


Distributional Online Weighted Paging with Limited Horizon

Yaron Fairstein  

Amazon.com, Haifa, Israel¹

Joseph (Seffi) Naor 

Computer Science Department, Technion, Haifa, Israel

Tomer Tsachor  

Computer Science Department, Technion, Haifa, Israel

Abstract

In this work we study the classic problem of online weighted paging with a probabilistic prediction model, in which we are given additional information about the input in the form of distributions over page requests, known as *distributional online paging* (DOP). This work continues a recent line of research on learning-augmented algorithms that incorporates machine-learning predictions in online algorithms, so as to go beyond traditional worst-case competitive analysis, thus circumventing known lower bounds for online paging. We first provide an efficient online algorithm that achieves a constant factor competitive ratio with respect to the best online algorithm (policy) for weighted DOP that follows from earlier work on the stochastic k -server problem.

Our main contribution concerns the question of whether distributional information over a limited horizon suffices for obtaining a constant competitive factor. To this end, we define in a natural way a new predictive model with limited horizon, which we call *Per-Request Stochastic Prediction* (PRSP). We show that we can obtain a constant factor competitive algorithm with respect to the optimal online algorithm for this model.

2012 ACM Subject Classification Theory of computation → Caching and paging algorithms; Theory of computation → Probabilistic computation; Theory of computation → Linear programming

Keywords and phrases Online algorithms, Caching, Stochastic analysis, Predictions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.15

Category APPROX

Funding *Joseph (Seffi) Naor*: Supported in part by Israel Science Foundation grant 2233/19 and United States – Israel Binational Science Foundation (BSF) grant 2033185.

1 Introduction

In the weighted paging problem there is a universe of n pages, where each page has a weight², and there is a cache that can hold up to k pages. At each time step a page is requested, and if the requested page is already in the cache then no cost is incurred, otherwise, the page must be loaded into the cache, incurring a cost equal to its weight. The goal is to minimize the total cost incurred.

Paging is one of the earliest and most extensively studied problems in online computation and competitive analysis [38, 22, 40, 42, 36, 7, 6, 1, 4, 11, 10, 27, 28, 29], including works on non-standard caching models, e.g., elastic caches [25], caching with time windows [26], caching with dynamic weights [21], and caching with machine learning predictions [35]. In fact, online paging has become a focal point for many of the recent developments in competitive analysis, e.g., the online primal-dual method, projections, and mirror descent [17, 16, 15].

¹ Work done while at the Technion before joining Amazon.

² In the unweighted version of the problem, all weights are equal (unit weights).



In their seminal paper, Sleator and Tarjan [38] showed that any deterministic online algorithm is at least k -competitive and that the LRU policy (Least Recently Used) is exactly k -competitive for unweighted paging. The k -competitive bound was later generalized to weighted paging as well [19, 41]. When randomization is allowed, Fiat et al. [22] gave the elegant randomized marking algorithm for unweighted paging, which is $\Theta(\log k)$ -competitive against an oblivious adversary. Bansal et al. [7] gave a $\Theta(\log k)$ -competitive randomized algorithm for weighted paging based on the online primal-dual framework [17].

Algorithms with predictions, or learning-augmented algorithms, is an emerging field of research lying at the intersection of machine learning and foundations of algorithms (e.g. [5]). The goal is to use machine-generated predictions, that can be either deterministic or probabilistic, so as to go beyond traditional worst-case competitive analysis, and relax the overly pessimistic assumption of not having any prior knowledge of the future. This is in line with recent momentum in deploying machine learning techniques for various applications, e.g., search, business processes, and health.

The study of online paging with predictions has been a catalyst for the development of this new field. In an influential paper, Lykouris et al. [35] studied a simple predictor that provides for each requested page the next time step in which it is requested again, called PRP (Per-Request Prediction). Clearly, for unweighted paging, PRP suffices for implementing Belady's algorithm. Lykouris et al. [35] analyzed the robustness of PRP.

Computationally, weighted paging is a very different problem from unweighted paging, since it requires more global information about the request sequence to obtain (near) optimal algorithms. For example, Belady's local rule suffices to define an optimal offline algorithm in the unweighted case, while a minimum cost flow procedure is needed for computing an optimal solution in the weighted case. This is also manifested in the online setting, where PRP does not improve on the competitive factor in the weighted case [31]. Even with precise PRP, any deterministic online algorithm remains $\Omega(k)$ -competitive, and any randomized algorithm is $\Omega(\log k)$ -competitive.

Distributional Online Paging

We focus on probabilistic prediction models for online weighted paging. Suppose that an online algorithm is given *in advance*, for each time step $t \in \{1, 2, \dots, T\}$, a distribution over page requests at t . Thus, the request at time t is drawn according to D_t . The given distributions are assumed to be *independent* between different time steps and the distributions are not necessarily identical. This model is known in the literature as *distributional online paging*, or DOP [34], and it can be viewed as the probabilistic counterpart of PRP. DOP is also a special case of the stochastic uber problem studied by [20]. (See more about this problem in the sequel.)

Define the cost of an online algorithm (or policy) for DOP to be the expected cost taken over all possible request sequences, where the probability of a request sequence is determined by the distributions D_1, \dots, D_T . An optimal online algorithm minimizes the expected cost and is defined by a Markov Decision Process. It can be computed by either a dynamic program or a linear program. Unfortunately, the state space of the dynamic program, or the size of the linear program, is of exponential size in k , rendering it computationally impractical. The computational hardness of finding an optimal algorithm for DOP is still open to the best of our knowledge³.

³ It is stated as an open problem in [13]. For general metrics (i.e., the k -server problem), [20] gives a simple reduction to prove hardness: the uniform distribution over the points in the metric is given at all times; thus, a solution to the k -median problem on the metric defines the placement of the servers in an optimal online algorithm.

For unweighted DOP, the work of Lund et al. [34] is seminal. They show that *full* information about the distributions in each time step is actually not needed in order to get a near-optimal online algorithm. Specifically, for general distributions over page requests, if the probability that p is requested before q is available for any pair of pages p and q , then this information can be leveraged to get an efficient and simple 4-competitive algorithm with respect to the best online algorithm. However, these ideas do not seem to generalize to the weighted setting, due to the more global nature of weighted paging, as indicated before.

1.1 Our Results

We study the weighted DOP problem. Our goal is to provide an *efficient* online algorithm that achieves a constant competitive factor with respect to the best online algorithm (policy) for weighted DOP. Our starting point is a linear program for DOP which is based on the work of [20] for the stochastic k -server problem. In the case of the paging problem, this linear program specifies to which cache slot a page is loaded. The linear program provides a lower bound on the cost of any non-adaptive algorithm for DOP. However, [20] show that an optimal non-adaptive algorithm can cost at most thrice the cost of an optimal online algorithm for DOP. In Section 3 we provide the details for rounding the k -server linear program, yielding a constant competitive algorithm for the weighted DOP problem.⁴ This is summarized in the following theorem.

► **Theorem 1.** *There exists an efficient algorithm for weighted DOP with $O(1)$ -competitive ratio.*

The constant competitive factor obtained in Theorem 1 strongly utilizes information about the page distributions over the *entire* time horizon. However, such distributional information may not always be available. Thus, a natural question is whether distributional information over a limited horizon suffices for obtaining a constant competitive factor. This is the main focus of our paper.

In [31], a new deterministic predictive model for online weighted paging is suggested, due to the weakness of PRP in the weighted setting, as indicated earlier. This model, called SPRP, assumes that when a page p is requested, the full request sequence up to the next request for p is revealed. It turns out that the SPRP predictive model is strong enough to obtain a 2-competitive algorithm for online weighted paging in the adversarial setting [31].

Our first contribution is a novel limited horizon distributional model which we call the *Per-Request Stochastic Prediction* (PRSP) model. In this model, at any point of time, the known horizon of (future) distributions guarantees that for each page p in the cache, the sum of the probabilities of requesting p (in the known horizon) is at least one. Interestingly, this model captures the property needed from a limited horizon distributional model in order to design a near-optimal online algorithm.

Note that in a deterministic setting, D_t is equal to zero for all pages, except for one page, for which it is equal to one. Thus, when PRSP is restricted to a deterministic setting, it is equivalent to the SPRP model of [31].

We show that any algorithm for (full horizon) weighted DOP can be used in a black-box manner in the PRSP model, while increasing the competitive factor only by a constant. We thus obtain the next theorem.

⁴ It is interesting to note that a natural linear programming formulation for the weighted DOP problem that provides a lower bound on the best online algorithm has a large integrality gap which depends on the maximum page weight. Thus, the work of [20] manages to circumvent this gap by utilizing a stronger LP.

► **Theorem 2.** *If there exists an α -competitive algorithm for weighted DOP, then there exists an $O(\alpha)$ -competitive algorithm for weighted DOP under the PRSP model. I.e., there exists an efficient $O(1)$ -competitive algorithm for weighted DOP under the PRSP model.*

To prove Theorem 2, we design an algorithm called Split-and-Solve. The algorithm is very natural and it splits the time horizon into phases, solving each one separately. Using the properties of the PRSP model, the phases are chosen such that at the beginning of the phase the distributions are known for all times in the phase. Therefore, each phase can be regarded as a weighted DOP instance. As the analysis of Split-and-Solve does not make any assumptions regarding the solutions of the phases, any algorithm for weighted DOP can be employed in a black-box manner (e.g., Theorem 1). However, as each phase commences, the cache is reset to be the final cache state of the optimal offline solution of the realization of the request sequence of the previous phase.

The difficulty with analyzing the Split-and-Solve algorithm is in stitching together the performance of the different phases. Even though the policy employed in each phase by itself may be optimal, note that the initial cache states (in each phase) may be very different from the corresponding cache states of the optimal online algorithm (which is familiar with the full horizon). In particular, since our caching problem is weighted, the gap (in terms of weight) between cache states can be arbitrarily large, and may also further lead to poor performance within the phase. However, the crucial ingredient for bounding the performance of each phase is the property of the PRSP model that guarantees that for each page in the cache the sum of the probabilities in the known horizon adds up to at least 1. Thus, the performance of each phase can be related to the performance of the optimal online algorithm in the phase with a multiplicative constant factor (together with an additive term). Using a global analysis that considers all phases, the loss incurred by the sum of the additive terms can be charged to the cost of the optimal online algorithm, yielding Theorem 2.

1.2 Related Work

The k -server problem generalizes the paging problem to arbitrary metric spaces. (In paging the underlying metric is a weighted star.) A natural generalization of DOP is distributional k -server, where in every time step there is a given distribution over the possible request point. This problem was studied by [20], who also introduced the stochastic Uber problem, where each request is defined by two points in the metric. The server satisfying a request must travel to the start point of the request and then to its end point, incurring a cost equal to the total distance traveled. [20] gave a constant competitive factor algorithm for the case where the metric is a line. For general metrics, they gave an $O(\log n)$ -competitive algorithm, where n is the number of points in the metric.

Work on distributional paging goes back more than fifty years. Franaszek and Wagner [24] compared FIFO and LRU in a model where every request is drawn from a fixed probability distribution over time. Shedler and Tung [37] suggested a Markov model for generating requests. This model and its extensions were analyzed in [32]. They also showed a gap of $\Omega(\log k)$ between optimal online and offline algorithms in the case of a uniform distribution.

Besides stochastic models, several paging models assuming partial knowledge of future requests have been studied. For example, [8] studied the PRP model of Lykouris et al. [35] (mentioned earlier) in the weighted setting. A very sophisticated algorithm is given in [8] for this model whose competitive factor is at most a logarithm of the number of weight classes. Other examples for models with predictions are paging with locality of reference [12, 23, 30], paging with lookahead [3, 14, 39] and interleaved paging [9, 18, 33].

2 Preliminaries

2.1 Distributional Online Paging

In the weighted paging problem there is a universe of n pages, denoted by $P = \{p_1, p_2, \dots, p_n\}$, and a cache of size k . The initial cache state is $C_0 \subseteq P$. Each page p is associated with a weight w_p , the cost of loading or evicting page p to the cache. For a sequence of requests $\sigma_1, \sigma_2, \dots, \sigma_T$, an algorithm \mathcal{A} determines a series of cache states C_1, \dots, C_T , such that $\forall t : \sigma_t \in C_t$. The cost of serving the request sequence by \mathcal{A} is $\sum_{t=1}^T \sum_{p \in C_t \Delta C_{t-1}} w_p$, which is equal to the sum of the loading costs and the eviction costs. Note that the sum of the eviction costs and the loading costs can differ by at most an additive constant depending only on the initial and final cache contents. For simplicity, we assume the initial cache and final cache are identical, i.e., $C_0 = C_T$. This means that the loading costs are equal to the eviction costs.

We focus on *distributional* prediction models in this paper. Suppose that we are given *in advance* at time $t = 0$, for each (future) time step $t \in \{1, 2, \dots, T\}$, a distribution over page requests at t . The given distributions are not necessarily identical, yet assumed to be *independent* between different time steps. More formally, for every $t \in [T]$, a probability distribution D_t is given from which request σ_t is drawn at time t . This model is called *distributional online paging*, or DOP.

► **Definition 3.** For an algorithm \mathcal{A} , let $E(\mathcal{A})$ denote the expected cost of \mathcal{A} over all realizations of input sequences generated according to distributions D_1, \dots, D_T .

Denote by O the best online algorithm (in expectation) for DOP, i.e., it minimizes $E(O)$. We emphasize that algorithm O is only familiar with the distributions D_1, \dots, D_t , but does not have prior knowledge of the actual realization of the requests. Let $E(O) = OPT$.

2.2 Limited Horizon

So far we have assumed that all distributions D_1, \dots, D_T are given as input at the beginning. This is, of course, unreasonable in many cases as T may be very large. We aspire to bound the horizon that an online algorithm needs to “see”, i.e., at any time t , how many distributions into the future are available to the algorithm.

In [31] it was shown that the per-request-prediction (PRP) model and the lookahead model are not sufficient to circumvent the known lower bounds for online paging. In [31] a constant competitive factor for deterministic online weighted paging is achieved only through a combination of two models which they call SPRP. Specifically, upon arrival of a page, the time of its next request t is given, as in PRP, as well as the full sequence of page requests up to t .

Here, we propose an extension of the above model for our stochastic setting called Per-Request Stochastic Prediction (PRSP). The PRSP model requires that at any time t a sequence of future distributions is revealed such that sufficient information is revealed for each page in the current cache, as follows.

► **Definition 4.** Given a cache $C \subseteq P$ and time t , let $N(C, t)$ be the earliest time t' that satisfies:

$$\forall p \in C : \sum_{\tau=t}^{t'} D_\tau(p) \geq 1.$$

Note that in the deterministic setting, D_t is equal to zero for all pages, except for one page, for which it is equal to one. Assuming all pages in C are there because they were previously requested, Definition 4, restricted to a deterministic setting, is equivalent to the SPRP model of [31].

Using the definition of $N(C, t)$ we can now give a formal definition of the PRSP model.

► **Definition 5.** *In the PRSP model at each time step t , where the cache is C_t , the sequence of distributions $(D_t, \dots, D_{N(C_t, t)})$ is revealed to the algorithm.*

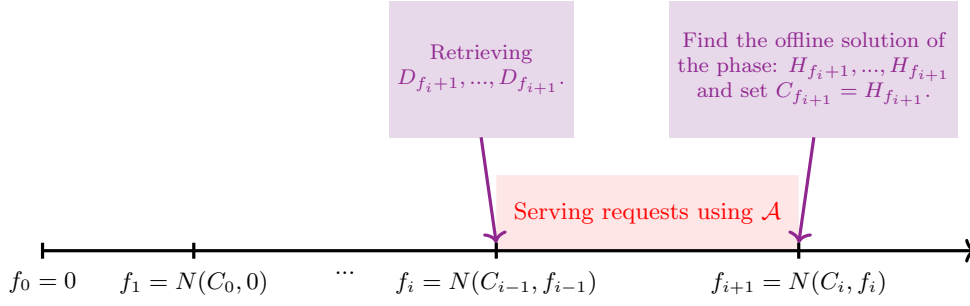
In a deterministic setting, at each time t it holds that there exists a single page p such that $D_t(p) = 1$ and it is equal to zero for all other pages. Thus, in the PRSP model, when restricted to a deterministic setting, at any time t , $N(C_t, t)$ is equal to the latest time over the next arrivals of all pages in C_t . In reality, the distributions up to time $N(C_t, t)$ were revealed earlier when the pages in C_t were requested and they were loaded to the cache. Thus, in the deterministic setting, we can interpret PRSP as SPRP since it reveals upon a request to a page the whole sequence of pages up to its next request.

3 Full Horizon

In this section we consider the weighted DOP problem when all the distributions are given in advance and provide a proof for Theorem 1. The proof essentially follows from the work of [20] on the stochastic k -server problem. Recall that weighted paging is the special case of k -server when the underlying metric is a weighted star. We show here that when applying the linear program for stochastic k -server to the special case of DOP, it can be rounded yielding a constant competitive algorithm. It is interesting to note that a natural linear programming formulation for the weighted DOP problem has a large gap compared to the best online algorithm, where the gap depends on the maximum page weight. Thus, the work of [20] manages to circumvent this gap by utilizing a stronger LP.

In [20], an algorithm \mathcal{A} for DOP is defined to be *non-adaptive* if it satisfies the following. First, algorithm \mathcal{A} pre-computes a sequence of cache configurations C_1, \dots, C_T ; then it serves the request sequence as follows. Upon arrival of request σ_t at time t : (i) \mathcal{A} changes the cache contents to configuration C_t ; (ii) \mathcal{A} replaces the lightest page in C_t with σ_t ; (iii) \mathcal{A} changes the cache contents back to the configuration that preceded the arrival of σ_t . The linear program for stochastic k -server suggested in [20] provides a lower bound on the cost of any non-adaptive algorithm. However, an online algorithm for the stochastic k -server problem does not necessarily imply a feasible solution for the latter linear program. It is shown in [20, Theorem 1.3] that an optimal non-adaptive online algorithm is a 3-approximation with respect to an optimal online algorithm for stochastic k -server. Thus, non-adaptive algorithms provide a useful tool for obtaining a competitive algorithm for stochastic k -server.

In what follows we describe the specialization of the linear program for the stochastic k -server problem [20] to DOP. Essentially, this means specifying to which cache slot is a page loaded. Thus, for each page p , there is a variable $b_{t,p}$ that indicates the fraction of p that is not in the cache. For each page p and possible request q , variable $x_{t,p,q}$ indicates whether q is served by replacing p .



■ **Figure 1** Algorithm *Split-and-Solve* splits the timeline into phases and separately solves each phase as DOP. As phase i begins at f_i the distributions till $N(C_{f_i}, f_i)$ are revealed. The DOP black box \mathcal{A} serves the requests during the phase. As the phase terminates at $N(C_{f_i}, f_i)$, the optimal offline solution $H_{f_{i+1}}, \dots, H_{f_{i+1}}$ is computed with the realization of the phase and $H_{f_{i+1}}$ is loaded into $C_{f_{i+1}}$.

$$\begin{aligned}
\min \quad & \sum_{p,q,t \geq 1} (w_p + w_q) \cdot D_t(q) \cdot x_{t,p,q} + \sum_{p,t \geq 1} w_p \cdot |b_{t,p} - b_{t-1,p}| \quad \text{s.t.} \\
\forall t, q : \quad & \sum_{p \neq q} x_{t,p,q} \geq b_{t,q} \\
\forall t, p, q : \quad & x_{t,p,q} \leq 1 - b_{t,p} \\
\forall t : \quad & \sum_p b_{t,p} \geq n - k
\end{aligned} \tag{31}$$

The cost function accounts for the weight of the pages that make two switches when serving a request. By doing so it forces the solution to be non-adaptive. Note that (31) gives us a lower bound only on the optimal non-adaptive algorithm. Thus, not every paging algorithm can be mapped into a solution for (31), only a non-adaptive one.

The details for rounding the LP solution so as to yield a 60-competitive algorithm are given in Appendix A.

4 The Split-and-Solve Algorithm

Section 3 provides us with a constant competitive factor algorithm, but requires that at time 0 all distributions D_1, \dots, D_T are known. In this section we present the Split-and-Solve algorithm that provides a constant-competitive factor for DOP in the PRSP model. The algorithm is very natural. It first splits the time horizon into phases: if a phase begins at time t , then it ends at time $N(C_t, t)$. As the horizon of the algorithm is at most $\max\{N(C_{t'}, t') \mid t' \leq t\}$ for every time step t , the distributions $D_{t+1}, \dots, D_{N(C_t, t)}$ can be retrieved by the properties of the PRSP model.

Each phase is solved independently using a (full horizon) DOP algorithm in a black-box manner. However, as each phase commences, the cache is reset to be the final cache state of the optimal offline solution of the realization of the request sequence of the previous phase. This cache state can be computed by running a min cost flow algorithm. This guarantees that the definition of phases is independent of the algorithm \mathcal{A} . The steps of the algorithm are depicted in Figure 1. We are now ready to give a formal definition of the Split-and-Solve algorithm.

■ Algorithm 1 Split-and-Solve (SaS).

Input: Instance \mathbf{I} of DOP and Algorithm \mathcal{A} for DOP.

- 1: Initialize $i = 0, f_0 = 0, H_0 = C_0$.
 - 2: **while** $f_i < T$ **do**
 - 3: Let $f_{i+1} = N(C_{f_i}, f_i)$.
 - 4: Retrieve $D_{f_{i+1}}, \dots, D_{f_{i+1}}$.
 - 5: Set $C_{f_{i+1}}, \dots, C_{f_{i+1}}$ as the solution returned by \mathcal{A} for serving requests in time range $[f_i, f_{i+1}]$ with initial cache H_{f_i} .
 - 6: Let $H_{f_{i+1}}, \dots, H_{f_{i+1}}$ be the optimal offline solution for time range $[f_i, f_{i+1}]$ with initial cache C_{f_i} .
 - 7: After serving $\sigma_{f_{i+1}}$, set $C_{f_{i+1}} = H_{f_{i+1}}$ as the initial cache of the next phase.
 - 8: Update $i := i + 1$.
-

Before analyzing the algorithm, we need the following notation. We denote by $F + 1$ the number of phases into which the time horizon is split. The optimal online algorithm is denoted by O_{on} and its caches are denoted by $\{O_t\}_{t \in [T]}$.

In the sequel we will show that given an α -competitive algorithm for DOP, the Split-and-Solve (SaS) algorithm has an $O(\alpha)$ competitive ratio, losing only an additional constant factor. To do so, we first bound in Lemma 7 the cost of our algorithm by the cost of the online algorithm when we also reset its cache at the beginning of each phase (i.e., similarly to Step 6, where at the beginning of each phase the cache is set to H_{f_i}). Then, Lemma 8 bounds the cost of the online algorithm (with the cache reset) at phase i by the sum of three components:

1. The expected cost of O_{on} during the phase.
2. The total weight of $H_{f_i} \setminus O_{f_i}$.
3. The sum over pages in $O_{f_i} \setminus H_{f_i}$ of the page weight times the probability it is requested in phase i .

Lemmas 9, 11 and 12 bound the last two components by a constant factor of the cost of O_{on} .

In the following definition we provide a notation for a partial solution.

► **Definition 6.** Let $t_1, t_2 \in [T]$ such that $t_1 < t_2$ and $C \subseteq P$. We denote the expected cost of an algorithm \mathcal{A} on the sub-range $[t_1, t_2]$ with $C_{t_1} = C$ as its initial cache by $\mathcal{A}(C, t_1, t_2)$.

From the definition of the optimal offline and online algorithms it is easy to see that for any $t_1 < t_2$ and cache C it holds that $O_{off}(C, t_1, t_2) \leq O_{on}(C, t_1, t_2)$. The following lemma bounds the expected cost of phase i by $2\alpha + 1$ times the expected cost of O during the phase when O begins the phase with the same cache.

► **Lemma 7.** Given an α -competitive algorithm \mathcal{A} for DOP it holds that for every phase i :

$$\text{SaS}(H_{f_i}, f_i, f_{i+1}) \leq (2\alpha + 1) \cdot O_{on}(H_{f_i}, f_i, f_{i+1}).$$

Proof. At the beginning of each phase i we set the initial cache passed to algorithm \mathcal{A} in Step 5 to be the cache of the optimal offline solution H_{f_i} . For simplicity, we associate this cost with phase $i - 1$ (note that for $i = 0$ the cost is zero as $H_0 = C_0$). Thus, in phase i we need to bound the cost of serving requests as well as the cost of loading cache $H_{f_{i+1}}$ at the end of the phase.

The expected cost of serving requests in phase i is at most $\alpha \cdot O_{on}(H_{f_i}, f_i, f_{i+1})$ due to the competitive ratio of \mathcal{A} . Next, loading $H_{f_{i+1}}$ can be bounded by the cost of loading H_{f_i} and only then loading $H_{f_{i+1}}$. Loading H_{f_i} must cost less than the cost of SaS at this phase (as eviction and loading costs are symmetric). Afterwards, loading $H_{f_{i+1}}$ is bounded by the cost of the optimal offline algorithm, but

$$O_{off}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(H_{f_i}, f_i, f_{i+1}).$$

Summarizing over the three cost components produces the desired bound. \blacktriangleleft

The following lemma bounds $O_{on}(H_{f_i}, f_i, f_{i+1})$ with the total cost of evicting $H_{f_i} \setminus O_{f_i}$, loading the requested pages from $O_{f_i} \setminus H_{f_i}$ and then serving the requests as *on*.

► **Lemma 8.** *Let $\gamma_{i,p}$ be the event that page p is requested in phase i , i.e., $\gamma_{i,p} = \mathbb{1}_p$ is requested in $[f_i, f_{i+1}]$. Then, it holds that:*

$$O_{on}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(O_{f_i}, f_i, f_{i+1}) + \sum_{p \in H_{f_i} \setminus O_{f_i}} w_p + \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right].$$

Proof. Consider the following online algorithm for serving phase i . First, evict the pages in $H_{f_i} \setminus O_{f_i}$, incurring a cost of $\sum_{p \in H_{f_i} \setminus O_{f_i}} w_p$. Next, run the optimal online algorithm. A feasible solution for the online algorithm would be to act as if $O_{f_i} \setminus H_{f_i}$ are in the cache, incurring a cost of $O_{on}(O_{f_i}, f_i, f_{i+1})$. Nonetheless, it might be that a page $p \in O_{f_i} \setminus H_{f_i}$ is requested, incurring an additional cost of w_p , though this only happens with probability $\mathbb{E}[\gamma_{i,p}]$. Summing over the three terms produces the desired bound. \blacktriangleleft

After bounding the cost at each phase, we will now evaluate the cost of all phases, bounding the cost of splitting the time horizon. To do so we must bound $\sum_i \sum_{p \in H_{f_i} \setminus O_{f_i}} w_p$ and $\sum_i \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right]$. The following lemma bounds the former term. It strongly uses the property of the PRSP model that for every page in cache the sum of the probabilities of requesting this page in the known horizon is at least 1.

► **Lemma 9.** *In each phase $i \in [F]$ it holds that $\sum_{p \in H_{f_i} \setminus O_{f_i}} w_p \leq \frac{e}{e-1} \cdot O_{on}(O_{f_i}, f_i, f_{i+1})$.*

Proof. At the beginning of each phase we set $f_{i+1} = N(H_{f_i}, f_i)$. Thus from Definition 4 it holds that $\sum_{t=f_i}^{f_{i+1}-1} D_t(p) \geq 1$ for each page $p \in H_{f_i}$. Using the inequality $1 - x \leq e^{-x}$ for $x \in [0, 1]$ we get that for $p \in H_{f_i}$:

$$\mathbb{E}[\gamma_{i,p}] = 1 - \prod_{t=f_i}^{f_{i+1}-1} (1 - D_t(p)) \geq 1 - \prod_{t=f_i}^{f_{i+1}-1} e^{-D_t(p)} = 1 - e^{-\sum_{t=f_i}^{f_{i+1}-1} D_t(p)} \geq 1 - e^{-1}. \quad (42)$$

Each page $p \in H_{f_i} \setminus O_{f_i}$ must be loaded by O_{on} at phase i if it is requested. Thus from Equation (42) it follows that,

$$O_{on}(O_{f_i}, f_i, f_{i+1}) \geq \sum_{p \in H_{f_i} \setminus O_{f_i}} \mathbb{E}[\gamma_{i,p}] \cdot w_p \geq \sum_{p \in H_{f_i} \setminus O_{f_i}} (1 - e^{-1}) \cdot w_p.$$

Dividing by $1 - e^{-1}$ completes the proof of the lemma. \blacktriangleleft

The next corollary follows from lemmas 8 and 9.

► **Corollary 10.**

$$O_{on}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right].$$

15:10 Distributional Online Weighted Paging with Limited Horizon

Next, we bound $\sum_i \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right]$. To do so we need the following auxiliary lemma which bounds the expected eviction costs of O_{off} .

► **Lemma 11.** *Let $Ev_{off}(H_{f_i}, f_i, f_{i+1})$ be the expected eviction costs of the optimal offline algorithm at phase i when initialized with cache H_{f_i} . It holds that,*

$$Ev_{off}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}).$$

Proof. Due to the optimality of the offline algorithm, $O_{off}(H_{f_i}, f_i, f_{i+1}) \leq O_{on}(H_{f_i}, f_i, f_{i+1})$. So from Corollary 10 it follows that,

$$O_{off}(H_{f_i}, f_i, f_{i+1}) \leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]. \quad (43)$$

In addition, the offline optimal algorithm must load any page $p \notin H_{f_i}$ in phase i if it is requested. Thus, the expected loading costs of the offline optimal algorithm are at least $\mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]$. By combining the bound on the loading costs and Equation (43) we get that,

$$\begin{aligned} Ev_{off}(H_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right] &\leq O_{off}(H_{f_i}, f_i, f_{i+1}) \\ &\leq \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}) + \mathbb{E} \left[\sum_{p \in O_{f_i} \setminus H_{f_i}} W_p \cdot \gamma_{i,p} \right]. \end{aligned}$$

Subtracting the last term from both sides proves the lemma. ◀

► **Lemma 12.**

$$\mathbb{E} \left[\sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}).$$

Proof. In Section 2.1 it is stated that we assume $C_0 = C_T$. If this was not the case, we can simply load C_T with only a constant additional cost. From this assumption it follows that the total eviction costs are equal to the total expected loading costs.

At each phase i , O_{off} must load pages in $O_{f_i} \setminus H_{f_i}$ if they are requested. Thus, in the event $\gamma_{i,p}$ it will incur a loading cost of w_p . As the total expected eviction costs are equal to total expected loading costs we get that,

$$\mathbb{E} \left[\sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} Ev_{off}(H_{f_i}, f_i, f_{i+1}).$$

Due to Lemma 11 it holds that

$$\mathbb{E} \left[\sum_{i \in [F]} \sum_{p \in O_{f_i} \setminus H_{f_i}} w_p \cdot \gamma_{i,p} \right] \leq \sum_{i \in [F]} \frac{2e-1}{e-1} O_{on}(O_{f_i}, f_i, f_{i+1}). \quad \blacktriangleleft$$

The following lemma combines the above results to produce a bound on the competitive ratio of SaS.

► **Lemma 13.** *Given an α -competitive algorithm \mathcal{A} for DOP, the Split-and-Solve algorithm is $(2\alpha + 1) \cdot \frac{4e-2}{e-1}$ -competitive.*

Proof. From Lemma 7 the cost of the Split-and-Solve algorithm is at most

$$\sum_{i \in [F]} (2\alpha + 1) \cdot O_{on}(H_{f_i}, f_i, f_{i+1}).$$

Combining with Corollary 10 and Lemma 12 which provides a bound on $\sum_{i \in [F]} O_{on}(H_{f_i}, f_i, f_{i+1})$, we can bound the competitive ratio of the Split-and-Save algorithm by $(2\alpha + 1) \cdot \frac{4e-2}{e-1}$. ◀

Theorem 2 follows immediately from Lemma 13.

References

- 1 Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000. doi:10.1016/S0304-3975(98)00116-9.
- 2 Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An $o(\log k)$ -competitive algorithm for generalized caching. *ACM Trans. Algorithms*, 15(1), November 2018. doi:10.1145/3280826.
- 3 Susanne Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997. doi:10.1007/PL00009158.
- 4 Susanne Albers, Sanjeev Arora, and Sanjeev Khanna. Page replacement for general caching problems. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 31–40. ACM/SIAM, 1999. URL: <http://dl.acm.org/citation.cfm?id=314500.314528>.
- 5 Algorithms with predictions (ALPS). <https://algorithms-with-predictions.github.io/>. Accessed: 2022-11-07.
- 6 Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012. doi:10.1137/090779000.
- 7 Nikhil Bansal, Niv Buchbinder, and Joseph Seffi Naor. A primal-dual randomized algorithm for weighted paging. *Journal of the ACM (JACM)*, 59(4):19, 2012.
- 8 Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Scale-free allocation, amortized convexity, and myopic weighted paging. *CoRR*, abs/2011.09076, 2020. arXiv:2011.09076.
- 9 Rakesh D. Barve, Edward F. Grove, and Jeffrey Scott Vitter. Application-controlled paging for a shared cache. *SIAM Journal on Computing*, 29(4):1290–1303, 2000. doi:10.1137/S0097539797324278.
- 10 A. Blum, M. Furst, and A. Tomkins. What to do with your free time: algorithms for infrequent requests and randomized weighted caching, 1996.
- 11 Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 450–458. IEEE Computer Society, 1999. doi:10.1109/SFFCS.1999.814617.
- 12 A. Borodin, S. Irani, P. Raghavan, and B. Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, 1995. doi:10.1006/jcss.1995.1021.
- 13 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
- 14 Dany Breslauer. On competitive on-line paging with lookahead. *Theor. Comput. Sci.*, 209(1-2):365–375, 1998. doi:10.1016/S0304-3975(98)00118-2.

- 15 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 3–16. ACM, 2018. doi:10.1145/3188745.3188798.
- 16 Niv Buchbinder, Shahar Chen, and Joseph Naor. Competitive analysis via regularization. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 436–444. SIAM, 2014. doi:10.1137/1.9781611973402.32.
- 17 Niv Buchbinder, Joseph Seffi Naor, et al. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.
- 18 Pei Cao, Edward W. Felten, and Kai Li. Application-Controlled file caching policies. In *USENIX Summer 1994 Technical Conference (USENIX Summer 1994 Technical Conference)*, Boston, MA, June 1994. USENIX Association. URL: <https://www.usenix.org/conference/usenix-summer-1994-technical-conference/application-controlled-file-caching-policies>.
- 19 Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, 1991. doi:10.1137/0220008.
- 20 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-server: How should uber work? In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPICs*, pages 126:1–126:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.ICALP.2017.126.
- 21 Guy Even, Moti Medina, and Dror Rawitz. Online generalized caching with varying weights and costs. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 205–212. ACM, 2018. doi:10.1145/3210377.3210404.
- 22 Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991. doi:10.1016/0196-6774(91)90041-V.
- 23 Amos Fiat and Manor Mendel. Truly online paging with locality of reference. *CoRR*, abs/cs/0601127, 2006. arXiv:cs/0601127.
- 24 Peter A. Franaszek and T. J. Wagner. Some distribution-free aspects of paging algorithm performance. *J. ACM*, 21(1):31–39, 1974.
- 25 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Elastic caching. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 143–156. SIAM, 2019. doi:10.1137/1.9781611975482.10.
- 26 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Caching with time windows. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1125–1138. ACM, 2020. doi:10.1145/3357713.3384277.
- 27 Sandy Irani. Competitive analysis of paging. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art (the book grow out of a Dagstuhl Seminar, June 1996)*, volume 1442 of *Lecture Notes in Computer Science*, pages 52–73. Springer, 1996. doi:10.1007/BFb0029564.
- 28 Sandy Irani. Page replacement with multi-size pages and applications to web caching. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 701–710. ACM, 1997. doi:10.1145/258533.258666.

- 29 Sandy Irani. Randomized weighted caching with two page weights. *Algorithmica*, 32(4):624–640, 2002. doi:10.1007/s00453-001-0095-6.
- 30 Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996. doi:10.1137/S0097539792236353.
- 31 Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPICs*, pages 69:1–69:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ICALP.2020.69.
- 32 Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 208–217. IEEE Computer Society, 1992. doi:10.1109/SFCS.1992.267771.
- 33 Ravi Kumar, Manish Purohit, Zoya Svitkina, and Erik Vee. Interleaved caching with access graphs. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '20*, pages 1846–1858, USA, 2020. Society for Industrial and Applied Mathematics.
- 34 Carsten Lund, Steven J. Phillips, and Nick Reingold. Paging against a distribution and IP networking. *J. Comput. Syst. Sci.*, 58(1):222–232, 1999. doi:10.1006/jcss.1997.1498.
- 35 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/lykouris18a.html>.
- 36 Lyle A. McGeoch and Daniel Dominic Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6):816–825, 1991. doi:10.1007/BF01759073.
- 37 Gerald S. Shedler and C. Tung. Locality in page reference strings. *SIAM J. Comput.*, 1(3):218–241, 1972.
- 38 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.
- 39 Neal Young. *Competitive paging and dual-guided on-line weighted caching and matching algorithms*. Princeton University, 1991.
- 40 Neal E. Young. On-line caching as cache size varies. In Alok Aggarwal, editor, *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California, USA*, pages 241–250. ACM/SIAM, 1991. URL: <http://dl.acm.org/citation.cfm?id=127787.127832>.
- 41 Neal E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994. doi:10.1007/BF01189992.
- 42 Neal E. Young. On-line file caching. In *Symposium on Discrete algorithms*, pages 82–86, 1998.

A Rounding the LP

We show how to discretize to multiples of $1/k$ a solution to the LP. We use techniques from [2] and transform each $x_{t,p}$ to a multiple of $\frac{1}{8k}$. First, We set $b'_{t,p} := \min\{1, 2 \cdot b_{t,p}\}$. Assume $a_{t,p} \in [8k]$ denote $\lceil \frac{8k \cdot b'_{t,p}}{8k} \rceil = \frac{a_{t,p}}{8k}$. For every $p \in P$, we do the following iterative process for $t = 1 \dots T$:

1. If $a_{t,p}$ is even, set $y_{t,p} := \frac{a_{t,p}}{8k}$.
2. Else, if $y_{t-1,p} > \frac{a_{t,p}}{8k}$, set $y_{t,p} := \frac{a_{t,p}+1}{8k}$.
3. Otherwise, set $y_{t,p} := \frac{a_{t,p}-1}{8k}$.

► **Lemma 14.** *The following statements hold:*

15:14 Distributional Online Weighted Paging with Limited Horizon

1. $\forall t, p : \text{if } b_{t,p} = 0 \text{ then } y_{t,p} = 0.$
2. $\forall t, p : \text{if } b_{t,p} = 1 \text{ then } y_{t,p} = 1.$
3. $\forall t, p : y_{t,p} \leq 4 \cdot b_{t,p}.$
4. $\forall p : \sum_{t \in [1, T]} |y_{t,p} - y_{t-1,p}| \leq 4 \cdot \sum_{t \in [1, T]} |b_{t,p} - b_{t-1,p}|.$
5. $\forall t : \sum_{p \in P} y_{t,p} \geq n - k.$

Proof.

1. $b_{t,p} = 0 \rightarrow a_{t,p} = 0 \rightarrow y_{t,p} = 0.$
2. $b_{t,p} = 1 \rightarrow a_{t,p} = 8k \rightarrow y_{t,p} = 1.$
3. If $b_{t,p} < \frac{1}{16k}$, then $a_{t,p} = 0$ implying $y_{t,p} = 0$. Otherwise, $y_{t,p} \leq 2b_{t,p} + \frac{1}{8k} \leq 4b_{t,p}.$
4. For a page $p \in P$, we prove the claim by induction on T .

Basis: for $t = 0$ the claim holds trivially.

Inductive step: Assume the claim holds for every $\tau' < t$. We assume w.l.o.g that $a_{t,p} \leq a_{t-1,p}$. We split to cases:

- a. If $a_{t,p} = a_{t-1,p}$ then also $y_{t,p} = y_{t-1,p}.$
- b. If $a_{t,p} = a_{t-1,p} - 2$ then $|b'_{t,p} - b'_{t-1,p}| \geq \frac{1}{8k}$, in addition note that $|y_{t,p} - y_{t-1,p}| = \frac{1}{4k}.$ Overall, $|y_{t,p} - y_{t-1,p}| = \frac{1}{4k} \leq 2|b'_{t,p} - b'_{t-1,p}| \leq 4|b_{t,p} - b_{t-1,p}|.$
- c. If $a_{t,p} \leq a_{t-1,p} - 3$ then $|b'_{t,p} - b'_{t-1,p}| \geq \frac{1}{4k}$, in addition $|y_{t,p} - y_{t-1,p}| \leq |b'_{t,p} - b'_{t-1,p}| - \frac{1}{4k}.$ Overall, $|y_{t,p} - y_{t-1,p}| \leq |b'_{t,p} - b'_{t-1,p}| - \frac{1}{4k} \leq 2|b'_{t,p} - b'_{t-1,p}| \leq 4|b_{t,p} - b_{t-1,p}|.$
- d. Else, $a_{t,p} = a_{t-1,p} - 1$. In case that $a_{t,p}$ is odd we that $y_{t,p} = y_{t-1,p}.$ Otherwise, let t' be the last time before $t-1$ such that $a_{t',p} \neq a_{t-1,p}.$ It holds that $a_{t',p} \leq a_{t,p} - 2$ and we get, similar to Cases 4b and 4c, $\sum_{i=t'+1}^t |y_{i,p} - y_{i-1,p}| = |y_{t,p} - y_{t',p}| \leq 2|b_{t,p} - b_{t',p}|.$

We show that we can find $t' < t$ such that:

$\sum_{\tau \in [t', t]} |y_{\tau,p} - y_{\tau-1,p}| \leq 4 \cdot \sum_{\tau \in [t', t]} |b_{\tau,p} - b_{\tau-1,p}|.$ Finally, we apply the inductive assumption for $t'.$

5. For time t we note $A = \{p | b_{t,p} < 0.5\}.$ It holds for every p that if $p \in A$ then $y_{t,p} \geq 2 \cdot b_{t,p} - \frac{1}{8k}$, else $y_{t,p} = 1.$ Therefore, if $|A| \leq k$ we are done. Else, $\sum_{p \in P} y_{t,p} = \sum_{p \in A} y_{t,p} + \sum_{p \in P \setminus A} y_{t,p} \geq \sum_{p \in A} (2 \cdot b_{t,p} - \frac{1}{8k}) + |P| - |A| \geq 2(|A| - k) - \frac{|A|}{8k} + |P| - |A| = |P| + |A| - \frac{|A|}{8k} - 2k.$ Now, note that if $k + 1 \leq |A| \leq 2k$, then $|A| \geq k + \frac{|A|}{8k}$ so $|P| + |A| - \frac{|A|}{8k} - 2k \geq n - k.$ Else, $|A| > 2k$ and $|P| + |A| - \frac{|A|}{8k} - 2k \geq n - k. \quad \blacktriangleleft$

A similar process is required for discretizing the value of the x variables. For simplicity we assume there is an additional page z with weight 0 such that

$y_z = \max \left\{ 0, \sum_p y_{t,p} - (n + 1 - k) \right\}.$ For time t and pages $p, q \neq z$ in P : if $x_{t,p,q} \geq 0.5$ then $v_{t,p,q} = 1 - y_{t,p}$, else $v_{t,p,q} = \min \{ 1 - y_{t,p}, x_{t,p,q} \}.$ For z , $v_{t,z,q} = \max \left\{ 0, y_{t,z} - \sum_p v_{t,p,q} \right\}.$

► **Lemma 15.** *The following statements hold:*

1. $\forall t, p, q : v_{t,p,q} \leq 1 - y_{t,p}.$
2. $\forall t, q : \sum_p v_{t,p,q} = y_{t,q}.$
3. $\sum_{p,q,t \geq 1} (w_p + w_q) \cdot v_{t,p,q} \leq 2 \cdot \sum_{p,q,t \geq 1} (w_p + w_q) \cdot x_{t,p,q}.$

Proof.

1. For every page $p \neq z$ the statement holds by definition. For z , since $(1 - b_{t,q}) + \sum_p x_{t,p,q} \geq 1$, we can view it as a rounding of a cache with (at least) one slot. The value of $v_{t,p,z}$ is the empty space in the cache, which is at most the empty space in the fractional cache Y , i.e. $1 - y_{t,z}.$
2. Holds immediately following definition of $v_{t,z,q}.$

3. We can rewrite the sum: $\sum_{p,q,t \geq 1} w_p \cdot v_{t,p,q} + \sum_{p,q,t \geq 1} w_q \cdot v_{t,p,q}$. Since $\sum_q v_{t,p,q} = y_{t,p}$ and $y_{t,p} \leq 2b_{t,p}$, implying $\sum_{p,q,t \geq 1} w_p \cdot v_{t,p,q} \leq 2 \cdot \sum_{p,q,t \geq 1} w_p \cdot x_{t,p,q}$. In addition, for every page q , $v_{t,p,q} \leq 2 \cdot x_{t,p,q}$. ◀

We use Lemma 7.3 from [17] that provides a method for transforming distributions over pages into distributions over cache states. It is immediate from the proof of this lemma that if every distribution over the pages is a multiple of $\frac{1}{L}$, for some $L \in \mathbb{N}$, then the size of the distribution is polynomial in L , n and T .

► **Definition 16.** For a page p and a cache C , $W(C, p) = 0$ if $p \in C$, otherwise $W(C, p) = \min \{w_q | w \in C\}$.

We state here a lemma that summarizes the desired construction and its properties.

► **Lemma 17.** Given a solution (B, X) to the LP, a collection of random integral cache states $R(B, X) = \{R_1, \dots, R_T\}$ can be constructed in polynomial time (in n , k , and T) such that:

1. $\forall t, p$: if $b_{t,p} = 0$, then $p \in R_t$; if $b_{t,p} = 1$, then $p \notin R_t$.
2. $\forall t, p$: $\Pr[p \notin R_t] \leq 4 \cdot b_{t,p}$.
3. $\mathbb{E} \left[\sum_{t \in [1, T], p \in R_t \Delta R_{t-1}} w_p \right] \leq 20 \cdot \sum_{t \in [1, T]} |b_{t,p} - b_{t-1,p}| \cdot w_p$.
4. $\forall t, p$: $\mathbb{E}[W(R(t), p)] \leq 8 \sum_{p,q,t \geq 1} w_p \cdot x_{t,p,q}$.

We use the construction in the above lemma as a black box. When X is obvious from the context, we replace $R(X)$ with R . Thus, we get the following algorithm:

■ **Algorithm 2** DOP Algorithm.

Input: Fractional solution B to LP (31).

- 1: Initialize: let $R(B) = \{R_1, \dots, R_T\}$ (see Lemma 17).
 - 2: **for** time t and request σ_t **do**
 - 3: Set $C_t = R_t$.
 - 4: **if** $\sigma_t \notin R_t$ **then**
 - 5: Evicts the lightest page in R_t and loads σ_t instead.
 - 6: Set $C_t = R_t$.
-

► **Lemma 18.** Algorithm 2 is 60-competitive.

Proof. The expected cost of Step 3 is $20 \cdot \sum_{p,t \geq 1} w_p \cdot |B_{t,p} - B_{t-1,p}|$. In Step 5 the cost is 0 if $\sigma_t \in R_t$ and w_{σ_t} otherwise. Therefore the expected cost of Step 5 at time t is $\sum_{t,p} y_{t,p} \cdot w_p$. Now let us assume for time t that $\sigma_t \notin R_t$. In this case, the cost of Step 6 is equal to $\min \{w_q | q \in R_t\} = W(\sigma_t, C_t)$. Therefore the expected cost of this step is $\mathbb{E}[W(\sigma_t, R(t))]$. From the construction of R_t in [17], this value is at most $8 \cdot \sum_q w_q \cdot v_{t,\sigma_t,q}$. In total, the expected cost of the algorithm is at most 20 times the optimal value of the linear program (31), hence at most 20 times the cost of the best non-adaptive algorithm. With [20, Theorem 1.3] we get that the expected cost is at most $60 \cdot OPT$. ◀

Note that all the cache configurations during the execution of the algorithm contain at most k pages. In addition, for every time t , the request σ_t is loaded in case it is not part of C_t .