New and Improved Bounds for Markov Paging

Chirag Pabbaraju[*]   Ali Vakilian[†]

September 14, 2025

Abstract

In the Markov paging model, one assumes that page requests are drawn from a Markov chain over the pages in memory, and the goal is to maintain a fast cache that suffers few page faults in expectation. While computing the optimal online algorithm (OPT) for this problem naively takes time exponential in the size of the cache, the best-known polynomial-time approximation algorithm is the dominating distribution algorithm due to Lund, Phillips and Reingold (FOCS 1994), who showed that the algorithm is 4-competitive against OPT. We substantially improve their analysis and show that the dominating distribution algorithm is in fact 2-competitive against OPT. We also show a lower bound of 1.5907-competitiveness for this algorithm—to the best of our knowledge, no such lower bound was previously known.

---

[*]Stanford University. Email: `cpabbara@cs.stanford.edu`.
[†]Toyota Technological Institute at Chicago. Email: `vakilian@ttic.edu`.

1 Introduction

The online paging problem is a fundamental problem in the area of online algorithms. There are n pages in slow memory, and requests for these pages arrive in a sequential manner. We are allowed to maintain a fast cache of size k, which initially comprises of some k pages. At time step t, if the page that is requested, say p_t, exists in the cache, this corresponds to a cache hit: we suffer a zero cost, the cache stays as is, and we move on to the next request. Otherwise, we incur a cache miss/page fault, suffer a unit cost, and are forced to evict some page from the cache so as to bring p_t from the slow memory into the cache. A paging algorithm/policy is then specified by how it chooses the page to evict whenever it suffers a cache miss.

An optimal offline algorithm for this problem is the algorithm that has entire knowledge of the page request sequence, and thereafter makes its eviction choices in a way that minimizes the total number of cache misses it incurs. The classical Farthest-in-Future algorithm [Belady, 1966], which at any cache miss, evicts the page that is next requested latest in the remainder of the sequence, is known to be an optimal offline policy. In the spirit of competitive analysis as introduced by Sleator and Tarjan [1985], one way to benchmark any online algorithm (i.e., one which does not have knowledge of the page request sequence) is to compute the ratio of the number of page faults that the algorithm suffers, to the number of page faults that the optimal offline algorithm suffers, on a *worst-case* sequence of page requests. It is known that any deterministic online algorithm has a worst-case ratio that is at least k, while any randomized online algorithm has a worst-case ratio of $\Omega(\log k)$.

Taking a slightly more optimistic view, one can also consider benchmarking an algorithm against the best *online* algorithm, that does not know beforehand the realized sequence of page requests. Additionally, in the context of beyond worst-case analysis, one can impose assumptions on the sequence of page requests to more accurately model the behavioral characteristics of real-world page requests on computers. For example, page requests generally follow the *locality of reference* principle, either in time (the next time a particular page is requested is close to its previous time) or in space (the page that will be requested next is likely to be in a nearby memory location).

The Markov Paging model, introduced by Karlin et al. [1992], is one of the ways to model such locality of reference in the page requests. The model assumes that page requests are drawn from a (known) Markov chain over the entire set of n pages. Once such a distributional assumption over the page requests is imposed, one can ask how an online paging algorithm fares, in comparison to the optimal online algorithm that suffers the smallest number of page misses *in expectation* over the draw of a page sequence from the distribution. As it turns out, one can compute such an optimal online algorithm, but only in time exponential in the cache size k. Therefore, one seeks efficient polynomial-time algorithms that are approximately optimal.

In 1994, Lund et al. [1994] proposed an extremely elegant randomized algorithm for the Markov Paging problem, known as the dominating distribution algorithm. The dominating distribution algorithm, whenever it suffers a cache miss, evicts a random page drawn from a special distribution, which has the property that a randomly drawn page is likely to be next requested latest among all the existing pages in the cache. This algorithm, which has since become a popular textbook algorithm due to its simplicity (e.g., see Chapter 5 in Borodin and El-Yaniv [2005], or the lecture notes by Roughgarden [2010]), runs in time polynomial in k; furthermore, Lund et al. [1994] show that the algorithm suffers only 4 times more cache misses in expectation than the optimal online algorithm. Since then, this has remained state-of-the-art—we do not know any other polynomial-time algorithms that achieve a better performance guarantee. It has nevertheless been conjectured

that the above guarantee for the dominating distribution algorithm is suboptimal (e.g., see Section 6.2 in Roughgarden [2010]).

Our main result in this work establishes an improved upper bound for the dominating distribution algorithm.

Theorem 1. *The dominating distribution algorithm suffers at most 2 times more cache misses in expectation compared to the optimal online algorithm in the Markov Paging model.*

In fact, as mentioned in Lund et al. [1994], our guarantee for the performance of the dominating distribution algorithm holds more generally for *pairwise-predictive* distributions. These are distributions for which one can compute any time, for any pair of pages p and q in the cache, the probability that p is next requested before q in the future (conditioned on the most recent request).

While the dominating distribution algorithm is a randomized algorithm, Lund et al. [1994] also propose and study a simple deterministic algorithm for the Markov Paging model, known as the median algorithm. On any cache miss, the median algorithm evicts the page in cache that has the largest median time at which it is requested next. Lund et al. [1994] show that the median algorithm suffers at most 5 times more cache misses in expectation than the optimal online algorithm—to the best of our knowledge, this is the state of the art for deterministic algorithms. As an additional convenient consequence of our improved analysis, we are also able to improve the performance guarantee for the median algorithm.

Theorem 2. *The median algorithm suffers at most 4 times more cache misses in expectation compared to the optimal online algorithm in the Markov Paging model.*

Given the improved performance guarantee in Theorem 1, one can ask: is this the best bound that we can show for the dominating distribution algorithm? Or is there hope to show that it is *the* optimal online algorithm? A detail here is that the dominating distribution algorithm is defined as any algorithm satisfying a specific property (see (4)); in particular, there can exist different dominating distribution algorithms. Our upper bound (Theorem 1) holds uniformly for *all* dominating distribution algorithms. However, our next theorem shows a lower bound that at least one of these is significantly suboptimal.

Theorem 3. *There exists a dominating distribution algorithm that suffers at least 1.5907 times more cache misses in expectation compared to the optimal online algorithm in the Markov Paging model.*

Theorem 3 shows that one cannot hope to show optimality uniformly for all dominating distribution algorithms. To the best of our knowledge, no such lower bounds for the dominating distribution algorithm have been previously shown. While we believe that the bound in Theorem 1 is the correct bound, it is an interesting open direction to close the gap between the upper and lower bound.

Finally, we also consider a setting where the algorithm does not exactly know the Markov chain generating the page requests, but only has access to a dataset of past page requests. In this case, one can use standard estimators from the literature to approximate the transition matrix of the Markov chain from this dataset. But can an approximate estimate of the transition matrix translate to a tight competitive ratio for a paging algorithm, and if so, how large of a dataset does it require to achieve it? The robustness of the analysis[1] of Theorem 1 lets us derive a precise learning-theoretic

---

[1]In fact, we borrow this robustness from the original analysis of Lund et al. [1994].

result ([Theorem 6](#) in [Section 7](#)) showing that the dominating distribution algorithm, when trained on a dataset of size $O(n^2/\varepsilon^2)$, is no more than $\frac{2}{1-2\varepsilon}$ times worse than the optimal online algorithm on a fresh sequence of page requests.

This result is especially relevant in the context of data-driven and learning-augmented algorithms. For instance, by treating the learned Markov chain as a learned oracle and combining it with a worst-case approach, such as the Marker algorithm, similarly to the approach proposed in [Mahdian et al., 2012, Lykouris and Vassilvitskii, 2021], the resulting algorithm achieves $O(1)$-consistency and $O(\log k)$-robustness guarantees for the online paging problem.

Algorithm	Reference	Upper Bound	Lower Bound
Dominating Distribution (Randomized)	Lund et al. [1994]	4 (Theorem 3.5)	-
	This Work	**2** ([Theorem 1](#))	**1.5907** ([Theorem 3](#))
Median (Deterministic)	Lund et al. [1994]	5 (Theorem 2.4)	**1.511**[2] (Theorem 5.4)
	This Work	**4** ([Theorem 2](#))	-

Table 1: State of the art for Markov Paging (best-known bounds are in bold). Both the median and the dominating distribution algorithm are due to Lund et al. [1994].

1.1 Other Related Work

There is a rich literature by this point on studying the online paging problem under assumptions on the permissible page requests that are supposed to model real-world scenarios (see e.g., the excellent surveys by Irani [2005] and Dorrigiv and López-Ortiz [2005]). In particular, several models are motivated by the aforementioned locality of reference principle. The access graph model, proposed originally by Borodin et al. [1991], and developed further in the works of Irani et al. [1992], Fiat and Karlin [1995] and Fiat and Mendel [1997], assumes an underlying graph on the pages, which a request sequence is supposed to abide by. Namely, if a page p_t has been requested at time t, then only the neighbors of p_t in the graph may be requested at time $t + 1$. In this sense, the Markov Paging model can be thought of as a probabilistic variant of the access graph model. Torng [1998] models locality of reference based on the "working sets" concept, introduced by Denning [1968], and also shows how a finite lookahead of the page sequence can be used to obtain improved guarantees. A related form of competitive analysis, proposed by Koutsoupias and Papadimitriou [2000], assumes that there is a family of valid distributions, and page sequences are drawn from some distribution belonging to this family. The goal of an algorithm is then to be simultaneously competitive with the optimal algorithm for every distribution in the family. A particular family of interest suggested by Koutsoupias and Papadimitriou [2000] (which is incomparable to the Markov Paging model) is that of *diffuse adversaries*, which has since been developed further in follow-up works by Young [1998, 2000] and Becchetti [2004]. Finally, Angelopoulos and Schweitzer [2009] study paging under the *bijective analysis* framework, which compares the performance of two algorithms under bijections on page request sequences.

---

[2]Lund et al. [1994] only provide a proof for a lower bound of 1.4, but mention that they also obtained 1.511.

2 Background and Setup

2.1 Markov Paging

The Markov Paging model assumes that the page requests are drawn from a time-homogeneous Markov chain, whose state space is the set of all pages. More precisely, the Markov Paging model is specified by an initial distribution on the n pages, and a transition matrix $M \in [0,1]^{n \times n}$, where $M_{i,j}$ specifies the probability that page j is requested next, conditioned on the most recently requested page being i. The form of the initial distribution will not be too important for our purposes, but for concreteness, we can assume it to be the uniform distribution on the n pages. It is typically assumed that the transition matrix M is completely known to the paging algorithm/page eviction policy—however, the realized page sequence is not. Upon drawing T page requests from the Markov chain, the objective function that a page eviction policy $\mathcal{A}$ seeks to minimize is:

$$\mathbb{E}[\text{Cost}(\mathcal{A}, T)] = \mathbb{E}_{M, \mathcal{A}}\left[\sum_{t=1}^{T} \mathbb{1}[\mathcal{A} \text{ suffers a cache miss at time } t]\right], \tag{1}$$

where the expectation is with respect to both the random page request sequence drawn from M, and the randomness in the policy $\mathcal{A}$. We denote by OPT the policy that minimizes the objective in (1). For $c \geq 1$, an eviction policy $\mathcal{A}$ is said to c-competitive against OPT if for every T, $\mathbb{E}[\text{Cost}(\mathcal{A}, T)] \leq c \cdot \mathbb{E}[\text{Cost}(\text{OPT}, T)]$.

It is worth emphasizing again that the competitive analysis above is with respect to the optimal *online* strategy, in that OPT does not know beforehand the realized sequence of page requests. Contrast this to the best *offline* strategy, which also additionally knows the realized sequence of page requests. As mentioned previously, the Farthest-in-Future policy [Belady, 1966], which at any cache miss, evicts the page that is next requested latest in the remainder of the sequence, is known to be an optimal offline policy.

It is indeed possible to compute OPT exactly—however, the best known ways do this are computationally very expensive, requiring an exponential amount of computation in k.

Theorem 4 (Theorems 1, 2 in Karlin et al. [1992]). *The optimal online policy that minimizes* $\lim_{T \to \infty} \frac{1}{T} \cdot \mathbb{E}[\text{Cost}(\mathcal{A}, T)]$ *over all policies* $\mathcal{A}$ *can be computed exactly by solving a linear program in* $n\binom{n}{k}$ *variables. Furthermore, for any finite* T, *an optimal online policy that minimizes* $\mathbb{E}[\text{Cost}(\mathcal{A}, T)]$ *can be computed in time* $T \cdot n^{\Omega(k)}$.

In the latter case above when T is finite, the optimal online policy can be computed, for example, via dynamic programming (using the "Bellman operator"), where one writes the recurrence for taking a single step of the optimal policy, and unrolls time backwards from T. However, the state space in the dynamic program comprises of all {cache state, requested page} pairs—there are $n^{\Omega(k)}$ many of these. Similarly, the linear program in the case where $T \to \infty$ also comprises of a single variable for each such state. Unfortunately, we do not know of any better algorithms that are both optimal and run in polynomial time in k. Therefore, one seeks efficient polynomial time (in k) algorithms that are approximately optimal, i.e., achieve c-competitiveness with OPT for c as close to 1 as possible.

Towards this, Karlin et al. [1992] showed that several intuitive eviction policies (e.g., on a cache miss, evict the page in cache that has the largest *expected* time to be requested next) are necessarily at least an $\Omega(k)$ factor suboptimal compared to OPT. They then showed a policy $\mathcal{A}$ that

is provably $O(1)$-competitive against OPT; however, the constant in the $O(1)$ is somewhat large. Thereafter, Lund et al. [1999][3] proposed a simple and elegant polynomial-time policy, referred to as the *dominating distribution algorithm*, and showed that it achieves 4-competitiveness against OPT. We describe this policy ahead.

2.2 Dominating Distribution Algorithm

The dominating distribution algorithm, which was proposed by Lund et al. [1999] and is denoted $\mathcal{A}_{\text{dom}}$ hereon, operates as follows. At any page request, if $\mathcal{A}_{\text{dom}}$ suffers a cache miss, it computes a so-called *dominating* distribution μ over the currently-existing pages in the cache. Thereafter, it draws a page $p \sim \mu$, and evicts p. First, we will specify the properties of this dominating distribution.

Intuitively, we want μ to be a distribution such that a *typical* page drawn from μ will be highly likely to be next requested later than *every* other page in the cache. Formally, suppose that the most recently requested page (that caused a cache miss) is s. Fix any two pages p and q (not necessarily in the cache), and let

$$\alpha(p < q|s) := \Pr_M[p \text{ is next requested before } q \mid s \text{ is the most recently requested page}], \quad (2)$$

and let $\alpha(p < p|s) = 0$ for all pages p. Here, the probability is only over the draw of page requests from the Markov chain. The claim is that these $\alpha(p < q|s)$ values can be computed efficiently, in total time $O(n^4)$ for all pairs p, q in cache, provided that we know the transition matrix M. To see this, let $x_s = \alpha(p < q|s)$. Note that $x_p = 1$, $x_q = 0$. For any other page $r \neq p, q$, note that

$$x_r = M_{r,1}x_1 + \cdots + M_{r,n}x_n.$$

Thus, upon solving the linear system given by

$$\begin{bmatrix} M_{1,1} - 1 & M_{1,2} & \dots & M_{1,n} \\ M_{2,1} & M_{2,2} - 1 & \dots & M_{1,n} \\ \vdots & \vdots & \dots & \vdots \\ e_p & & & \\ \vdots & \vdots & \dots & \vdots \\ e_q & & & \\ \vdots & \vdots & \dots & \vdots \\ M_{n,1} & M_{n,2} & \dots & M_{n,n} - 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \\ \vdots \\ x_q \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad (3)$$

where e_p and e_q are row vectors with a 1 in the pth and q^{th} coordinates, respectively, and 0 elsewhere, x_s is the desired value $\alpha(p < q|s)$. In this way, we can solve a linear system for each of the $\leq n^2$ possible pairs for q, p, and compute all $\alpha(p < q|s)$ values in time $O(n^4)$. For notational convenience, we will hereafter frequently denote $\alpha(p < q|s)$ simply by $\alpha(p < q)$, implicitly assuming conditioning on the most frequently requested page.

Equipped with all these $\alpha(p < q)$ values, let us define μ to be a distribution over the pages in the cache satisfying the following property: for every fixed page q in the cache,

$$\mathbb{E}_{p \sim \mu}[\alpha(p < q)] \leq \frac{1}{2}. \quad (4)$$

---

[3]We reference the journal version [Lund et al., 1999] in lieu of the conference version [Lund et al., 1994] hereafter.

This is the required dominating distribution. What may be somewhat surprising here is that such a dominating distribution, which satisfies the condition in (4) provably *always* exists, and furthermore can be constructed efficiently by solving a linear program with just $O(k)$ variables (i.e., in $\text{poly}(k)$ time).

Theorem 5 (Theorem 3.4 in Lund et al. [1999])**.** *A distribution μ satisfying the condition in (4) necessarily exists and can be computed by solving a linear program in $O(k)$ variables.*

Thus, the total computation required by $\mathcal{A}_{\text{dom}}$ at each cache miss is only $\text{poly}(k)$ (to solve the linear program that constructs μ), assuming all $\alpha(p < q)$ values are precomputed initially. In summary, the overall computation cost of $\mathcal{A}_{\text{dom}}$ across T page requests is at most $T \cdot \text{poly}(n, k)$.

It remains to argue that if $\mathcal{A}_{\text{dom}}$ evicts $p \sim \mu$ whenever it suffers a cache miss, then it fares well in comparison to OPT. A convenient observation towards showing this is the following claim, whose proof follows from the definition of μ:

Claim 2.1. *Let s be the most recently requested page. The dominating distribution μ satisfies the following property: for every fixed page q in the cache, we have that*

$$\Pr_{p \sim \mu, M}[q \text{ is next requested no later than } p \mid s \text{ is the most recently requested page}] \geq \frac{1}{2}. \quad (5)$$

Proof. TOPROVE 0 □

Using this property of μ, Lund et al. [1999] show that $\mathcal{A}_{\text{dom}}$ incurs only 4 times as many cache misses as OPT in expectation. The following section provides this part of their result.

3   4-competitive Analysis of Lund et al. [1999]

We restate and provide a more detailed proof of the crucial technical lemma from Lund et al. [1999], which is particularly useful for our improved 2-competitive analysis in Section 5.

Lemma 3.1 (Lemma 2.5 in Lund et al. [1999])**.** *Let $\mathcal{A}$ be a paging algorithm. Suppose that whenever $\mathcal{A}$ suffers a cache miss, the page p that it chooses to evict is chosen from a distribution such that the following property is satisfied: for every page q in the cache, the probability (over the choice of p and the random sequence ahead, conditioned on the most recently requested page) that q is next requested no later than the next request for p is at least $1/c$. Then $\mathcal{A}$ is $2c$-competitive against OPT.*

Proof. TOPROVE 1 □

Corollary 3.2. *The dominating distribution algorithm $\mathcal{A}_{\text{dom}}$ is 4-competitive against OPT.*

Proof. TOPROVE 2 □

Remark 1. Consider setting $c = 1$ in Lemma 3.1: this corresponds to $\mathcal{A}$ effectively being an *offline* algorithm, which *knows* which page in the cache is going to next be requested farthest in the future compared to all the other pages in the cache. We already know that such an algorithm is optimal, i.e., is 1-competitive against OPT. However, the guarantee given by Lemma 3.1 for such an algorithm is still only 2-competitiveness. This at least suggests that there is scope to improve the guarantee to c-competitiveness.

4 Sources of Looseness in the Analysis

We systematically identify the sources of looseness in the above analysis with illustrative examples, before proceeding to individually tighten them in the subsequent section.

4.1 A Conservative Approach to Handling Doubly-Charged Pages

Recall that in the summation $\sum_{t \leq T} \beta(t)$ above, a cache miss for OPT may be double-counted if the same page is assigned two charges. Although one way to address this overcounting is to divide the summation by 2, this approach is overly conservative, as it *under*counts every cache miss in OPT that results from a request to a singly-charged page. For instance, consider the following situation:
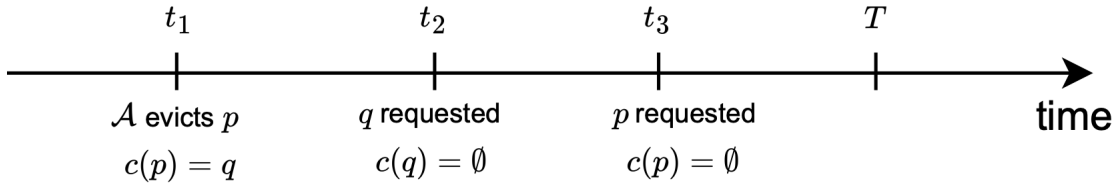

Figure 1: Request for q (a singly charged page) at t_2 constitutes a unit cache miss for OPT, but the division by 2 undercounts this (and every other singly-charged cache miss in OPT).

Here, at time t_1, $\mathcal{A}$ evicts a page p. Suppose that p continues to live in the cache of OPT after the request at t_1, and hence $c(p)$ is assigned to some q in $\mathcal{A}$'s cache. Now, say that q is requested at time t_2, which is before $t_3 = r(t_1, p)$. Furthermore, assume that at t_2, q only had the single charge on itself (by p), i.e., q was not evicted by $\mathcal{A}$ in the time between t_1 and t_2. Then, $\beta(t_1) = 1$, but we are wastefully dividing it by 2 in our calculation.

4.2 Inexhaustive Clearing of Charges upon a Page Request

Consider the first item in the charging scheme (**??**)—whenever a page s is requested, any charges that s might be *giving* are cleared (i.e., $c(s) = \emptyset$). Intuitively, this is supposed to account for the fact that, while s was holding some page $c(s)$ responsible for being requested before s itself, either this did happen, in which case we happily clear the already-paid charge, or this did not quite happen and s got requested before $c(s)$, in which case we should let go of the charge and focus on the future. However, consider instead the case where s does not have a charge on any other page, but is itself the bearer of a charge by some other page, say p. In this case, s has successfully paid the charge that was expected of it—*but this charge would only be cleared upon the next request to p!* If it so happens that p is next requested *after* the time horizon T, then even if s successfully paid the charge due to p, since this charge was not cleared when it was requested, it would be counted in $\mathcal{O}$ as part of the open charges post time T, and wastefully subtracted in the calculation. This is concretely illustrated in the situation below:
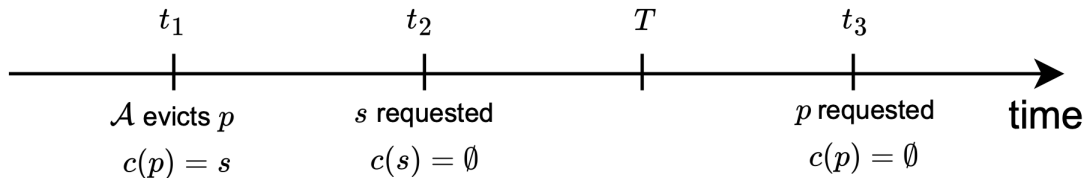
Figure 2: Here, $\beta(t_1) = 1$, and OPT suffers a cache miss at t_2. However, p still holds a charge on s at time T because it is requested at $t_3 > T$. Thus, this charge is included in $\mathcal{O}$, canceling out the contribution due to the cache miss at t_2.

This suggests that whenever a page is requested, we should clear not only the charges it gives but also any charges it *bears*, preventing unnecessary inclusion in $\mathcal{O}$. This approach ensures that a page discards its imposed charge immediately upon "paying it off" rather than with a delay.

4.3 No Accounting for Uncharged, Non-first-timer Pages

Finally, observe that in the accounting of cache misses for OPT, we only count those that occur due to requests to charged/savior pages (as counted by the $\beta(t)$'s), and those that occur due to first-time requests to pages not initially in the cache (as counted by $\mathcal{I}$). However, OPT can also suffer cache misses due to a request to an *uncharged* page. Namely, if there is a request to a page q that was previously evicted by OPT, but at the time that it is requested, q is bearing no charges at all, then the cache miss due to q is not being counted in the calculation. This is illustrated by the scenario below:

Figure 3: At t_3, q does not have any charges on it, but still causes a cache miss for OPT.

$\mathcal{A}$ evicts p at time t_1 and assigns a charge to $q \neq p$, implying that q is not in the cache of OPT after the request at t_1 (but was instead previously evicted by OPT). Next, at t_2, p is requested, and this clears the charge it had on q. Since p got requested before q, $\beta(t_1) = 0$. Then, at t_3, q is requested—q has no charges on it at this point. Notice that this still causes a cache miss for OPT at t_3. However, this cache miss is not accounted for in our calculation, either by $\beta(t_1)$, or by $\mathcal{I}$ (since this is not the first time that q is being brought into the cache).

5 Tightening the Analysis to 2-competitiveness

Having identified the loose ends in the analysis of Lund et al. [1999] above, we are able to tighten their analysis and prove the following lemma:

Lemma 5.1. *Let $\mathcal{A}$ be a paging algorithm. Suppose that whenever $\mathcal{A}$ suffers a cache miss, the page p that it chooses to evict is chosen from a distribution such that the following property is satisfied:*

9

for every page q in the cache, the probability (over the choice of p and the random sequence ahead, conditioned on the most recently requested page) that q is next requested no later than the next request for p is at least $1/c$. Then $\mathcal{A}$ is c-competitive against OPT.

Proof. TOPROVE 3 □

Corollary 5.2. *The dominating distribution algorithm $\mathcal{A}_{\mathrm{dom}}$ is 2-competitive against* OPT.

Proof. TOPROVE 4 □

Remark 2. As in Remark 1, if we set $c = 1$ in Lemma 5.1, our lemma says that a policy that has essentially seen the future and evicts the page that is next scheduled to be requested latest is 1-competitive against OPT, i.e., it is optimal. Thus, our analysis, while establishing the best-possible guarantee for algorithms satisfying the condition of the lemma, additionally recovers an alternate, charging-scheme-based proof of the optimality of the Farthest-in-Future eviction policy.

Remark 3. The factor of 4 in the analysis of Lund et al. [1994] constituted a factor of 2 arising due to doubly-charged pages, and a factor of 2 arising from the property of the dominating distribution algorithm. While we got rid of the first factor, the second factor appears to be necessary, stemming from an inherent property (Equation (4)) of the algorithm itself. It would indeed be very interesting to see if this is not the case, and if the upper bound could further be improved from 2.

Lemma 5.1 also allows us to improve the approximation guarantee for another intuitive *deterministic* algorithm considered by Lund et al. [1999], namely the *median* algorithm. At any cache miss, the median algorithm evicts the page in cache that has the largest median time of next request. For deterministic algorithms satisfying the condition of Lemma 3.1, Lund et al. [1999] employ a slightly more specialized analysis of the charging scheme in **??** (Lemma 2.3 in Lund et al. [1999]) to obtain a $(c + 1)$-competitive guarantee against OPT (instead of $2c$-competitiveness). Thereafter, by arguing that the median algorithm satisfies the condition with $c = 4$, (Theorem 2.4 in Lund et al. [1999]), they are able to show that the median algorithm is 5-competitive against OPT. Our tighter analysis in Lemma 5.1 applies to any algorithm (deterministic/randomized), and hence also improves the guarantee for deterministic algorithms obtained by Lund et al. [1999]. More importantly, it improves the performance guarantee that we can state for the median algorithm.

Corollary 5.3. *The median algorithm is 4-competitive against* OPT.

6 Lower bound for Dominating Distribution Algorithms

In this section, we show that there exist problem instances where a dominating distribution algorithm[4] can provably be at least c times worse than OPT, for $c \geq 1.5907$. We start with a warm-up analysis that captures the central idea behind the lower bound, and shows a bound of 1.5, before generalizing the analysis to improve the bound. Our lower bound instances throughout are small and simple: they comprise of a Markov chain on $n = 3$ pages with a cache size $k = 2$.

---

[4]Note that we cannot hope to prove a lower bound that applies to *all* dominating distribution algorithms, since the optimal Farthest-in-Future algorithm is technically also a dominating distribution algorithm.

6.1 Warm-up: A 1.5-competitive Lower Bound

Consider a simple setting with 3 total pages, and a cache of size 2. At each time step independently, page 1 is requested with probability $1 - \varepsilon$, and page 2 and 3 are requested each with probability $\varepsilon/2$. This corresponds to the Markov chain with the transition matrix

$$\begin{bmatrix} 1 - \varepsilon & \varepsilon/2 & \varepsilon/2 \\ 1 - \varepsilon & \varepsilon/2 & \varepsilon/2 \\ 1 - \varepsilon & \varepsilon/2 & \varepsilon/2 \end{bmatrix}.$$

We start with the initial cache being $[1, 2]$. The reference algorithm $\mathcal{A}_{\mathrm{ref}}$ we will compete against always keeps page 1 in its cache. If it suffers a cache miss, it swaps out the other page with the requested page. At any time t, the probability that $\mathcal{A}_{\mathrm{ref}}$ suffers a miss is therefore exactly $\varepsilon/2$, and hence the expected total number of misses through T timesteps is $\varepsilon T/2$.

Now, consider $\mathcal{A}_{\mathrm{dom}}$. We want to capitalize on the fact that $\mathcal{A}_{\mathrm{dom}}$ evicts page 1 with some positive probability, in contrast to $\mathcal{A}_{\mathrm{ref}}$. Namely, observe that

$$\Pr[\mathcal{A}_{\mathrm{dom}} \text{ suffers cache miss at time } t] = \Pr[1 \text{ in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t] \cdot \frac{\varepsilon}{2} +$$
$$\Pr[1 \text{ not in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t] \cdot (1 - \varepsilon)$$
$$= p_t \cdot \frac{\varepsilon}{2} + (1 - p_t) \cdot (1 - \varepsilon), \tag{6}$$

where we denote $p_t = \Pr[1 \text{ in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t]$. The latter probability is non-zero for $\mathcal{A}_{\mathrm{dom}}$, which already makes its expected total number of misses larger than that of $\mathcal{A}_{\mathrm{ref}}$. We will therefore aim to maximize this difference.

Observe that

$$p_t = \Pr[1 \text{ requested at } t - 1] + \Pr[1 \text{ in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t - 1] \cdot \Pr[1 \text{ not requested at } t - 1] \cdot$$
$$\Pr[1 \text{ not evicted by } \mathcal{A}_{\mathrm{dom}} \text{ at } t - 1 \mid 1 \text{ not requested}, 1 \text{ in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t - 1]$$
$$= 1 - \varepsilon + p_{t-1} \cdot \varepsilon \cdot \Pr[1 \text{ not evicted by } \mathcal{A}_{\mathrm{dom}} \text{ at } t - 1 \mid 1 \text{ not requested}, 1 \text{ in cache of } \mathcal{A}_{\mathrm{dom}} \text{ at } t - 1].$$

We can analytically calculate the last probability. Say the cache of $\mathcal{A}_{\mathrm{dom}}$ at $t - 1$ is, without loss of generality, $[1, 2]$. Given that 1 is not requested, we have that either of 2 or 3 are requested, each with probability $1/2$. If 2 is requested, there is no cache miss. If on the other hand, 3 is requested, $\mathcal{A}_{\mathrm{dom}}$ calculates a dominating distribution μ over $[1, 2]$. This can be easily calculated. Let $\alpha(p < q)$ again be the probability that p is requested next before q (conditioned on the current page, which in this case, does not matter since the distributions at every time step are independent and identical). We have that

$$\alpha(1 < 1) = \alpha(2 < 2) = 0,$$
$$\alpha(2 < 1) = \varepsilon/2 + \varepsilon/2 \cdot \alpha(2 < 1) \implies \alpha(2 < 1) = \frac{\varepsilon/2}{1 - \varepsilon/2},$$
$$\alpha(1 < 2) = 1 - \varepsilon + \varepsilon/2 \cdot \alpha(1 < 2) \implies \alpha(1 < 2) = \frac{1 - \varepsilon}{1 - \varepsilon/2}.$$

The condition for the dominating distribution is that, for every fixed page q in the cache,

$$\mathbb{E}_{p \sim \mu}[\alpha(p < q)] \leq 1/2.$$

11

Letting $\mu(1) = x$ so that $\mu(2) = 1 - x$, this translates to

$$(1-x) \cdot \alpha(2 < 1) \leq 1/2, \quad x \cdot \alpha(1 < 2) \leq 1/2$$

$$\implies \quad \frac{3\varepsilon - 2}{2\varepsilon} \leq x \leq \frac{2-\varepsilon}{4-4\varepsilon}.$$

We want x to be large as possible, because we want $\mathcal{A}_{\text{dom}}$ to evict 1 with the highest feasible probability, so that it will suffer a lot of cache misses. Thus, we set $x = \frac{2-\varepsilon}{4-4\varepsilon}$. This gives us that,

$$\Pr[1 \text{ not evicted by } \mathcal{A}_{\text{dom}} \text{ at } t-1 \mid 1 \text{ not requested, 1 in cache of } \mathcal{A}_{\text{dom}} \text{ at } t-1]$$
$$= \frac{1}{2} + \frac{1}{2} \cdot (1-x) = 1 - \frac{x}{2} = \frac{6-7\varepsilon}{8-8\varepsilon}.$$

In total, we get that

$$p_t = \underbrace{1-\varepsilon}_{c} + p_{t-1} \cdot \underbrace{\frac{\varepsilon(6-7\varepsilon)}{8-8\varepsilon}}_{d}.$$

Unrolling the recurrence, and using that $p_1 = 1$, we obtain that for $t \geq 2$,

$$p_t = c(1 + d + \cdots + d^{t-2}) + d^{t-1} = c \cdot \frac{1 - d^{t-1}}{1-d} + d^{t-1}.$$

Summing up (6) until T, and substituting p_t from above, we get that

$$\mathbb{E}[\text{Cost}(\mathcal{A}_{\text{dom}}, T)] = T(1-\varepsilon) + \left(\frac{3\varepsilon}{2} - 1\right) \sum_{t=1}^{T} p_t$$

$$= T(1-\varepsilon) + \left(\frac{3\varepsilon}{2} - 1\right) \left[\frac{c}{1-d} \left(T - \frac{1-d^T}{1-d}\right) + \frac{1-d^T}{1-d}\right].$$

Taking the limit as $T \to \infty$, we get that

$$\lim_{T \to \infty} \frac{\mathbb{E}[\text{Cost}(\mathcal{A}_{\text{dom}}, T)]}{\mathbb{E}[\text{Cost}(\mathcal{A}_{\text{ref}}, T)]} = \frac{1 - \varepsilon + \left(\frac{3\varepsilon}{2} - 1\right)\left(\frac{c}{1-d}\right)}{\varepsilon/2}.$$

Substituting the values of c and d, and taking the limit as $\epsilon \to 0$, the ratio above converges to 1.5.

6.2 A Slightly More General Analysis that Gives a Better Lower Bound

In the instance above, instead of assigning $\varepsilon/2$ mass equally to pages 2 and 3, we can also imagine splitting the ε mass unequally between these pages. Namely, let the underlying distribution on the pages be such that, at each time step, page 1 is requested with probability $1 - \varepsilon$, page 2 is requested with probability ε_1 and page 3 is requested with probability $\varepsilon - \varepsilon_1$, where the regime we want to keep in mind is $1 - \varepsilon \geq \varepsilon_1 \geq \varepsilon - \varepsilon_1$. Note that $\varepsilon_1 = \varepsilon/2$ corresponds to the instance above. Let the initial cache be $[1, 2]$.

The reference algorithm $\mathcal{A}_{\text{ref}}$ still always maintains page 1 in its cache. Let

$$p_t = \Pr[\mathcal{A}_{\text{ref}}\text{'s state is } [1, 2] \text{ at } t].$$

We have that

$$p_t = p_{t-1} \cdot (1 - \varepsilon + \varepsilon_1) + (1 - p_{t-1}) \cdot \varepsilon_1 = \varepsilon_1 + p_{t-1} \cdot (1 - \varepsilon) = \varepsilon_1 \left(\frac{1 - (1 - \varepsilon)^{t-1}}{\varepsilon} \right) + (1 - \varepsilon)^{t-1},$$

where we unrolled the recurrence in the last step, and used that $p_1 = 1$. Then,

$$\Pr[\mathcal{A}_{\text{ref}} \text{ suffers miss at } t] = p_t \cdot (\varepsilon - \varepsilon_1) + (1 - p_t) \cdot \varepsilon_1 = \varepsilon_1 + (\varepsilon - 2\varepsilon_1) \cdot p_t,$$

and hence

$$\mathbb{E}[\text{Cost}(\mathcal{A}_{\text{ref}}, T)] = T\varepsilon_1 + (\varepsilon - 2\varepsilon_1) \sum_{t=1}^{T} p_t$$

$$= T\varepsilon_1 + (\varepsilon - 2\varepsilon_1) \left[\frac{\varepsilon_1}{\varepsilon} \left(T - \frac{1 - (1 - \varepsilon)^T}{\varepsilon} \right) + \frac{1 - (1 - \varepsilon)^T}{\varepsilon} \right]. \tag{7}$$

Now, we analyze the behavior of $\mathcal{A}_{\text{dom}}$. Let

$$p_t = \Pr[\mathcal{A}_{\text{dom}}\text{'s state is } [1, 2] \text{ at } t]$$
$$q_t = \Pr[\mathcal{A}_{\text{dom}}\text{'s state is } [1, 3] \text{ at } t]$$
$$r_t = \Pr[\mathcal{A}_{\text{dom}}\text{'s state is } [2, 3] \text{ at } t],$$

where $p_1 = 1, q_1 = 0, r_1 = 0$. Then, we have that

$$\Pr[\mathcal{A}_{\text{dom}} \text{ suffers miss at } t] = p_t \cdot (\varepsilon - \varepsilon_1) + q_t \cdot \varepsilon_1 + r_t \cdot (1 - \varepsilon). \tag{8}$$

Observe further that

$$p_t = p_{t-1} \cdot \Pr[1 \text{ or } 2 \text{ requested at } t - 1] + q_{t-1} \cdot \varepsilon_1 \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 3 \text{ from } [1, 3]]$$
$$+ r_{t-1} \cdot (1 - \varepsilon) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 3 \text{ from } [2, 3]]$$
$$= p_{t-1} \cdot (1 - \varepsilon + \varepsilon_1) + q_{t-1} \cdot \varepsilon_1 \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 3 \text{ from } [1, 3]]$$
$$+ r_{t-1} \cdot (1 - \varepsilon) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 3 \text{ from } [2, 3]]$$
$$q_t = q_{t-1} \cdot \Pr[1 \text{ or } 3 \text{ requested at } t - 1] + p_{t-1} \cdot (\varepsilon - \varepsilon_1) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 2 \text{ from } [1, 2]]$$
$$+ r_{t-1} \cdot (1 - \varepsilon) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 2 \text{ from } [2, 3]]$$
$$= q_{t-1} \cdot (1 - \varepsilon_1) + p_{t-1} \cdot (\varepsilon - \varepsilon_1) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 2 \text{ from } [1, 2]]$$
$$+ r_{t-1} \cdot (1 - \varepsilon) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 2 \text{ from } [2, 3]]$$
$$r_t = r_{t-1} \cdot \Pr[2 \text{ or } 3 \text{ requested at } t - 1] + p_{t-1} \cdot (\varepsilon - \varepsilon_1) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 1 \text{ from } [1, 2]]$$
$$+ q_{t-1} \cdot \varepsilon_1 \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 1 \text{ from } [1, 3]]$$
$$= r_{t-1} \cdot \varepsilon + p_{t-1} \cdot (\varepsilon - \varepsilon_1) \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 1 \text{ from } [1, 2]] + q_{t-1} \cdot \varepsilon_1 \cdot \Pr[\mathcal{A}_{\text{dom}} \text{ evicts } 1 \text{ from } [1, 3]].$$

We can calculate each of the eviction probabilities above while respecting the properties that $\mathcal{A}_{\text{dom}}$ must satisfy. Namely, consider that the cache state is $[1, 2]$. Let μ be the dominating distribution that $\mathcal{A}_{\text{dom}}$ constructs, where $\mu(1) = x$ and $\mu(2) = 1 - x$. We have that

$$\alpha(2 < 1) = \varepsilon_1 + (\varepsilon - \varepsilon_1)\alpha(2 < 1) \implies \alpha(2 < 1) = \frac{\varepsilon_1}{1 - \varepsilon + \varepsilon_1}$$

$$\alpha(1 < 2) = 1 - \alpha(2 < 1) = \frac{1 - \varepsilon}{1 - \varepsilon + \varepsilon_1}.$$

The dominating distribution condition requires that

$$(1-x) \cdot \alpha(2 < 1) \le \frac{1}{2}, \quad x \cdot \alpha(1 < 2) \le \frac{1}{2}$$

$$\implies \quad \frac{\alpha(2 < 1) - 1/2}{\alpha(2 < 1)} \le x \le \frac{1}{2\alpha(1 < 2)}.$$

We choose to set the upper condition to be tight, which results in

$$x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 1 \text{ from } [1,2]] = \frac{1 - \varepsilon + \varepsilon_1}{2 - 2\varepsilon},$$

$$1 - x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 2 \text{ from } [1,2]] = \frac{1 - \varepsilon - \varepsilon_1}{2 - 2\varepsilon}.$$

Now, consider the same calculation, where the cache state is $[1,3]$, $\mu(1) = x$ and $\mu(3) = 1 - x$. We have that

$$\alpha(3 < 1) = \varepsilon - \varepsilon_1 + \varepsilon_1 \cdot \alpha(3 < 1) \implies \alpha(3 < 1) = \frac{\varepsilon - \varepsilon_1}{1 - \varepsilon_1}$$

$$\alpha(1 < 3) = 1 - \alpha(3 < 1) = \frac{1 - \varepsilon}{1 - \varepsilon_1}.$$

The dominating distribution condition requires

$$\frac{\alpha(3 < 1) - 1/2}{\alpha(3 < 1)} \le x \le \frac{1}{2\alpha(1 < 3)}.$$

Again, we make the upper bound tight to get

$$x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 1 \text{ from } [1,3]] = \frac{1 - \varepsilon_1}{2 - 2\varepsilon},$$

$$1 - x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 3 \text{ from } [1,3]] = \frac{1 + \varepsilon_1 - 2\varepsilon}{2 - 2\varepsilon}.$$

Finally, consider the calculation, where the cache state is $[2,3]$, $\mu(2) = x$ and $\mu(3) = 1 - x$. We have that

$$\alpha(3 < 2) = \varepsilon - \varepsilon_1 + (1 - \varepsilon) \cdot \alpha(3 < 2) \implies \alpha(3 < 2) = \frac{\varepsilon - \varepsilon_1}{\varepsilon}$$

$$\alpha(2 < 3) = 1 - \alpha(3 < 2) = \frac{\varepsilon_1}{\varepsilon}.$$

The dominating distribution condition requires

$$\frac{\alpha(3 < 2) - 1/2}{\alpha(3 < 2)} \le x \le \frac{1}{2\alpha(2 < 3)},$$

so we set

$$x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 2 \text{ from } [2,3]] = \frac{\varepsilon}{2\varepsilon_1},$$

$$1 - x = \Pr[\mathcal{A}_{\mathrm{dom}} \text{ evicts } 3 \text{ from } [2,3]] = \frac{2\varepsilon_1 - \varepsilon}{2\varepsilon_1}.$$

Substituting all these eviction probabilities, and writing the system of equations for p_t, q_t, r_t above in matrix form, we obtain

$$
\begin{bmatrix} p_t \\ q_t \\ r_t \end{bmatrix} = \begin{bmatrix} 1 - \varepsilon + \varepsilon_1 & \varepsilon_1 \cdot \frac{1+\varepsilon_1 - 2\varepsilon}{2-2\varepsilon} & (1-\varepsilon) \cdot \frac{2\varepsilon_1 - \varepsilon}{2\varepsilon_1} \\ (\varepsilon - \varepsilon_1) \cdot \frac{1-\varepsilon-\varepsilon_1}{2-2\varepsilon} & 1 - \varepsilon_1 & (1-\varepsilon) \cdot \frac{\varepsilon}{2\varepsilon_1} \\ (\varepsilon - \varepsilon_1) \cdot \frac{1-\varepsilon+\varepsilon_1}{2-2\varepsilon} & \varepsilon_1 \cdot \frac{1-\varepsilon_1}{2-2\varepsilon} & \varepsilon \end{bmatrix} \begin{bmatrix} p_{t-1} \\ q_{t-1} \\ r_{t-1} \end{bmatrix}
$$

$$
= B^{t-1} \begin{bmatrix} p_1 \\ q_1 \\ r_1 \end{bmatrix} = B^{t-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} ,
$$

where we denoted the matrix by B and rolled out the recurrence. Substituting in (8), we have that

$$
\Pr[\mathcal{A}_{\mathrm{dom}} \text{ suffers miss at } t] = \begin{bmatrix} \varepsilon - \varepsilon_1 & \varepsilon_1 & 1 - \varepsilon \end{bmatrix} B^{t-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} ,
$$

and thus,

$$
\mathbb{E}[\mathrm{Cost}(\mathcal{A}_{\mathrm{dom}}, T)] = \begin{bmatrix} \varepsilon - \varepsilon_1 & \varepsilon_1 & 1 - \varepsilon \end{bmatrix} (I + B + B^2 + \cdots + B^{T-1}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} . \tag{9}
$$

Now that we have exact expressions given by (7) and (9), we can do a computer search over assignments to $T, \varepsilon, \varepsilon_1$—each assignment gives a lower bound for $\frac{\mathbb{E}[\mathrm{Cost}(\mathcal{A}_{\mathrm{dom}}, T)]}{\mathbb{E}[\mathrm{Cost}(\mathcal{A}_{\mathrm{ref}}, T)]}$. For example, our (admittedly limited) search yields

$$
\frac{\mathbb{E}[\mathrm{Cost}(\mathcal{A}_{\mathrm{dom}}, T)]}{\mathbb{E}[\mathrm{Cost}(\mathcal{A}_{\mathrm{ref}}, T)]} \geq 1.5907,
$$

for $\varepsilon = 10^{-5}, T = 10^8, \varepsilon_1 = 0.7069\varepsilon$.

Remark 4. While we do believe that further tuning $\varepsilon, \varepsilon_1$ and T will likely not improve the lower bound by too much, it is conceivable that a better lower bound may be obtained by going beyond $n = 3, k = 2$ (albeit with messier calculations). For example, there appears to be scope for improvement within instances having the same blueprint as above for slightly larger values of n and k, where there are certain high-probability pages that $\mathcal{A}_{\mathrm{ref}}$ always keeps in the cache.

7 Learning the Markov Chain from Samples

It is worth pointing out that the form of Lemma 5.1 allows us to obtain performance guarantees even when we only have approximate estimates of the $\alpha(p < q)$ values, as also noted in passing by Lund et al. [1999]. In particular, suppose that we only have ε-approximate (multiplicate or additive) estimates $\hat{\alpha}(p < q)$ of $\alpha(p < q)$, that still satisfy $\hat{\alpha}(p < q) + \hat{\alpha}(q < p) = 1$, and $\hat{\alpha}(p < p) = 0$. Suppose that at each cache miss, we use these $\hat{\alpha}(p < q)$ values to compute a dominating distribution $\hat{\mu}$, and draw the page p to evict from $\hat{\mu}$.

Even with such an approximate dominating distribution, we can still guarantee either $\frac{2}{1-2\varepsilon}$-competitiveness (in the case that $\hat{\alpha}(p < q) \in [\alpha(p < q) - \varepsilon, \alpha(p < q) + \varepsilon]$), or $\frac{2-2\varepsilon}{1-2\varepsilon}$-competitiveness

(in the case that $\hat{\alpha}(p < q) \in [(1-\varepsilon)\alpha(p < q), (1+\varepsilon)\alpha(p < q)]$). This follows as a direct consequence of Claim 2.1 and Lemma 5.1.

Perhaps the first scenario to consider here is when we do not assume prior knowledge of the transition matrix M, but only have access to samples previously drawn from the Markov chain. For example, we can imagine that the algorithm designer has available a large training dataset of past page requests. This dataset can be used to derive estimates of the entries in the transition matrix M, for instance, using one of the several existing estimators [Hao et al., 2018, Wolfer and Kontorovich, 2021, Huang and Li, 2024]. In fact, recall that the computation of the $\alpha(p < q)$ values requires only for us to solve a linear system determined by the entries in the transition matrix (Equation (3)). Hence, if we have accurate estimates for the entries in M, we can use the theory of perturbed linear systems to bound the resulting error in the estimates of the $\alpha(p < q)$ values. Thereafter, using the argument from the previous paragraph, we can obtain a performance guarantee for the competitiveness of a dominating distribution algorithm that uses these estimates.

We can turn the above informal argument into a formal learning-theoretic result. Formally, suppose that every entry in M is at least some $\delta > 0$. In this case, the Markov chain is irreducible (meaning that there is a positive, and unique probability of eventually getting from any state to another). Thus, in this case, for every fixed p, q, the linear system (3) has a (unique) solution, meaning that matrix associated with it is invertible. The relevant quantity here will be the conditioning of the *worst* linear system that we might ever have to solve. Towards this, let $L_{p,q}$ be the matrix associated with the linear system in (3) for pages p, q, and define:

$$\gamma = \sup_{p \neq q} \|L_{p,q}^{-1}\|_\infty, \tag{10}$$

where for a square matrix $A \in \mathbb{R}^{n \times n}$, $\|A\|_\infty = \sup_i \sum_j |A_{i,j}|$. Observe from (3) that $1 \leq \|L_{p,q}\|_\infty \leq 2$ for every p, q.

We choose to adopt the following result of Hao et al. [2018] for our exposition:

Theorem 6 (Theorem 4 in Hao et al. [2018])**.** *Suppose we unroll a Markov chain to obtain m samples $X_1, \ldots, X_m$, where the initial distribution of X_1 is arbitrary, but the transition matrix $M \in [0,1]^{n \times n}$ of the Markov chain satisfies $M_{i,j} \geq \delta > 0$ for every i, j. Then, there exists an estimator $\hat{M} = \hat{M}(X_1, \ldots, X_m)$ that satisfies*

$$\mathbb{E}_{X_1,\ldots,X_m} \|M(i,:) - \hat{M}(i,:)\|_2^2 \leq O\left(\frac{1}{m\delta}\right)$$

for every $i \in [n]$, where $M(i,:)$ (respectively $\hat{M}(i,:)$) denotes the ith row of M (respectively $\hat{M}$).

Using this theorem, we can obtain the following result:

Theorem 7. *Suppose that the page requests are generated from an unknown Markov chain where every entry in the transition matrix M is at least $\delta > 0$. Let γ be as defined in (10). Given a training dataset of $m = O\left(\frac{\gamma^2 n^2}{\varepsilon^2 \delta}\right)$ past page requests from M, there is an algorithm $\mathcal{A}$ which is $\frac{2}{1-2\varepsilon}$-competitive against OPT with probability at least 0.99 over the m samples.*

Proof. TOPROVE 5 □

16

Acknowledgements

References

S. Angelopoulos and P. Schweitzer. Paging and list update under bijective analysis. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1136–1145. SIAM, 2009.

L. Becchetti. Modeling locality: A probabilistic analysis of lru and fwf. In *Algorithms–ESA 2004: 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004. Proceedings 12*, pages 98–109. Springer, 2004.

L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems journal*, 5(2):78–101, 1966.

A. Borodin and R. El-Yaniv. *Online computation and competitive analysis.* cambridge university press, 2005.

A. Borodin, P. Raghavan, S. Irani, and B. Schieber. Competitive paging with locality of reference. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 249–259, 1991.

P. J. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5): 323–333, 1968.

R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News*, 36(3):67–81, 2005.

R. S. Falk. Math 574 Numerical Analysis Lecture Notes 3: Perturbation theory for linear systems of equations. https://ow3.math.rutgers.edu/~falk/math574/lecture3.pdf, 2016.

A. Fiat and A. R. Karlin. Randomized and multipointer paging with locality of reference. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 626–634, 1995.

A. Fiat and M. Mendel. Truly online paging with locality of reference. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 326–335. IEEE, 1997.

Y. Hao, A. Orlitsky, and V. Pichapati. On learning markov chains. *Advances in Neural Information Processing Systems*, 31, 2018.

D. Huang and X. Li. Non-asymptotic estimates for markov transition matrices with rigorous error bounds. *arXiv preprint arXiv:2408.05963*, 2024.

S. Irani. Competitive analysis of paging. *Online Algorithms: The State of the Art*, pages 52–73, 2005.

S. Irani, A. R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 228–236, 1992.

A. Karlin, S. Phillips, and P. Raghavan. Markov paging. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 208–217, 1992.

E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.

C. Lund, S. Phillips, and N. Reingold. IP over connection-oriented networks and distributional paging. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 424–434. IEEE, 1994.

C. Lund, S. Phillips, and N. Reingold. Paging against a distribution and IP networking. *Journal of Computer and System Sciences*, 58(1):222–231, 1999.

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

M. Mahdian, H. Nazerzadeh, and A. Saberi. Online optimization with uncertain information. *ACM Transactions on Algorithms (TALG)*, 8(1):1–29, 2012.

T. Roughgarden. CS 369N Beyond Worst-Case Analysis, Lecture Notes 2: Models of Data in Online Paging. https://timroughgarden.org/f11/l2.pdf, 2010.

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

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

G. Wolfer and A. Kontorovich. Statistical estimation of ergodic markov chain kernel over discrete state space. *Bernoulli*, 27(1):532–553, 2021.

N. E. Young. Bounding the diffuse adversary. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 420–425, 1998.

N. E. Young. On-line paging against adversarially biased random inputs. *Journal of Algorithms*, 37(1):218–235, 2000.