist, then just flush the cache, i.e., evict all pages that it contains. The analysis is based on deriving appropriate bounds on the page-faults for both ALG and OPT within any two consecutive flushes, as well as the respective prediction error.

Theorem 1.3. *There is a randomized $(1, 2H_k, 1)$-competitive algorithm for the discard-predictions setup.*

Compared to the deterministic algorithm above, the algorithm from Theorem 1.3 uses an approach resembling the classical MARK algorithm when evicting pages predicted 0. However, we note that it does not fall into the class of so-called *marking algorithms* (see Section 2), as pages predicted 1 are evicted sooner. This is essential for achieving $\alpha = 1$ but requires a different definition of phases and a novel way of charging evictions of pages predicted 0.

Phase-predictions setup upper bounds. For phase predictions, in Section 4 we design an algorithm called MARK&PREDICT which can be seen as a modification of the classical MARK algorithm giving priority to pages predicted 1 when choosing a page to evict. We prove two bounds for this algorithm. The first one is sharper for small η_1 and, in fact, it holds even with deterministic evictions of pages predicted 1.

Theorem 1.4. MARK&PREDICT *is a randomized $(2, H_k, 1)$-competitive algorithm for the phase-predictions setup.*

The second bound exploits the random eviction of pages predicted 1 and gives a much stronger result for larger η_1.

Theorem 1.5. MARK&PREDICT *is a randomized $\big(2, H_k, \gamma(\eta_1/\text{OPT})\big)$-competitive algorithm for the phase-predictions setup, where*

$$\gamma(x) = 2x^{-1}\left(\ln(x+1) + 1\right).$$

In other words, the (expected) cost of MARK&PREDICT is at most

$$2\left(\ln\left(\frac{\eta_1}{\text{OPT}} + 1\right) + 2\right) \cdot \text{OPT} + H_k \cdot \eta_0.$$

Note that this expression should not be considered when $\eta_1 \leq \text{OPT}$ as $\gamma(1) > 1$ so the guarantee of the previous theorem would then be stronger. For $\eta_1 > \text{OPT}$, multiple possibilities exist to phrase the above expression into our (α, β, γ)-competitiveness notion, so we chose the one matching the previously established value of α. To illustrate the gain over the previous bound, with $\eta_1/\text{OPT} = \Omega(k)$, we obtain $\gamma(\eta_1/\text{OPT}) = O\left(\frac{\log k}{k}\right)$, thus asymptotically matching the lower bound of Theorem 1.9.

From the proof of the above theorem, it easily follows that MARK&PREDICT is robust.

Theorem 1.6. MARK&PREDICT *is $(O(\log k))$-robust.*

in what follows, we abuse the notation and use ALG and OPT to denote both the algorithms and their respective costs incurred on the implicitly understood instance.

[4] Actually, Fiat et al. (1991) show the more general result that one can combine m algorithms such that for any input instance I this combination incurs a cost that is within a factor c_i from the cost of each corresponding algorithm i on I. The constants c_i can be chosen arbitrarily as long as they satisfy $\sum_{i=1}^{m} 1/c_i \leq 1$.

Lower bounds. In Appendix C, we give lower bounds that show that the upper bounds above are essentially tight. More specifically, we prove the following for the two considered setups.

Theorem 1.7. *In both the discard-predictions and phase-predictions setups, there is no deterministic (α, β, γ)-competitive algorithm such that either $\alpha + \beta < k$ or $\alpha + (k-1) \cdot \gamma < k$.*

This directly implies that if α is a constant independent of k, then $\beta = \Omega(k)$ and $\gamma = \Omega(1)$. A special case is that any 1-consistent deterministic algorithm must have β at least $k-1$ and γ at least 1, matching the upper bound of Theorem 1.2, or, more precisely:

Corollary 1.8. *In both setups, no deterministic paging algorithm is $(1, k-1-\epsilon, \gamma)$- or $(1, \beta, 1-\epsilon)$-competitive, for any constant $\epsilon > 0$ and any value of β and γ.*

An analogous lower bound can be obtained for randomized algorithms as well.

Theorem 1.9. *In both the discard-predictions and phase-predictions setups, there is no (α, β, γ)-competitive randomized algorithm such that either $\alpha + \beta < H_k$ or $\alpha + (k-1) \cdot \gamma < H_k$, where $H_i = \ln i + O(1)$ is the i-th harmonic number.*

This result implies that, in the upper bounds of Theorems 1.4 and 1.5 for MARK&PREDICT, the value of β is tight up to an additive term of 2 and the asymptotic value of γ, when η_1 / OPT is large, cannot be improved by more than a constant factor in Theorem 1.5.

Corollary 1.10. *In both setups, no randomized paging algorithm is $(2, H_k - 2 - \epsilon, \gamma)$- or $(2, \beta, \frac{H_k - 2}{k-1} - \epsilon)$-competitive, for any constant $\epsilon > 0$ and any value of β and γ.*

The previous theorem implies a subconstant lower bound ($\approx \frac{1}{k} \log k$) on γ for a value of α up to $o(\log k)$. We complement it by showing that γ is lower bounded by a constant if we want to achieve $\alpha = 1$.

Theorem 1.11. *There is no $(1, \beta, \gamma)$-competitive randomized algorithm such that $\gamma < 1/7$ for the discard-predictions setup or $\gamma < 1/2$ for the phase-predictions setup.*

The last two theorems imply that, in the upper bound of Theorem 1.3, the values of β and γ cannot be improved by more than a constant factor.

Corollary 1.12. *In the discard-predictions setup, no randomized paging algorithm is $(1, H_k - 1 - \epsilon, \gamma)$- or $(1, \beta, \frac{1}{7} - \epsilon)$-competitive, for any constant $\epsilon > 0$ and any value of β and γ.*

Similarly to the lower bounds known for classical paging, all three of our lower bound results are based on instances coming from a universe of $k+1$ many pages. However, in order to achieve the desired bounds, we need to carefully define the prediction sequence. Somewhat surprisingly, in each of our lower bound results, we are able to use the same prediction sequence for both prediction setups.

We provide in Appendix A a more extensive discussion on related work regarding learning-augmented algorithms, the usage of few predictions, the differences between our setups and the classic lookahead, the field of advice complexity and the practicality of discard predictions.

1.2 Open Problems

Better dependence on η_1 in discard-predictions setup. For the case of large η_1, we provide a stronger guarantee for MARK&PREDICT in Theorem 1.5. However, we were not able to obtain a comparable result for the discard-predictions setup, and it would be interesting to further close this gap. Somewhat surprisingly, an important challenge towards that direction seems to be that of recognizing the presence of an incorrect 0-prediction early enough. This can be easily done in the phase-predictions setup; and we do actually properly account for all incorrect 0-predictions (see Observation 4.1). On the other hand, our criterion in the discard-predictions setup (see Observation 3.1) may overlook some of them. This in turn may lead an algorithm to keep the cache full with pages associated with 0-predictions, forcing it to evict all pages with 1-predictions, implying $\gamma \geq 1$.

Other online problems with succinct predictions. For many online problems, the possibility of obtaining good succinct predictions might be more realistic than obtaining more precise, lengthy predictions. It would be interesting to see if such predictions still allow for effective learning-augmented algorithms. Prior results on advice complexity give meaningful lower bounds on the size of such predictions and may provide guidance on what to predict.

2 Preliminaries

Classical paging. In paging, we have a (potentially large) universe U of pages and a cache of size k. At each time step $i = 1, \ldots, n$, we receive a request r_i to a page in U which needs to be satisfied by loading the page associated to r_i to the cache (if it is not in the cache already). This may require evicting some other page to make space for the requested page. The goal of an algorithm is to serve the whole request sequence at minimal *cost*. The cost of an algorithm is the number of page loads (or page faults) performed to serve the request sequence. Note that this number is within an additive term k from the number of page evictions. In our analyses, we can choose to work with whichever of these two quantities is easier to estimate, because of the additive constant in the definition of competitiveness.

When making space for the page associated to r_i, *online* algorithms have to decide which page to evict without knowledge of $r_{i+1}, \ldots, r_n$, while *offline* algorithms have this information.

Marking algorithms. For the purpose of designing marking algorithms, we partition the request sequence into k-*phases*. A k-phase is a maximal subsequence of at most k distinct pages. The first k-phase starts at the first request, and any subsequent k-phase i starts at the first request following the last request of k-phase $i-1$.

The following automatic procedure helps designing algorithms for caching: at the beginning of each k-phase, we unmark all pages. Whenever a page is requested for the first time in a k-phase, we mark it. We say that an algorithm belongs to the class of *marking algorithms*, if it never evicts a marked page. All marking algorithms are (at most) k-competitive (Torng, 1998) and they have the same cache content at the end of each k-phase: the k marked pages which were requested during that k-phase.

Algorithm MARK (Fiat et al., 1991) evicts an unmarked page chosen uniformly at random. In the ith k-phase, with c_i pages requested that were not requested in k-phase $i-1$ (we call such pages *new*, the others are called *old*), it has in expectation $\sum_{j=1}^{k-c_i} \frac{c_i}{k-(j-1)} \leq c_i(H_k - H_{c_i} + 1)$ page faults. One can show that $\text{OPT} \geq \frac{1}{2}\sum_{i=1}^{m} c_i$, where m is the total number of k-phases in the request sequence, and hence MARK is at most $2H_k$-competitive. We refer to Borodin & El-Yaniv (1998) for more details.

Receiving predictions. Each request r_i comes with a *prediction*, $p_i \in \{0, 1\}$. If a request comes with a prediction of 0 (resp. 1), we call it a 0-prediction (resp. 1-prediction), and the requested page a 0-page (resp. 1-page) until the next time it is requested. Throughout the paper, we use r_i to refer both to the request and to the page associated with that request.

Discard-predictions setup. In this setup, we fix an optimal offline algorithm, say LFD. When a requested page is not in cache, LFD evicts any page that will never be requested again, if such a page exists, and otherwise evicts the unique page of the k pages in cache that will be requested again furthest out in the future.

Prediction p_i for request r_i is supposed to predict the ground truth p_i^* defined as:

$$p_i^* = \begin{cases} 1, & \text{if LFD evicts } r_i \text{ before it is requested again.} \\ 0, & \text{otherwise.} \end{cases}$$

For a page r_i that LFD retains in cache until the end of the request sequence, $p_i^* = 0$.

For simplicity, we define p^* with respect to a fixed optimal algorithm. However, if the prediction vector p happens to predict well the behavior of any other (good but not necessarily optimal) algorithm, then our upper bounds hold also with respect to the performance of that algorithm in place of OPT.

Phase-predictions setup. In this setup, we partition the request sequence into k-phases, as described above in the paragraph on marking algorithms. We stress that the notion of k-phases is algorithm-independent and only depends on the input sequence.

We define the ground truth p_i^* for request r_i in some k-phase j as follows:

$$p_i^* = \begin{cases} 1, & \text{if } r_i \text{ is not requested in } k\text{-phase } j+1, \\ 0, & \text{if } r_i \text{ is requested in } k\text{-phase } j+1. \end{cases}$$

Note that, in both setups, at the point where a decision is made as to which page to evict, the algorithms only consider the most recent prediction for each page, the one from the most recent request to the page. In the discard-predictions setup, this is the only possibility. In the phase-predictions setup, there could be a page, p, requested more than once in phase i, where the prediction (whether or not it will be requested in phase $i+1$) is inconsistent within phase i. We assume that the last prediction is the most relevant, so only this one is used by our algorithms, and only this one contributes towards a possible error in η_0 or η_1. In fact, we could avoid running the predictor at repeated requests by only producing predictions at the end of each phase, for the pages in the cache at that point. In addition, in the phase-predictions setup, predictions in the last phase do not count at all, and in particular, do not count in η_h.

In a given phase, the pages that are in cache at the beginning of the phase are called *old* pages. Pages requested within a phase that are not old are called *new* pages. Thus, all requests in the first phase are to new pages.

3 Algorithms with Discard Predictions

We first investigate the discard-predictions setup. The following simple observation is useful in the analyses of our algorithms.

Observation 3.1. Consider a moment when there is a set S of 0-pages (whose most recent prediction is 0) of size $k + c - 1$ and a page $r \notin S$ is requested. Then, at least c pages from S have incorrect predictions.

Proof. Page r surely has to be in cache. Each $\rho \in S$ has prediction 0 by the definition of S. Since the cache has

size k, any algorithm needs to have evicted at least $1 + |S| - k = c$ pages from S. In particular this is true for LFD. Therefore at least c pages from S have incorrect predictions. □

3.1 Deterministic Algorithm

Our first algorithm is deterministic, and, despite being very simple, it attains the best possible (α, β, γ)-competitiveness for 1-consistent deterministic algorithms (see the lower bound in Theorem 1.7).

Theorem 1.2. *There is a deterministic $(1, k-1, 1)$-competitive algorithm for the discard-predictions setup.*

Proof. Consider the deterministic algorithm, ALG, that on a fault evicts an arbitrary 1-page, if there is such a page in cache, and flushes the cache otherwise.

We count evictions, and note that up to an additive constant (depending on k), this is the same as the number of faults. We divide the request sequence into *stages*, starting a new stage when ALG flushes the cache (i.e., when it is full and contains only 0-pages). We assume an integer number of stages (an assumption that also only adds up to an additive constant, depending on k) and consider one stage at a time.

First consider 0-pages that are evicted. By definition, ALG evicts k such pages in the stage. Since k 0-pages have arrived in the stage, and a new page must arrive for ALG to flush, at least one of the 0-pages has an incorrect prediction (obvious here, but captured more generally by Observation 3.1) and OPT must have evicted at least one of these $k+1$ pages.

Letting a superscript, s, denote the values of just this stage, and a subscript denote 0-pages and 1-pages, respectively, since both OPT_0^s and η_0^s are at least one, $\text{ALG}_0^s \leq \text{OPT}_0^s + (k-1)\eta_0^s$.

Considering 1-pages, ALG clearly obtains the same result as OPT, except when there is a misprediction, which adds a cost of 1. Thus, $\text{ALG}_1^s \leq \text{OPT}_1^s + \eta_1^s$.

Summing over both predictions and all stages, $\text{ALG} \leq \text{OPT} + (k-1)\eta_0 + \eta_1$. □

Remark 3.2. In Theorem 1.2, the choices $\alpha = 1$ and $\beta = k-1$ can be generalized, showing that the algorithm is $(\alpha, k-\alpha, 1)$-competitive, for $1 \leq \alpha \leq k$ (compare with Theorem 1.7).

3.2 Randomized Algorithm

Now, we present MARK0, a randomized algorithm that evicts all 1-pages immediately. Therefore, whenever the cache is full and eviction is needed, all the pages in the cache must be 0-pages and this situation signals a presence of an incorrect 0-prediction. Since we cannot know which 0-page has an incorrect prediction, we evict a random unmarked one in order to make sure that such evictions can be charged to η_0 in the analysis. MARK0 is described in Algorithm 1. Before proving the competitive ratio of MARK0, we state a few observations, starting by a simple bound on the evictions of 1-pages.

Algorithm 1 MARK0 Eviction Strategy

```
1:  S := ∅
2:  evict all 1-pages
3:  for i = 1 to n do
4:      if r_i is not in cache then
5:          if cache is full and all pages from S in cache are
            marked then
6:              S := current cache content
7:              unmark all pages
8:          end if
9:          if r_i ∈ S is unmarked and cache contains some
            unmarked page from S then
10:             evict an unmarked page chosen uniformly at
                random
                {evict it even if the cache is not full}
11:         end if
12:         if cache is full then
13:             evict an unmarked page chosen uniformly at
                random
14:         end if
15:         bring r_i to cache
16:     end if
17:     mark r_i
18:     if p_i = 1 then
19:         evict r_i
20:     end if
21: end for
```

Pages are marked when they are requested and only unmarked at the end of the phase. So, all the unmarked pages considered for eviction in lines 10 and 13 were in cache from the beginning of the phase and belong to S. In line 19, we evict r_i regardless of its marking status. Therefore, MARK0 is not a marking algorithm[5].

Observation 3.3. The number of 1-pages that MARK0 evicts is at most $\text{OPT} + \eta_1$.

Therefore it is enough to count evictions of 0-pages. We call a period between two executions of line 7 a phase. Phases are similar to k-phases in marking algorithms, with the difference being that 1-pages are directly evicted by the

[5]For example, it can happen that a there is a page fault on a marked page belonging to S: A page in S could be evicted from cache, later requested with a prediction of 1 and evicted immediately, and finally requested with a prediction of 0, all in the same phase.

algorithm, even though such a page is still marked. Phase 1 starts the first time that the cache is full and a page-fault occurs (recall that this implies that there has been an incorrect prediction on a 0-page), since $S = \emptyset$ and the condition on all pages from S in cache being marked is vacuously true. We define phase 0 to be the time from the beginning of the request sequence until the start of Phase 1.

Observation 3.4 bounds the number of evictions of 0-pages based on the number of times an eviction is caused by a full cache. Such an eviction leads to an unmarked page from S being evicted. A classical probabilistic argument is then used to bound the number of times a randomly evicted unmarked page is requested again in the phase.

Observation 3.4. Consider a phase with c executions of line 13. The expected number of evictions of 0-pages is at most cH_k.

Proof. Note that the number of evictions of 0-pages is equal to the number of evictions in lines 13 and 10. There are c evictions made in line 13 and we just need to count evictions made in line 10.

In each execution of line 13, we evict a page from S. In each execution of line 10, one previously evicted page from S replaces another page from S in the cache (which is evicted). The former increases the number of evicted unmarked pages from S by one, while the latter maintains the number of evicted unmarked pages from S.

Consider the first time, t, when there are no unmarked pages from S contained in the cache. Until t, whenever an unmarked page from S is loaded to the cache, it is marked and another unmarked page from S is evicted. Therefore, there are precisely c unmarked pages from S which are not present in cache at time t: the pages evicted at line 13 or the ones these have replaced at line 10. Afterwards, no more evictions of 0-pages are made and such pages are only loaded to the cache until it becomes full and a new phase starts.

To count evictions made in line 10, we need to estimate the probability of a requested unmarked page from S being missing from the cache. We use an approach similar to the classical analysis of the algorithm MARK (Fiat et al., 1991; Borodin & El-Yaniv, 1998). Since it makes the situation only more costly for the algorithm, we can assume that all the evictions in line 13 are performed in the beginning of the phase and the evictions in line 10 are all performed afterwards. When the jth page from S is being marked, it is present in the cache with probability $\frac{k-c-(j-1)}{k-(j-1)}$ (the numerator is the number of unmarked pages from S present in the cache at that moment and the denominator is the total number of unmarked pages in S) and the probability of a page fault is $\frac{c}{k-(j-1)}$. Therefore, the expected number of evictions in line 10 until time t is

$$\sum_{j=1}^{k-c} \frac{c}{k-(j-1)} = c(H_k - H_c).$$

The total expected number of evictions of 0-pages during this phase is then

$$c + c(H_k - H_c) \leq cH_k. \qquad \square$$

We observe that as a consequence of how the phases are defined, every page residing in the cache at the timepoint between two consecutive phases must have received its prediction during the phase that just ended. More formally,

Observation 3.5. Let $S(i)$ be the content of the cache when phase $i-1$ ends and phase i starts. Then all the pages in $S(i)$ received their predictions during phase $i-1$.

Proof. This is a consequence of marking: Every page requested during phase $i-1$ received a new prediction. The only pages from $S(i-1)$ which did not, are the unmarked ones. Yet, such pages are not present in the cache at the end of phase $i-1$. And all pages from $S(i) \setminus S(i-1)$ must have been requested and loaded during phase $i-1$. $\square$

We are now ready to analyze the (α, β, γ)-competitiveness of the algorithm. It combines the previous results and uses the fact that evictions caused by a full cache can be charged to an erroneously predicted 0-page, as trusting the predictions would require keeping more than k pages in cache. An additional factor is required in the dependency on η_0 as a wrong prediction may impact both the current phase and the following one.

Theorem 1.3. *There is a randomized* $(1, 2H_k, 1)$*-competitive algorithm for the discard-predictions setup.*

Proof. Let us show that MARK0 is $(1, 2H_k, 1)$-competitive. Consider a request sequence with optimum cost, OPT, during which MARK0 performs m phases and receives η_0 incorrect predictions 0 and η_1 incorrect predictions 1. Let c_i denote the number of executions of line 13 during phase i. Combining Observations 3.3 and 3.4, the expected cost of MARK0 is at most

$$\text{OPT} + \eta_1 + \sum_{i=1}^{m} c_i H_k.$$

It is enough to show that $\sum_{i=1}^{m} c_i \leq 2\eta_0$ holds.

Consider the moment during phase i when line 13 is executed for the c_ith time. At this moment, there are k 0-pages in cache: some of them belong to $S(i)$, others were loaded during this phase. Moreover, there are $c_i - 1$ unmarked pages

from $S(i)$ already evicted, these are also 0-pages. By Observation 3.1, at least c_i of these pages must have an incorrect prediction of 0. This prediction was received either during phase $i-1$ (if it is an unmarked page from $S(i)$), or during phase i (all other cases). Therefore, denoting $\eta_0(i)$ the number of incorrect predictions 0 received during phase i, we have

$$\sum_i c_i \leq \sum_{i=1}^{m} \big(\eta_0(i-1) + \eta_0(i)\big) \leq 2\eta_0,$$

which concludes the proof. □

4 Algorithm with Phase Predictions

In this section, we consider the phase-predictions setup and give a randomized algorithm, MARK&PREDICT. The idea of this algorithm is to follow the classical MARK algorithm except that, instead of evicting a page uniformly at random among the set of unmarked pages, we select an unmarked 1-page if the cache contains one. We provide two analyses on the performance of MARK&PREDICT, which differ on the bound of the γ parameter, the second bound providing an improvement for large values of η_1. The second proof is deferred to Appendix B.

Algorithm 2 MARK&PREDICT Eviction Strategy

```
mark all pages in cache
for i = 1 to n do
    if r_i is not in cache then
        if all pages in cache are marked then
            unmark all pages {Start of a new phase}
        end if
        if there is an unmarked 1-page then
            evict an unmarked 1-page chosen uniformly at random
        else
            evict an unmarked 0-page chosen uniformly at random
        end if
        bring r_i into cache
    end if
    mark r_i
end for
```

The following observation is used in both proofs to estimate the value of η_0.

Observation 4.1. Consider a phase with c new pages. If $\ell \leq c$ of the 1-pages present at the beginning of the phase were not requested during the phase, then precisely $z = c - \ell$ pages had incorrect 0-predictions at the start of the phase.

Proof. Let S denote the set of 0-pages and L the set of 1-pages that were present at the beginning of the phase. Denote by z the number of pages in S which were not requested during this phase, so their predictions at the beginning of the phase were incorrect. During the phase $k = c + (|L| - \ell) + (|S| - z)$ distinct pages were requested. Since $|S| + |L| = k$, we get $z = c - \ell$. □

We first provide an analysis which also holds if an arbitrary 1-page is evicted at Line 8, in a deterministic manner, say using LRU.

Theorem 1.4. MARK&PREDICT *is a randomized* $(2, H_k, 1)$*-competitive algorithm for the phase-predictions setup.*

Proof. Again, we use standard arguments for the competitive analysis of the randomized paging algorithm, MARK (Fiat et al., 1991), using terminology from the textbook by Borodin & El-Yaniv (1998). We first consider the case where all predictions are correct. Pages that arrive are always marked, so they are never evicted in the current k-phase. Thus, the number of 1-pages that arrive in the current phase will be the number of 1-pages in cache at the beginning of the next phase. If all predictions in a phase are correct, the number of new pages in the next k-phase equals the number of 1-pages at the beginning of that phase, and the new pages will replace those 1-pages. There will be no faults on the 0-pages. Let c_i be the number of new pages in the ith k-phase and m be the total number of phases. Since the algorithm faults only on new pages, it faults $\sum_{i=1}^{m} c_i$ times. We now turn to OPT. During the $i-1$st and ith k-phases, at least $k + c_i$ distinct pages have been requested. Since OPT cannot have had more than k of them in cache at the beginning of phase $i-1$, it must have at least c_i faults in these two phases. Considering the even phases and the odd phases separately and taking the maximum, OPT must fault at least $\frac{1}{2}\sum_{i=1}^{m} c_i$ times. This proves 2-consistency.

As long as 1-pages are evicted, the faults are charged to OPT (if it is a correct prediction) or to η_1 (if the prediction is incorrect). Since OPT is at least $1/2$ times the total number of new pages, this gives a contribution of at most $2\,\text{OPT} + \eta_1$.

If the algorithm runs out of pages with 1-predictions to evict, there are only 0-pages from the previous phase remaining. For each new page processed after this point, there is an incorrect 0-prediction. Let z_i be the number of new pages causing a 0-page to be evicted in Phase i. These new pages, causing evictions of pages with 0-predictions, arrive after the new pages that evicted pages with 1-predictions. The number of 1-pages present in the cache at the start of Phase i is $c_i - z_i$.

We can assume that all new pages arrive before any of the old pages, as this only increases the algorithm's cost. When the first new page evicting a 0-page arrives, there are $k - (c_i - z_i)$ pages from the previous phase still in cache and

these $k-(c_i-z_i)$ pages are all 0-pages. When the first old page arrives, there are $k-c_i$ pages from the previous phase in cache, so the arriving page has a probability of $\frac{k-c_i}{k-(c_i-z_i)}$ of still being in the cache.

Consider the probability that the jth old page (in the order they arrive in this phase) is in cache the first time it is requested in the ith phase. This probability is $\frac{k-c_i-(j-1))}{k-(c_i-z_i)-(j-1)}$, so the probability that there is a fault on it is $\frac{z_i}{k-(c_i-z_i)-(j-1)}$. Hence the expected number of faults in Phase i due to incorrect 0-predictions is at most

$$z_i + \sum_{j=1}^{k-c_i} \frac{z_i}{k-(c_i-z_i)-(j-1)}$$
$$= z_i(1 + H_{k-c_i+z_i} - H_{z_i}) \leq z_i H_{k-c_i+z_i}.$$

By Observation 4.1, the number of pages with incorrect 0 prediction at the beginning of the phase i is z_i. So, this sum over all phases is at most $H_k\eta_0$.

The total number of faults is at most $2\,\textsc{Opt} + H_k\eta_0 + \eta_1$. □

Acknowledgements

Boyar, Favrholdt, and Larsen were supported in part by the Independent Research Fund Denmark, Natural Sciences, grant DFF-0135-00018B and in part by the Innovation Fund Denmark, grant 9142-00001B, Digital Research Centre Denmark, project P40: Online Algorithms with Predictions. Polak was supported in part by Swiss National Science Foundation projects *Lattice Algorithms and Integer Programming* (185030) and *Complexity of integer Programming* (CRFS-2_207365). Part of the research was carried out during the Workshop on Algorithms with Predictions in the Bernoulli Center for Fundamental Studies at EPFL.

References

Achlioptas, D., Chrobak, M., and Noga, J. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1–2):203–218, 2000.

Albers, S. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997.

Angelopoulos, S., Dorrigiv, R., and López-Ortiz, A. On the separation and equivalence of paging strategies and other online algorithms. *Algorithmica*, 81(3):1152–1179, 2019.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., and Simon, B. Online metric algorithms with untrusted predictions. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 345–355. PMLR, 2020a.

Antoniadis, A., Gouleakis, T., Kleer, P., and Kolev, P. Secretary and online matching problems with machine learned advice. In *NeurIPS*, 2020b.

Antoniadis, A., Coester, C., Eliás, M., Polak, A., and Simon, B. Learning-augmented dynamic power management with multiple states via new ski rental bounds. In *NeurIPS*, pp. 16714–16726, 2021.

Antoniadis, A., Ganje, P. J., and Shahkarami, G. A novel prediction setup for online speed-scaling. In *SWAT*, volume 227 of *LIPIcs*, pp. 9:1–9:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Bamas, É., Maggiori, A., Rohwedder, L., and Svensson, O. Learning augmented energy minimization via speed scaling. In *NeurIPS*, 2020.

Bansal, N., Coester, C., Kumar, R., Purohit, M., and Vee, E. Learning-augmented weighted paging. In *SODA*, pp. 67–89. SIAM, 2022.

Blum, A. and Burch, C. On-line learning and the metrical task system problem. *Mach. Learn.*, 39(1):35–58, 2000. doi: 10.1023/A:1007621832648. URL https://doi.org/10.1023/A:1007621832648.

Böckenhauer, H., Komm, D., Královic, R., Královic, R., and Mömke, T. Online algorithms with advice: The tape model. *Inf. Comput.*, 254:59–83, 2017.

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

Boyar, J., Favrholdt, L. M., and Larsen, K. S. The relative worst-order ratio applied to paging. *Journal of Computer and System Sciences*, 73(5):818–843, 2007.

Boyar, J., Favrholdt, L. M., Kudahl, C., Larsen, K. S., and Mikkelsen, J. W. Online Algorithms with Advice: A Survey. *ACM Computing Surveys*, 50(2):1–34, 2017. Article No. 19.

Boyar, J., Favrholdt, L. M., and Larsen, K. S. Online unit profit knapsack with untrusted predictions. In *SWAT*, volume 227 of *LIPIcs*, pp. 20:1–20:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Breslauer, D. On competitive on-line paging with lookahead. *Theoretical Computer Science*, 209(1–2):365–375, 1998.

Chen, J., Silwal, S., Vakilian, A., and Zhang, F. Faster fundamental graph algorithms via learned predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 3583–3602. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/chen22v.html.

Dinitz, M., Im, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Faster matchings via learned duals. In *NeurIPS*, pp. 10393–10406, 2021.

Dobrev, S., Královič, R., and Pardubská, D. Measuring the problem-relevant information in input. *RAIRO - Theor. Inf. Appl.*, 43(3):585–613, 2009.

Dorrigiv, R., López-Ortiz, A., and Munro, J. I. On the relative dominance of paging algorithms. *Theoretical Computer Science*, 410:3694–3701, 2009.

Dütting, P., Lattanzi, S., Leme, R. P., and Vassilvitskii, S. Secretaries with advice. In *EC*, pp. 409–429. ACM, 2021.

Eberle, F., Lindermayr, A., Megow, N., Nölke, L., and Schlöter, J. Robustification of online graph exploration methods. In *AAAI*, pp. 9732–9740. AAAI Press, 2022.

Emek, Y., Fraigniaud, P., Korman, A., and Rosén, A. Online computation with advice. *Theor. Comput. Sci.*, 412(24): 2642–2656, 2011.

Ergun, J. C., Feng, Z., Silwal, S., Woodruff, D. P., and Zhou, S. Learning-augmented k-means clustering. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=X8cLTHexYyY.

Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D., and Young, N. E. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

Hromkovič, J., Královič, R., and Královič, R. Information complexity of online problems. In *MFCS*, volume 6281 of *LNCS*, pp. 24–36. Springer, 2010.

Im, S., Kumar, R., Qaem, M. M., and Purohit, M. Online knapsack with frequency predictions. In *NeurIPS*, pp. 2733–2743, 2021.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9588–9601. PMLR, 2022.

Jain, A. and Lin, C. Back to the future: Leveraging Belady's algorithm for improved cache replacement. In *43rd ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2016*, pp. 78–89. IEEE Computer Society, 2016. doi: 10.1109/ISCA.2016.17.

Jiang, Z., Panigrahi, D., and Sun, K. Online algorithms for weighted paging with predictions. *ACM Trans. Algorithms*, 18(4):article 39, 27 pages, 2022.

Lindermayr, A. and Megow, N. Permutation predictions for non-clairvoyant scheduling. In *SPAA*, pp. 357–368. ACM, 2022a.

Lindermayr, A. and Megow, N. Algorithms with predictions. https://algorithms-with-predictions.github.io, 2022b. URL https://algorithms-with-predictions.github.io. [Online; accessed 8-September-2022].

Lindermayr, A., Megow, N., and Simon, B. Double coverage with machine-learned advice. In *ITCS*, volume 215 of *LIPIcs*, pp. 99:1–99:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Liu, E. Z., Hashemi, M., Swersky, K., Ranganathan, P., and Ahn, J. An imitation learning approach for cache replacement. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6237–6247. PMLR, 2020. URL http://proceedings.mlr.press/v119/liu20f.html.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *J. ACM*, 68(4):24:1–24:25, 2021.

McGeoch, L. A. and Sleator, D. D. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

Mikkelsen, J. W. Randomization can be as helpful as a glimpse of the future in online computation. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55 of *LIPIcs*, pp. 39:1–39:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. In *Beyond the Worst-Case Analysis of Algorithms*, pp. 646–662. Cambridge University Press, 2020.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. *Commun. ACM*, 65(7):33–35, 2022.

Polak, A. and Zub, M. Learning-augmented maximum flow. *CoRR*, abs/2207.12911, 2022.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ML predictions. In *NeurIPS*, pp. 9684–9693, 2018.

Rohatgi, D. Near-optimal bounds for online caching with machine-learned advice. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1834–1845, 2020.

Sakaue, S. and Oki, T. Discrete-convex-analysis-based framework for warm-starting algorithms with predictions. *CoRR*, abs/2205.09961, 2022. doi: 10.48550/arXiv.2205.09961.

Shi, Z., Huang, X., Jain, A., and Lin, C. Applying deep learning to the cache replacement problem. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2019*, pp. 413–425. ACM, 2019. doi: 10.1145/3352460.3358319.

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

Torng, E. A unified analysis of paging and caching. *Algorithmica*, 20:175–200, 1998.

Wei, A. Better and simpler learning-augmented online caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176 of *LIPIcs*, pp. 60:1–60:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Young, N. E. Competitive paging and dual-guided algorithms for weighted caching and matching (phd thesis). Technical Report CS-TR-348-91, Computer Science Department, Princeton University, 1991.

Zeynali, A., Sun, B., Hajiesmaili, M. H., and Wierman, A. Data-driven competitive algorithms for online knapsack and set cover. In *AAAI*, pp. 10833–10841. AAAI Press, 2021.

A Further Related Work

Paging with few predictions. In a recent paper, Im et al. (2022) consider a different approach to limiting the amount of predicted information within learning-augmented paging. Their algorithm has access to an ML-oracle which can be queried at any time about the reoccurrence prediction for any page in the cache. They analyze the tradeoffs between the number of queries, the prediction error and algorithm performance. The competitive ratio of the obtained algorithms is $O(\min\{\log_{b+1} n + \frac{E[\eta]}{\mathrm{OPT}}], \log_k\})$, where b is the number of queries per page fault. Thus, the consistency of the algorithm would generally be quite far from those of the algorithms presented in this paper. In particular, to achieve a constant consistency, the algorithm has to make $\mathrm{poly}(k)$ queries for each cache miss, each query requiring $\log n$ bits, where n is the number of requests. Therefore, the number of requested bits is smaller than for our algorithms only for instances with very low cache-miss rates.

Comparison to lookahead. Due to possibilities of pipe-lining in computer hardware, online paging has been studied theoretically using the concept of lookahead (Young, 1991; Albers, 1997; Breslauer, 1998; Dorrigiv et al., 2009; Boyar et al., 2007; Angelopoulos et al., 2019). An algorithm with lookahead has access to accurate predictions for a set of future requests, the lookahead, satisfying some property. The models for lookahead are clearly different from our scenario in two major respects: They assume that the lookahead predictions are always correct, an oversimplification since some errors will occur due to imperfect branch predictions, and, additionally, the lookahead predictions contain much more information. However, Albers' *strong lookahead* (Albers, 1997) is very related to our phase-predictions setup for paging with predictions. The strong lookahead contains a consecutive sequence of pages containing at least $\ell + 1$ distinct pages, not counting the page for the current request. Thus, when $\ell = k - 2$, the pages in the k-phase that was just started are known. Albers proves, for example, that 2-competitiveness is optimal when $\ell = k - 2$. More than one algorithm is considered and a randomized algorithm considered is basically MARK with strong lookahead of size ℓ.

Other learning-augmented online algorithms. In addition to the already mentioned results on learning-augmented paging, several exciting learning-augmented algorithms have been developed for various online problems, including among others weighted paging (Bansal et al., 2022), k-server (Lindermayr et al., 2022), metrical task systems (Antoniadis et al., 2020a), ski-rental (Purohit et al., 2018; Antoniadis et al., 2021), non-clairvoyant scheduling (Purohit et al., 2018; Lindermayr & Megow, 2022a), online-knapsack (Im et al., 2021; Zeynali et al., 2021; Boyar et al., 2022), secretary and matching problems (Dütting et al., 2021; Antoniadis et al., 2020b), graph exploration (Eberle et al., 2022), as well as energy-efficient scheduling (Bamas et al., 2020; Antoniadis et al., 2022; 2021). Machine- learned predictions have also been considered for designing offline algorithms with an improved running time, see for instance the results of Dinitz et al. (2021) on matchings, Chen et al. (2022) on graph algorithms, Ergun et al. (2022) on k-means clustering, Sakaue & Oki (2022) on discrete optimization, and Polak & Zub (2022) on maximum flows. An extensive list of results in the area can be found on the website maintained by Lindermayr & Megow (2022b). We would also like to point the reader to the surveys by Mitzenmacher & Vassilvitskii (2020; 2022).

We note that, although our work is closer in spirit to the aforementioned results on learning-augmented paging, our notion of (α, β, γ)-competitiveness is an extension of the (ρ, μ)-competitiveness from Antoniadis et al. (2021). While (ρ, μ)-competitiveness captures the tradeoff between the dependence on the optimal cost and the prediction error, (α, β, γ)-competitiveness captures the three-way tradeoff between the dependence on the optimal cost and the two kinds of prediction errors.

Advice complexity. An inspiration for considering paging with succinct predictions is that ideas from the research area of *advice complexity* could possibly be applied to learning-augmented algorithms; in particular, the advice from Dobrev et al. (2009) for the paging problem. The goal, when studying online algorithms with advice, is to determine for online problems how much information about the future is necessary and sufficient to perform optimally or to achieve a certain competitive ratio. This is formalized in different computational models, all of which assume that the online algorithm is given some number of bits of advice (Dobrev et al., 2009; Hromkovič et al., 2010; Böckenhauer et al., 2017; Emek et al., 2011). (See the survey on online algorithms with advice (Boyar et al., 2017).) The difference from learning-augmented algorithms is that the advice is always correct, so robustness is not a consideration, and the emphasis is on the number of bits the algorithms use, rather than if one could realistically expect that the advice could be obtained. The advice-complexity result that is probably the closest to our work is by Dobrev et al. (2009) who studied advice that is equivalent to the ground truth for our discard predictions. Their result implies for our setting that when the predictions are guaranteed to be perfect (as one assumes in advice complexity), then one can obtain a simple 1-competitive algorithm, with predictions of just one bit per

request. However, it does not immediately imply a positive result in our setting when the predictions are of unknown quality.

Discard predictions in practice. Previous research shows the practicality of the succinct predictions presented in this paper. Jain & Lin (2016) proposed Hawkeye, an SVM-based binary classifier whose goal is to predict whether a requested page is likely to be kept in cache by the optimal Belady's algorithm. The classifier labels each page as either *cache-friendly* or *cache-averse*, which directly corresponds to zero and one, respectively, in our discard-predictions setup. Hawkeye's predictions were accurate enough for winning the 2nd Cache Replacement Championship. Later, Hawkeye was outperformed by Shi et al.'s Glider (Shi et al., 2019), a deep learning LSTM-based predictor that solves the same binary classification problem. In short, accurate succinct discard predictions are available for practical applications. On the other hand, machine-learning models capable of producing reoccurrence predictions and state predictions only recently started being developed, and, while they also have a surprisingly high accuracy, they are prohibitively large and slow to evaluate for performance-critical applications (Liu et al., 2020).

B Complementary Analysis of MARK&PREDICT

We provide another analysis of MARK&PREDICT (see Section 4) which exploits the uniformly-random selection of an unmarked 1-page to evict in line 8, and improves on the bound from Theorem 1.4 for larger values of η_1. We also show that MARK&PREDICT is robust.

Lemma B.1. *Consider a phase with c new pages, such that* MARK&PREDICT *starts with η_0 and η_1 pages with incorrect predictions* 0 *and* 1 *in its cache. The expected cost incurred by* MARK&PREDICT *is at most*

$$c\big(H_{\eta_1+c} - H_c + 1\big) + H_k\,\eta_0.$$

Proof. Each phase starts at line 5 by unmarking all pages. We denote L the set of 1-pages contained in the cache at this moment. Note that any unmarked 1-page evicted at line 8 always belongs to L. We analyze two parts of the phase separately: (a) the first part when there are still unmarked 1-pages in the cache and evictions are done according to line 8 and (b) when all unmarked 1-pages are evicted and evictions are done by line 10.

Part (a) Marked pages are always in cache, therefore we only need to count page faults when an unmarked page r_i is requested. Let $c_a \leq c$ denote the number of new pages to arrive during the part (a). Without loss of generality, we can assume that all of them arrive in the beginning. There are three possibilities:

- r_i is new: MARK&PREDICT incurs a cost of 1.
- r_i is not new and was a 0-page (*i.e.*, the previous prediction on the page r_i is 0): MARK&PREDICT incurs a cost of 0 (all such pages are in cache now)
- r_i is not new and was a 1-page: MARK&PREDICT incurs a cost of ζ_i in expectation,

where $\zeta_i = c_a/(|L| - (j-1))$ if this was the j-th page from L being marked. This follows by an argument similar to the classical analysis of MARK, as in the proof of Theorem 1.4: The probability that r_i is in cache is $\frac{|L|-c_a-(j-1)}{|L|-(j-1)}$, implying that the probability that r_i is missing from the cache is $c_a/(|L| - (j-1))$.

Therefore, our expected cost during part (a) is at most

$$c_a + \sum_{j=1}^{|L|-c_a} \frac{c_a}{|L| - (j-1)} = c_a(1 + H_{|L|} - H_{c_a}).$$

At the end of the part (a), we have precisely c_a pages in L that are no longer in the cache, because part (b) starts only if there are more new pages in the phase than pages in L that are never marked. All the pages from L that are marked at the end of the phase had incorrect predictions, so we have $\eta_1 \geq |L| - c_a$ implying $|L| \leq \eta_1 + c_a$. Therefore, our expected cost is at most

$$c_a(H_{\eta_1+c_a} - H_{c_a} + 1) \leq c(H_{\eta_1+c} - H_c + 1).$$

Part (b) This part never happens if c is the number of pages in L with correct prediction 1, i.e., those left unmarked until the end of the phase. If c is higher, then there must have been some pages with incorrect prediction 0. Without loss of generality, we can assume that all $c - c_a$ new pages are requested in the beginning of part (b). Again, we only need to count page faults due to requests r_i where r_i is unmarked. We have the following cases:

- r_i is new: MARK&PREDICT incurs a cost of 1,
- $r_i \in L$: MARK&PREDICT incurs a cost of 1 because all unmarked pages from L are evicted by the end of part (a),
- $r_i \notin L$: MARK&PREDICT incurs a cost of ζ_i.

Similar to the previous case, we have $\zeta_i = (c - c_a)/(k - |L| - (j-1))$ if this is the jth 0-page being marked, because the phase starts with $k - |L|$ 0-pages in cache and they are not evicted during part (a). By Observation 4.1, each arrival of a new page and each request to a further unmarked page in L increases η_0 by 1. Moreover, we have $c - c_a \leq \eta_0$. Therefore, our cost is at most

$$\eta_0 + \sum_{j=1}^{k-|L|-(c-c_a)} \frac{\eta_0}{k - |L| - (j-1)} \leq \eta_0(H_k - H_{\eta_0} + 1) \leq \eta_0 H_k.$$

□

We next give a second upper bound on the (α, β, γ)-competitiveness of MARK&PREDICT, which is stronger for large values of η_1.

Theorem 1.5. MARK&PREDICT *is a randomized* $\big(2, H_k, \gamma(\eta_1/\,\text{OPT})\big)$*-competitive algorithm for the phase-predictions setup, where*

$$\gamma(x) = 2x^{-1}\left(\ln(x+1) + 1\right).$$

Proof. Let $c_i, \eta_1(i), \eta_0(i)$ be the number of new pages, 1-errors, and 0-errors in ith phase, respectively. Then, by the preceding lemma, the cost of the algorithm is at most

$$\sum_i \left(c_i\big(H_{\eta_1(i)+c_i} - H_{c_i} + 1\big) + H_k\,\eta_0(i) \right) \leq \sum_i c_i \left(\ln\big(\frac{\eta_1(i)}{c_i} + 1\big) + 2 \right) + H_k\,\eta_0,$$

where the sum is over all phases. The inequality above holds because $H_{\eta+c} - H_c \leq \ln(\frac{\eta+c}{c}) + 1$. By the concavity of logarithm, the worst case happens when $\eta_1(i)/c_i$ is the same in all the phases, i.e., $\eta_1(i)/c_i = \eta_1/\,\text{OPT}$ for all i. In addition, $\text{OPT} \geq \frac{1}{2}\sum_i c_i \geq \frac{1}{2}\,\text{OPT}$. Therefore, the expected cost of MARK&PREDICT is at most

$$2\,\text{OPT}\left(\ln(\frac{\eta_1}{\text{OPT}} + 1) + 2\right) + H_k\eta_0 \leq 2\,\text{OPT} + H_k\,\eta_0 + \left(\ln(\frac{\eta_1}{\text{OPT}} + 1) + 1\right)\frac{2\,\text{OPT}}{\eta_1}\eta_1.$$

□

Lemma B.1 can also be used to show that MARK&PREDICT is robust.

Theorem 1.6. MARK&PREDICT *is* $(O(\log k))$*-robust.*

Proof. If a phase has c new pages, there were c pages evicted during the previous phase. Let E denote the set of c pages evicted in that previous phase. The correct predictions for all of the pages in E is 1, and it is 0 for all other pages in that previous phase. Considering the current phase, the one using those predictions, $\eta_0 \leq |E| = c$ and $\eta_1 \leq k - |E| = k - c$. The bound from Lemma B.1 is thus at most $c(H_k - H_c + 1) + cH_k \in O(c \log k)$. Summing this over all phases, we can bound MARK&PREDICT's cost by $O((\log k)\,\text{OPT})$. □

C Lower Bounds

In this section, we provide lower bounds on the possible values of α, β and γ for (α, β, γ)-competitive algorithms, in both setups. These bounds imply that the results of the previous two sections are essentially tight.

We first consider deterministic algorithms.

Theorem 1.7. *In both the discard-predictions and phase-predictions setups, there is no deterministic (α, β, γ)-competitive algorithm such that either $\alpha + \beta < k$ or $\alpha + (k-1) \cdot \gamma < k$.*

Proof. Consider any deterministic paging algorithm ALG, and the following two paging problem instances on a universe of $k+1$ pages, each with n requests, where $n > k$ can be arbitrarily large. When there are only $k+1$ pages used, the concept of k-phases for marking algorithms (Torng, 1998) is used to show that LFD faults on the first occurrence of each of the first k pages requested in the first phase and on the first page in each phase after that, for a total of $\text{OPT} \leq k + \lceil \frac{n-k}{k} \rceil$ faults. Ignoring the first and last phases, LFD always evicts the only page not present in that phase, so correct predictions in the discard-predictions and phase-predictions setups are identical, with zeros for every request, except for the last occurrence of the page not requested in the next phase. (If the last phase contains fewer than k different pages, there could be more than one correct 1-prediction in the next to last phase, but one is sufficient. In the last phase, the correct predictions would all be zeros.)

In both instances, after k requests, one to each of k different pages, the unique page absent in the cache of ALG is always requested. This leads to a cost of n for ALG, since it faults on all requests.

In the first instance, all predictions are 0. Thus, $\eta_0 \leq \text{OPT} - k$ and $\eta_1 = 0$. Writing $\text{ALG} \leq \alpha\,\text{OPT} + \beta\eta_0 + \gamma\eta_1$, we obtain that

$$n = \text{ALG} \leq \alpha \cdot \left(k + \left\lceil \frac{n-k}{k} \right\rceil\right) + \beta \cdot \left\lceil \frac{n-k}{k} \right\rceil.$$

Taking the limit as n goes to infinity, one must have

$$\alpha + \beta \geq k.$$

In the second instance, all predictions are 1. Thus, $\eta_0 = 0$ and $\eta_1 \leq n - (\text{OPT} - k)$. Writing $\text{ALG} \leq \alpha\,\text{OPT} + \beta\eta_0 + \gamma\eta_1$, we obtain that

$$n = \text{ALG} \leq \alpha \cdot \text{OPT} + \gamma \cdot (n - (\text{OPT} - k)).$$

Since $\alpha \geq 1$, $\alpha \geq \gamma$. Then, $\text{OPT} \leq k + \lceil \frac{n-k}{k} \rceil$ implies that

$$n = \text{ALG} \leq \alpha \cdot \left(k + \left\lceil \frac{n-k}{k} \right\rceil\right) + \gamma \cdot \left(n - \left\lceil \frac{n-k}{k} \right\rceil\right).$$

Taking the limit as n goes to infinity, one must have

$$\alpha + (k-1) \cdot \gamma \geq k.$$

□

We now focus on randomized algorithms. The next result first considers a single instance with different predictions to exhibit two tradeoffs on the possible competitive ratios, and the second tradeoff is then improved using a different adversarial strategy.

Theorem 1.9. *In both the discard-predictions and phase-predictions setups, there is no (α, β, γ)-competitive randomized algorithm such that either $\alpha + \beta < H_k$ or $\alpha + (k-1) \cdot \gamma < H_k$, where $H_i = \ln i + O(1)$ is the i-th harmonic number.*

Proof. Consider any randomized paging algorithm ALG, and two paging problem instances on a universe of $k+1$ pages. In order to simplify the mathematical expressions, we assume that the instance starts with a full cache with predictions associated to each page. Since there is an additive constant in the definition of the competitive ratio, this does not affect the result.

In this proof, we build on the well-known proof using Yao's principle for the H_k lower bound for paging (Borodin & El-Yaniv, 1998).

In the first instance, for each request, one of the $k+1$ pages is chosen uniformly at random, with a prediction of 0. This leads to an expected cost of $n/(k+1)$ for ALG, as the probability that the requested page is the only one absent from the cache of ALG is $1/(k+1)$.

The expected optimal cost is equal to the expected number of *k-phases* in the instance. The expected length of a phase is, by the Coupon Collector problem, $(k+1)H_{k+1} - 1 = (k+1)H_k$, where H_i is the i-th harmonic number. So $\mathbb{E}[\text{OPT}] = n/((k+1)H_k)$.

For the discard-predictions setup, this means that $\eta_0 = \text{OPT}$ and $\eta_1 = 0$ as each optimal eviction is equivalent to a prediction error.

For the phase-predictions setup, this also means that $\eta_0 = \text{OPT}$ and $\eta_1 = 0$ as each phase contains a single erroneous prediction, on the last request of the page not requested in the following phase.

Hence, we obtain that, for both setups,

$$\begin{aligned}\frac{n}{k+1} &= \mathbb{E}[\text{ALG}] \leq \alpha\mathbb{E}[\text{OPT}] + \beta\mathbb{E}[\eta_0] + \gamma\eta_1 \\ &\leq (\alpha+\beta)\cdot\frac{n}{(k+1)H_k},\end{aligned}$$

so

$$\alpha + \beta \geq H_k.$$

We note below that replacing the predictions from 0 to 1 does not lead to the target bound. Indeed, consider an instance such that, at each round, one of the $k+1$ pages is requested at random, with a prediction of 1. This again leads to an expected cost of $n/(k+1)$ for ALG and $n/((k+1)H_k)$ for OPT. This means that $\eta_1 \leq n$ and $\eta_0 = 0$ for both setups. Hence, we obtain that, for both setups,

$$\begin{aligned}\frac{n}{k+1} &= \mathbb{E}[\text{ALG}] \leq \alpha\mathbb{E}[\text{OPT}] + \beta\eta_0 + \gamma\mathbb{E}[\eta_1] \\ &\leq (\alpha + (k+1)H_k\cdot\gamma)\cdot\frac{n}{(k+1)H_k},\end{aligned}$$

so

$$\alpha + (k+1)H_k\cdot\gamma \geq H_k.$$

In order to improve this bound, we keep a universe of $k+1$ pages and from an optimal solution, we build an instance phase by phase, based on OPT's cache, C, before the start of each phase. The first request is the page p_0 not in C. Then, we consider a uniformly random permutation $\sigma_1, \ldots, \sigma_{k-1}$ of $k-1$ among the k elements of C. The phase will then be described as a composition of *blocks* of requests, where the ith block contains $i+1$ page requests: p_0 and the σ_j for $j \leq i$. For instance, if the permutation is (a, b, c, d, e), the blocks will be:

$$p_0a, p_0ab, p_0abc, p_0abcd, p_0abcde.$$

Each block is furthermore repeated several times before requesting the next block to ensure that the cache of any sensible algorithm contains the pages inside a block afterwards.

We now compute a lower bound on the expected cost of any algorithm on such a sequence. Before the first block, p_0 is contained in the cache, so the probability that requesting a incurs a cache miss is $1/k$, as, except p_0, one of the k other pages in the universe incurs a cache miss. Similarly, the probability that the second block incurs a cache miss is $1/(k-1)$, and the total expected number of cache misses after the last block is $H_k - 1$. We now notice that we can also charge one eviction for p_0 at the start of the phase. Indeed, after the previous phase was finished, the algorithm's cache must contain the k pages of the last block, to avoid suffering too many evictions. Therefore, its cache at the start of the phase matches the one of OPT so does not contain p_0. So the algorithm's cost is at least H_k per phase while OPT's is one per phase.

We now describe predictions: all requests come with a prediction 0 except the last iteration of the last block where all k requests have a prediction 1.

For the discard-predictions setup, the zero-predictions are correct as these pages are requested again in the same phase, and $k-1$ one-predictions are wrong on the last iteration as only a single page should be evicted, so $\eta_0 = 0$ and $\eta_1 = k-1$ per phase.

For the phase-predictions setup, only the last iteration counts towards the error, and a single page will not appear in the next phase so we also have $\eta_0 = 0$ and $\eta_1 = k - 1$ per phase. Therefore, generalizing to all phases, we have

$$H_k = \mathbb{E}[\text{ALG}] \leq \alpha\mathbb{E}[\text{OPT}] + \beta\eta_0 + \gamma\mathbb{E}[\eta_1] \leq \alpha + (k-1) \cdot \gamma. \qquad \square$$

The previous result shows a subconstant lower bound on γ ($\approx \frac{1}{k} \ln k$) for a logarithmic value of α (up to $O(\log k)$), and we complement it by showing that γ is lower bounded by a constant if we want to achieve $\alpha = 1$.

Theorem 1.11. *There is no $(1, \beta, \gamma)$-competitive randomized algorithm such that $\gamma < 1/7$ for the discard-predictions setup or $\gamma < 1/2$ for the phase-predictions setup.*

Proof. We consider a universe of $k + 1$ pages. We construct an instance composed of m rounds of $k - 1$ requests, m being a large integer. At the start of each round, request the page 1, 2 or 3 with equal probability associated to a prediction 1. Then, all pages from 4 to $k + 1$ are requested with a prediction 0.

An optimal algorithm never evicts the pages 4 to $k + 1$ and needs to evict a single page per phase, where phases are defined as for marking algorithms. Any online algorithm has a probability at least $1/3$ to perform an eviction at each round: either one page among $\{1, 2, 3\}$ is not in the cache at the start of the round, or another page is absent which enforces an eviction.

The expected number of rounds in a phase is equal to the expected length of a phase of a uniformly random request sequence over 3 pages and $k = 2$, which is $3H_2 = 4.5$. So $\mathbb{E}[\text{OPT}] = m/4.5$.

We now focus on the prediction errors. First, note that $\eta_0 = 0$ in both setups: the pages predicted 0 are requested in every phase and should never be evicted by an optimal algorithm.

Then, we have $\mathbb{E}[\eta_1] = m - \mathbb{E}[\text{OPT}] = 3.5m/4.5$ in the discard-predictions setup. Indeed, the pages 1, 2 and 3 combined are predicted 1 a total of m times, are never predicted 0 and OPT only evicts these three pages. So there is an error when such a page is not evicted by OPT before its subsequent request.

In the phase-predictions setup, there is one error per phase, for the last prediction of the unique page among $\{1, 2, 3\}$ which is both requested in that phase but not the following one. Therefore, $\mathbb{E}[\eta_1] = m/3H_2 = m/4.5$.

Therefore, we have:

$$\begin{aligned}
\mathbb{E}[\text{ALG}] &\leq 1 \cdot \mathbb{E}[\text{OPT}] + \gamma\mathbb{E}[\eta_1] \\
\gamma &\geq \frac{\mathbb{E}[\text{ALG}] - \mathbb{E}[\text{OPT}]}{\mathbb{E}[\eta_1]} \\
\gamma &\geq \frac{m}{\mathbb{E}[\eta_1]} \cdot \left(\frac{1}{3} - \frac{1}{3H_2}\right) \geq \frac{m}{\mathbb{E}[\eta_1]} \cdot \left(\frac{1.5 - 1}{4.5}\right) \\
&\geq \frac{m}{9\mathbb{E}[\eta_1]}
\end{aligned}$$

So, in the discard-predictions setup, we get $\mathbb{E}[\text{ALG}] \geq 1/7$ and for the phase-predictions setup, we obtain $\mathbb{E}[\text{ALG}] \geq 1/2$. $\square$