Randomized Competitive Algorithms for Generalized Caching

Nikhil Bansal
IBM T. J. Watson Research
Yorktown Heights, USA
nikhil@us.ibm.com

Niv Buchbinder
Computer Science Dept.
Technion, Haifa, Israel
nivb@cs.technion.ac.il

Joseph (Seffi) Naor*
Computer Science Dept.
Technion, Haifa, Israel
naor@cs.technion.ac.il

ABSTRACT

We consider online algorithms for the generalized caching problem. Here we are given a cache of size k and pages with arbitrary sizes and fetching costs. Given a request sequence of pages, the goal is to minimize the total cost of fetching the pages into the cache. We give an online algorithm with competitive ratio $O(\log^2 k)$, which is the first algorithm for the problem with competitive ratio sublinear in k. We also give improved $O(\log k)$-competitive algorithms for the special cases of the Bit Model and Fault model. In the Bit Model, the fetching cost is proportional to the size of the page and in the Fault model all fetching costs are uniform. Previously, an $O(\log^2 k)$-competitive algorithm due to Irani [14] was known for both of these models. Our algorithms are based on an extension of the primal-dual framework for online algorithms which was developed by Buchbinder and Naor [7]. We first generate an $O(\log k)$-competitive fractional algorithm for the problem. This is done by using a strengthened LP formulation with knapsack-cover constraints, where exponentially many constraints are added upon arrival of a new request. Second, we round online the fractional solution and obtain a randomized online algorithm. Our techniques provide a unified framework for caching algorithms and are substantially simpler than those previously used.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms

*Supported by ISF grant 1366/07 and BSF grant 2002276. Part of this work was done while visiting Microsoft Research.

STOC'08, May 17–20, 2008, Victoria, British Columbia, Canada.

1. INTRODUCTION

Caching is one of the earliest and most effective techniques of accelerating the performance of computing systems. Thus, vast amounts of work have been invested in the improvement and refinement of caching techniques and algorithms. In the classic two-level caching problem we are given a collection of n pages and a fast memory (cache) which can hold up to k of these pages. At each time step one of the pages is requested. If the requested page is already in the cache then no cost is incurred, otherwise the algorithm must bring the page into the cache, possibly evicting some other page, and a cost of one unit is incurred. This simple model can be extended in two orthogonal directions. First, the cost of bringing a page into the cache may not be uniform for all pages. This version of the problem is called *weighted caching* and it models scenarios in which the cost of fetching a page is not the same due to different locations of the pages (e.g., main memory, disk, Internet). Second, the size of the pages need not be uniform. This is motivated by web caching where pages have varying sizes. Web caching is an extremely useful technique for enhancing the performance of World Wide Web applications. Since fetching a web page or any other information from the internet is usually costly, it is common practice to keep some of the pages closer to the client. This is done, for example, by the web browser itself by keeping some of the pages locally, and also by internet providers that maintain proxy servers for exactly the same purpose.

We study here several models of generalized caching from a competitive analysis point of view. In the most general setting, called the *General model*, pages have both non-uniform sizes and costs. Two commonly studied special cases, where the pages can have variable size, are the so-called *Bit model* and *Fault model*. In the Bit model, the cost of fetching a page is equal to its size, thus minimizing the fetching cost corresponds to minimizing the total traffic in the network. In the Fault model, the fetching cost is uniform for all pages, thus corresponding to the number of times a user has to wait for a page to be retrieved.

1.1 Previous Work

The caching model where both size and cost are uniform is very well understood. In their seminal paper, Sleator and Tarjan [17] showed that any deterministic algorithm is at least k-competitive, and also showed that LRU (Least Recently Used) is exactly k-competitive. When randomization is allowed, Fiat et al. [12] designed the Randomized Marking algorithm which is $2H_k$-competitive against an oblivious

adversary, where H_k is the k-th Harmonic number. They also showed that any randomized algorithm is at least H_k-competitive. Subsequently, a tight H_k competitive algorithm was obtained [16, 1]. More generally, the (h, k)-version of the problem has also been studied. Here the online algorithm with cache size k is compared to the offline algorithm with cache size $h \leq k$. A tight $k/(k-h+1)$ guarantee is known for the deterministic case [17], and a $2\ln(k/(k-h+1)$ guarantee is also known for the randomized case [18].

For weighted caching (uniform size but non-uniform costs), a tight k-competitive deterministic algorithm follows from the more general work of Chrobak et al. [10] for the k-server problem on trees. Subsequently, Young [19] gave a tight $k/(k-h+1)$-competitive deterministic algorithm for the more general (h, k)-caching problem. The problem of determining the randomized competitiveness of weighted caching remained open for a long time. Irani [15] gave an $O(\log k)$-competitive algorithm for the two weight case, i.e. when each page weight is either 1 or some fixed $M > 1$. The case of arbitrary weights was settled only recently by Bansal, Buchbinder and Naor [3] who designed an $O(\log k)$-competitive algorithm for the problem and another $O(\ln(k/(k-h+1))$-competitive algorithm for the (h, k)-version.

General caching where the page sizes are also non-uniform is substantially harder. In contrast to uniform page size caching, even the offline version of the problem is NP-hard, as it captures the knapsack problem[1]. Following a sequence of results [14, 2, 11], Bar-Noy et al. [4] gave a 4-approximation for the problem based on the local-ratio technique. This is currently the best known approximation for (offline) general caching. For the online case it is known that LRU is $(k+1)$-competitive for the Bit model and also for the Fault model [14], where k denotes the ratio between cache size and the size of the smallest page. Later on, Cao and Irani [8] and Young [20] gave a $(k+1)$-competitive algorithm for the General model based on a generalization of the Greedy-Dual algorithm of Young [19]. An alternate proof of this result was obtained by [11]. When randomization is allowed, Irani [14] designed an $O(\log^2 k)$-competitive algorithm for both Fault and Bit models. These algorithms are very complicated and are based on an approach combining offline algorithms with the Randomized Marking algorithm. For the General model, no $o(k)$ randomized algorithms are known. There has been extensive work on caching in other directions, and we refer the reader for further details to the excellent text by Borodin and El-Yaniv [6] and to the survey by Irani [13] on paging.

1.2 Results and techniques

We study in this work all three models mentioned above: the Bit model in which the cost of fetching a page is proportional to its size, the Fault model in which the cost of fetching each page is uniform, and the general model in which the cost of fetching each page is arbitrary. We prove the following theorem:

THEOREM 1.1. *There exist randomized algorithms for the generalized caching problem that are:*

- $O(\log k)$*-competitive for the Bit model.*
- $O(\log k)$*-competitive for the Fault model.*
- $O(\log^2 k)$*-competitive for the general model.*

Our approach for designing online algorithms for generalized caching follows two conceptual steps. First, an $O(\log k)$-competitive algorithm is developed for the fractional generalized caching problem. In the fractional problem the algorithm is allowed to keep fractions of pages as long as the total (fractional) size of the pages in the cache does not exceed the cache size. The cost of fetching an ϵ fraction of a page is then defined to be ϵ times the fetching cost of the whole page. The second step is to obtain a randomized algorithm for the problem. To this end, we maintain in an online manner, a distribution of cache states that is (almost) consistent with the fractional solution in hand. We show how to map the changes in the fractional distribution on pages to changes in the distribution of caches so that the cost incurred is not much more than the fractional cost[2].

The fractional algorithm we propose is based on the primal-dual framework developed by [7] for online packing and covering problems. This framework was recently extended and shown to be very useful for the weighted caching problem [3]. However, solving general caching with non-uniform page sizes requires several new ideas. One fundamental problem is that the natural LP for generalized caching (where it is required that the total size of the pages in the cache is at most k) can have an integrality gap of $\Omega(k)$, and therefore is not suitable for our purposes. For example, suppose the cache size is $k = 2\ell - 1$, and there are two pages of size ℓ, requested alternately. Only one page can be in the cache at any time and hence there is a cache miss in each request. A fractional solution on the other hand can keep almost one unit of each page and then it only needs to fetch an $O(1/k)$ fraction of a page in each request. To get around this problem, we strengthen the LP for generalized caching by adding exponentially many *knapsack-cover* inequalities, a technique introduced by [9]. (We note that the local ratio algorithm of [4] for generalized caching coupled with the techniques of [5] also yields knapsack-cover inequalities.) We then show how to apply a primal-dual technique to this LP to derive an $O(\log k)$-competitive fractional algorithm for generalized caching.

We obtain a randomized integral algorithm (online) from the fractional solution by generating a distribution on cache states, while bounding the payment to be at most c times the fractional cost. This part is model-dependent and it is done by maintaining a distribution with several (different) properties. We show: (i) Each cache state in the support of the distribution is valid, i.e., no duplicate pages and total size of pages is at most the cache size), and (ii) The distribution with the desired properties is maintained throughout the execution of the algorithm in an online manner. To this end, we partition the set of pages into classes. The partitioning is model-dependent. Our main tool is the notion of a *balanced distribution* over cache states in which the states in the support evict approximately the same number of pages from each class of pages. The constructions are presented in an increasing order of complexity. The most complex construction is for the Fault model where the partitioning into classes is dynamic. The analysis turns out to be rather involved and requires the use of amortized analysis to bound its cost.

[1] It remains NP-hard for the Bit model. For the Fault model it is open whether the problem is polynomially solvable [14].

[2] This mapping poses many challenges even if all pages have size 1; the reader is referred to a simple example in [3].

2. PRELIMINARIES

In the general caching problem there is a cache of size k and n pages of sizes $w_1 \leq w_2 \leq \ldots \leq w_n$, belonging to $\in [1, k]$. It is not assumed that page sizes are integral and k can be viewed as the ratio between cache size and the smallest page size. For any subset S of pages, let $W(S) = \sum_{i \in S} w_i$ be the sum of the sizes of the pages in S. Page p has a fetching cost of c_p. With this terminology, in the Fault model $c_p = 1$ for each page p, in the Bit model $c_p = w_p$ for each page p, and in the general model c_p and w_p are arbitrary.

2.1 LP formulation for general caching

Consider the following (natural) integer program for the (offline) general caching problem. Instead of charging for fetching pages into the cache we charge for evicting them, thus increasing the cost of any algorithm by at most an additive term (fetching the last k pages is "free"). Let $x(p, j)$ be an indicator variable for the event that page p is evicted from the cache between the jth time it is requested and the $(j+1)$st time it is requested. If $x(p, j) = 1$, we can assume without loss of generality that page p is evicted in the first time slot following the jth time it is requested. (As we later discuss, this assumption is not necessarily true in the online case.) For each page p, denote by $t(p, j)$ the time it is requested for the jth time, and denote by $r(p, t)$ the number of times page p is requested until time t (including t). For any time t, let $B(t) = \{p \mid r(p, t) \geq 1\}$ denote the set of pages that were requested until time t (including t). Let p_t be the page that was requested at time t. We need to satisfy the constraint that at any time t, the currently requested page must be present in the cache, i.e. $x(p_t, r(p, t)) = 0$, and that the total space used by pages in $B(t)$ can be at most k. Since w_{p_t} space is already used by p_t, this implies that at most $k - w_{p_t}$ space can be used by pages in $B(t) \setminus \{p_t\}$. Equivalently, pages in $B(t) \setminus \{p_t\}$ with cumulative size at least $W(B(t)) - w_{p_t} - (k - w_{p_t}) = W(B(t)) - k$ must be absent from the cache at time t. This gives the following exact formulation of the problem.

$$\min \quad \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} c_p \cdot x(p, j)$$

For any time t: $\sum_{p \in B(t) \setminus \{p_t\}} w_p x(p, r(p, t)) \geq W(B(t)) - k$

For any p, j: $x(p, j) \in \{0, 1\}$

In a fractional solution, we relax $x(p, j)$ to take any value between 0 and 1. However, as discussed in the Introduction this LP relaxation has a large gap and is therefore unsuitable for our purposes.

To get around this problem, we use an idea introduced by Carr et al. [9] of adding exponentially many *knapsack cover* inequalities. These constraints are redundant in the integer program, but they dramatically reduce the integrality gap of the LP relaxation. There are two main ideas. First, consider a subset of pages $S \subset B(t)$ such that $p_t \in S$ and $W(S) > k$. The pages in $S \setminus \{p_t\}$ can occupy at most $k - w_{p_t}$ size in the cache at time t. Thus, at least $W(S) - w_{p_t} - (k - w_{p_t}) = W(S) - k$ cumulative size of pages in $S \setminus \{p_t\}$ must be absent from the cache. Hence, we can add the constraint $\sum_{p \in S \setminus \{p_t\}} w_p x(p, r(p, t)) \geq W(S) - k$ for each such set S at time t. The second idea is that for each such constraint, we can truncate the size of a page to be equal to the right hand size of the constraint, i.e. we have $\sum_{p \in S \setminus \{p_t\}} \min(W(S) - k, w_p) x(p, r(p, t)) \geq W(S) - k$. Clearly, truncating the size has no effect on the integer program. Our LP is as follows.

$$\min \quad \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} c_p \cdot x(p, j)$$

For any time t and any set of requested pages $S \subseteq B(t)$ such that $p_t \in S$ and $W(S) > k$:

$$\sum_{p \in S \setminus \{p_t\}} \min\{W(S) - k, w_p\} x(p, r(p, t)) \geq W(S) - k \quad (1)$$

$$\text{For any } p, j: \qquad 0 \leq x(p, j) \leq 1 \quad (2)$$

We now note a simple observation about knapsack cover inequalities that will be quite useful.

OBSERVATION 2.1. *Given a fractional solution x, if a knapsack cover inequality is violated for a set S at time t, then it is also violated for the set $S' = S \setminus \{p : x(p, r(p, t)) = 1\}$, obtained by omitting pages which are already completely evicted from the cache.*

PROOF. Suppose Inequality (1) is violated for some S and $x(p, r(p, t)) = 1$ for $p \in S$. First, it must be the case that $\min(W(S) - k, w_p) < W(S) - k$, otherwise (1) is trivially satisfied. Suppose we delete p from S. The right hand size decreases by exactly w_p. The left hand side decreases by w_p and possibly more since the term $\min(W(S) - k, w_{p'})$ may decrease for pages $p' \in S$. Thus, Inequality (1) is also violated for $S \setminus \{p\}$. The result follows by applying the argument repeatedly. □

Observation 2.1 implies that in any feasible solution to the constraints given by (1), it does not help to have $x(i, j) > 1$. Hence, it can be assumed that $x(i, j) \leq 1$ without loss of generality. Thus, we can drop the upper bounds on $x(i, j)$ and simplify the LP formulation to:

$$\min \quad \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} c_p \cdot x(p, j) \qquad \text{(LP-Caching)}$$

For any time t and any set of requested pages $S \subseteq B(t)$ such that $p_t \in S$ and $W(S) > k$:

$$\sum_{p \in S \setminus \{p_t\}} \min\{W(S) - k, w_p\} x(p, r(p, t)) \geq W(S) - k \quad (3)$$

$$\text{For any } p, j\text{: } 0 \leq x(p, j) \quad (4)$$

In the dual program, there is a variable $y(t, S)$ for each time t and set $S \subseteq B(t)$ such that $p_t \in S$ and $W(S) > k$. The dual program is the following:

$$\max \quad \sum_{t} \sum_{S \subseteq B(t), p_t \in S} (W(S) - k)\, y(t, S)$$

For each page p and the jth time it is requested:

$$\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \mid p \in S} \min\{W(S) - k, w_p\} y(t, S) \leq c_p \quad (5)$$

We will sometimes denote $\min\{W(S) - k, w_p\}$ by $\tilde{w}_p^S$.

3. COMPUTING A COMPETITIVE FRACTIONAL SOLUTION

Our online caching algorithm produces fractional primal and dual solutions to LP-Caching. In the online case, the constraints of LP-Caching are revealed one-by-one. At any time t, exponentially many new linear knapsack-cover constraints are revealed to the online algorithm. The goal is to produce a feasible assignment to the (primal) variables that satisfies all the constraints. Since there are exponentially many constraints, this process may not run in polynomial-time. However, as we show next, for our purposes we can make the algorithm run in polynomial time.

Consider variable $x(p,j)$. In the offline case, we can assume without loss of generality that the value of $x(p,j)$ is determined at time $t(p,j)+1$. However, this is not necessarily true in the online case; here after time $t(p,j)$, the page p is gradually evicted until it is requested next at time $t(p,j+1)$. Thus, in the online setting, we stipulate that the values assigned to $x(p,j)$ in the time interval $[t(p,j)+1, t(p,j+1)-1]$ form a monotonically non-decreasing sequence. We start with a high level description of the algorithm. Upon arrival of the new constraints at time t, if all constraints are already satisfied, then the algorithm does nothing. Otherwise, the algorithm needs to satisfy all the current constraints by increasing some of the primal variables. We call a set S minimal if $x(p, r(p,t)) < 1$ for each $p \in S$. By Observation 2.1, it suffices to consider primal constraints corresponding to minimal sets. Satisfying all the constraints at time t guarantees that there is enough space (fractionally) in the cache to fetch the new page.

To this end, the algorithm arbitrarily picks an unsatisfied primal constraint corresponding to some minimal set S and starts increasing continuously its corresponding dual variable $y(t,S)$. This, in turn, tightens some of the dual constraints corresponding to primal variables $x(p,j)$ whose current value is 0. Whenever such an event happens, the value of $x(p,j)$ is increased from its initial setting of 0 to $1/k$. Thus, during the time preceding the increase of $x(p,j)$ from 0 to $1/k$, page p cannot be evicted at all from the cache. This part is somewhat similar to what happens in the Randomized Marking algorithm. Meanwhile, variables $x(p,j)$ which are already set to $1/k$ are increased (continuously) according to an exponential function of the new dual variable $y(t,S)$. Note that this exponential function is equal to $1/k$ when the constraint is tight. Thus, the algorithm is well defined. When variable $x(p,j)$ reaches 1, the set S is no longer minimal, and page p is dropped from S. As a result, from this time on, the value of $x(p,j)$ remains 1. When this primal constraint is satisfied the algorithm continues on to the next infeasible primal constraint.

Since there are exponentially many primal constraints in each iteration this process may not be polynomial. However, the rounding process we design in Section 4 does not need the solution to satisfy all primal constraints. Specifically, for each model we show that there exists a (different) value $\gamma > 1$ such that the algorithm needs to guarantee that at time t the primal constraint of the set $S = \{p \mid x(p, r(p,t)) < \frac{1}{\gamma}\}$ is satisfied. Thus, the algorithm may actually consider only that set[3]. Fortunately, the requirement of the online primal-dual framework that variables can only increase monotonically makes this task much simpler. In particular, as the primal variables increase some pages reach $1/\gamma$ and "leave" the set S. The algorithm then tries to satisfy the set S' that contains the rest of the pages. Since pages can only leave S, this process may continue for at most n rounds. We defer a detailed discussion of this issue to the full version of the paper. For simplicity, we describe the algorithm that satisfies all the constraints. The algorithm is presented in a continuous fashion, but it can easily be implemented in a discrete fashion. The algorithm is the following:

Fractional Caching Algorithm: At time t, when page p_t is requested:

- Set the new variable: $x(p_t, r(p_t,t)) \leftarrow 0$. (It can only be increased at times $t' > t$.)
- Until **all** the primal constraint corresponding to time t are satisfied do the following:
- Assume that the primal constraint of a minimal set S is not satisfied.

1. Increase variable $y(t,S)$ continuously; for each variable $x(p,j)$ such that $p \in S \setminus \{p_t\}$:
2. If $x(p,j) = 1$, then remove p from S, i.e. $S \leftarrow S \setminus \{p\}$.
3. If $x(p,j) = 0$ and $\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S: p \in S} \tilde{w}_p^S y(t,S) = c_p$, then $x(p,j) \leftarrow 1/k$.
4. If $1/k \le x(p,j) < 1$, increase $x(p,j)$ according to the following function:

$$\frac{1}{k} \cdot \exp\left(\frac{1}{c_p} \left[\left(\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S: p \in S} \tilde{w}_p^S y(t,S) \right) - c_p \right] \right)$$

where $\tilde{w}_p^S$ denotes $\min\{W(S) - k, w_p\}$.

THEOREM 3.1. *The algorithm is $O(\log k)$-competitive.*

PROOF. First, we note that the primal solution generated by the algorithm is feasible. This follows since, in each iteration, the variables $x(p,j)$ are increased until all new primal constraints are satisfied. Also, each variable $x(p,j)$ is never increased to be greater than 1.

Next, we show that the dual solution that we generate is feasible up to an $O(\log k)$ factor. Whenever $x(p,j)$ reaches 1, the variables $y(t,S)$ for sets S containing p do not increase anymore, and hence the value of $x(p,j)$ does not change any more. Thus, for the dual constraint corresponding to page p and the jth time it is requested, we get that:

$$x(p,j) = \frac{1}{k} \cdot \exp\left(\frac{1}{c_p} \left[\left(\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S: p \in S} \tilde{w}_p^S y(t,S) \right) - c_p \right] \right) \le 1$$

where $\tilde{w}_p^S = \min\{W(S) - k, w_p\}$. Simplifying, we get that:

$$\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \mid p \in S} \min\{W(S) - k, w_p\} y(t,S) \le c_p(1 + \ln k).$$

[3]In general, for knapsack cover constraints in an offline setting, all possible subsets may be needed since it is not clear apriori which set S will have this property, nor can it be expressed as a linear or even a convex program. See [9] for more details.

Thus, the dual solution can be made feasible by scaling it down by a factor of $(1+\ln k)$. We now prove that the primal cost is at most twice the dual profit, which means that the primal solution produced is $O(\log k)$-competitive.

We partition the primal cost into two parts, C_1 and C_2. Let C_1 be the contribution to the primal cost from Step (3) of the algorithm, due to the increase of variables $x(p,j)$ from 0 to $1/k$. Let C_2 be the contribution to the primal cost from Step (4) of the algorithm, due to the incremental increases of the variable $x(p,j)$ according to the exponential function.

Bounding C_1.

Let $\tilde{x}(p,j) = \min(x(p,j), \frac{1}{k})$. Our goal is to bound the term $\sum_{p=1}^{n}\sum_{j=1}^{r(p,t)} c_p\tilde{x}(p,j)$. To do this, we need two observations. First, from design of the algorithm, it follows that if $x(p,j) > 0$, and equivalently if $\tilde{x}(p,j) > 0$, then:

$$\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \mid p\in S} \min\{W(S)-k, w_p\} y(t,S) \geq c_p \qquad (6)$$

We shall refer to (6) as primal complementary slackness.

Next, in the dual solution if $y(t,S) > 0$, then:

$$\sum_{p\in S\setminus\{p_t\}} \min\{W(S)-k, w_p\}\tilde{x}(p,r(p,t)) \leq W(S)-k \qquad (7)$$

We shall refer to (7) as dual complementary slackness. To see why (7) holds, consider the following two cases depending on whether $W(S) \geq k+1$ or not. Recall that $\tilde{x}(p,r(p,t)) \leq 1/k$ for all pages. If $W(S) \geq k+1$ then:

$$\sum_{p\in S\setminus\{p_t\}} \frac{1}{k}\cdot\min\{W(S)-k, w_p\} \leq \frac{1}{k}\cdot\sum_{p\in S\setminus\{p_t\}} w_p$$
$$= \frac{W(S)-w(p_t)}{k} \leq \frac{W(S)-1}{k} \leq W(S)-k.$$

If $W(S) < k+1$, then the set S contains at most k pages. In this case we get that:

$$\sum_{p\in S\setminus\{p_t\}} \frac{1}{k}\cdot\min\{W(S)-k, w_p\} \leq \frac{1}{k}\cdot\sum_{p\in S\setminus\{p_t\}} (W(S)-k)$$
$$\leq \frac{k-1}{k}\cdot(W(S)-k) \leq W(S)-k.$$

The last inequality follows since $W(S) \geq k$. These primal and dual complementary slackness conditions imply the following.

$$\sum_{p=1}^{n}\sum_{j=1}^{r(p,t)} c_p\tilde{x}(p,j) \qquad (8)$$
$$\leq \sum_{p=1}^{n}\sum_{j=1}^{r(p,t)} \left(\sum_{t=t(p,j)+1}^{t(p,j+1)-1}\sum_{S \mid p\in S} \tilde{w}_p^S y(t,S)\right)\tilde{x}(p,j) \qquad (9)$$
$$= \sum_{t}\sum_{S\subseteq B(t), p_t\in S}\left(\sum_{p\in S\setminus\{p_t\}} \tilde{w}_p^S\tilde{x}(p,r(p,t))\right) y(t,S) \qquad (10)$$
$$\leq \sum_{t}\sum_{S\subseteq B(t), p_t\in S} (W(S)-k)\, y(t,S). \qquad (11)$$

Inequality (9) follows from Inequality (6), Equality (10) follows by changing the order of summation, and Inequality (11) follows from Inequality (7). Thus, C_1 is at most the profit of a *feasible* dual solution multiplied by $(1+\ln k)$.

Bounding C_2.

We bound the derivative of the primal cost of variables $x(p,j)$ in Step (4) by the derivative of the dual profit accrued in the same round. Variables $x(p,j)$ that have already reached the value of 1 do not contribute anymore to the primal cost. The derivative of a variable $x(p,j)$, $1/k \leq x(p,j) < 1$, as a function of $y(t)$ is:

$$\frac{dx(p,j)}{dy(t,S)} = \frac{\min\{W(S)-k, w_p\}}{c_p}\cdot x(p,j). \qquad (12)$$

Therefore, the derivative of the primal is at most:

$$\frac{dX}{dy(t,S)} = \sum_{p\in S\setminus\{p_t\}: x(p,r(p,t))<1} \tilde{w}_p^S x(p,r(p,t))$$
$$\leq W(S)-k = \frac{dY}{dy(t,S)}.$$

The inequality in the second step above follows since the primal constraint of the set S is unsatisfied yet. Thus, C_2 is at most the profit of a feasible dual solution multiplied by $(1+\ln k)$.

Completing the analysis.

It follows that $C_1 + C_2$ is at most twice the profit of a *feasible* dual solution multiplied by $(1+\ln k)$. Note that the profit of any dual feasible solution is always a lower bound on the optimal solution. Therefore, we conclude by weak duality that the algorithm is $O(\log k)$-competitive. □

3.1 The (h,k)-general caching problem

The above method can be extended to obtain an $O(\ln(k/(k-h+1)))$ competitive fractional algorithm for the (h,k)-general caching problem, where an online cache of size k is compared with an offline cache of size h. The details are deferred to the full version of the paper.

4. ROUNDING THE FRACTIONAL SOLUTION ONLINE

In this section we show how to obtain a randomized online algorithm from the fractional solution generated previously. For convenience of analysis, throughout this section we consider the (equivalent) cost version of the problem where we pay c_p for both fetching and evicting a page p. This assumption can change the cost of the fractional solution by at most a factor of two. At any time step, the LP solution $x_1,\ldots,x_n$, where we denote by $x_p \triangleq x(p,r(p,t))$, specifies the probability that each of the pages is absent from the cache. However, to obtain an actual randomized algorithm we need to specify a probability distribution over the various cache states that is consistent with the LP solution. That is, we need to simulate the moves of the LP over the set of pages by consistent moves over the actual cache states. We adopt the following approach to do this simulation.

Let $\gamma \geq 1$ be a parameter and set $y_p = \min(\gamma x_p, 1)$. Let μ be a distribution on subsets of pages. We say that μ is consistent with y (or γ-consistent with x) if μ induces the distribution y on the page set. That is,

$$\forall p: \qquad \sum_{D} A_p^D\cdot\mu(D) = y_p, \qquad (13)$$

where, for a set of pages D, $A_p^D = 1$ if $p\in D$ and 0 otherwise. We will view μ as a distribution over the complement of the cache states. To be a meaningful simulation, it suffices to require the following.

1. *Size Property:* For any set D with $\mu(D) > 0$, the sum of the sizes of the pages in D is at least $W(B(t)) - k$. That is, D corresponds to the complement of a valid cache.
2. *Bounded Cost Property:* If y changes to y' while incurring a fractional cost of d, the distribution μ can be changed to another distribution μ' which is consistent with y', while incurring a (possibly amortized) cost of at most βd, where $\beta > 0$.

It is easy to see that if x_p changes by ϵ, then y_p changes by at most $\gamma\epsilon$. Hence, given a fractional algorithm with competitive ratio c, the existence of a simulation with the above properties implies an actual randomized online algorithm with competitive ratio $\gamma\beta c$. We provide three different simulation procedures for the Bit model, General model, and the Fault Model. These are organized in increasing order of complexity.

4.1 The Bit Model

In this section we will show how to obtain an $O(\log k)$-competitive randomized algorithm for the general caching problem in the Bit model. Let $U \triangleq \lfloor \log_2 k \rfloor$. For $i = 0$ to U, we define the size class $S(i)$ to be the set of pages of sizes between 2^i and less than size 2^{i+1}. Formally, $S(i) = \{p \mid 2^i \leq w_p < 2^{i+1}\}$. Let $x_1, \ldots, x_n$ be the LP solution at the current time step. Recall that it satisfies the knapsack cover inequalities for all subsets. For each page p let $y_p = \min\{1, 3x_p\}$ (i.e. $\gamma = 3$).

DEFINITION 4.1 (BALANCED SUBSETS). *We say that a subset of pages D is balanced with respect to y if:*

1. *If $y_p = 1$ then p is evicted in all cache states, i.e. $A_p^D = 1$ for all D with $\mu(D) > 0$.*
2. *The following holds for all $0 \leq j \leq U$:*

$$\left\lfloor \sum_{i=j}^{U} \sum_{p \in S(i)} y_p \right\rfloor \leq \sum_{i=j}^{U} \sum_{p \in S(i)} A_p^D \leq \left\lceil \sum_{i=j}^{U} \sum_{p \in S(i)} y_p \right\rceil. \quad (14)$$

We first show that the size property follows from the requirement that sets are balanced.

LEMMA 4.2. *Let x and y be defined as above. Then, for any subset D which is balanced with respect to y, the sum of the sizes of all the pages in D is at least $W(B(t)) - k$.*

We first prove a simple mathematical claim.

CLAIM 4.3. *Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be two sequences of non-negative real numbers and let $0 = a_0 \leq a_1 \leq a_2 \leq \ldots, \leq a_n$ be a non-decreasing sequence of positive numbers. If for every $1 \leq j \leq n$: $\sum_{i=j}^{n} x_i \geq -1 + (\sum_{i=j}^{n} y_i)$, then: $\sum_{i=1}^{n} a_i x_i \geq -a_n + \sum_{i=1}^{n} a_i y_i$.*

PROOF. For every j, $1 \leq j \leq n$, multiply the jth inequality by $(a_j - a_{j-1})$ (which is non negative), yielding:

$$(a_j - a_{j-1}) \sum_{i=j}^{n} x_i \geq -(a_j - a_{j-1}) + (a_j - a_{j-1}) \sum_{i=j}^{n} y_i.$$

Summing up over all the inequalities yields the desired result. □

PROOF. *(Lemma 4.2).* For the proof it suffices to use the LHS of condition (14) (i.e., the lower bound). Let $S' \subseteq S$ be the set of pages with $y_p < 1$, and let $S'(i) = S' \cap S(i)$ be the class i pages in S'. Since $A_p^D = 1$ whenever $y_p = 1$, condition (14) implies that for every $0 \leq j \leq U$:

$$\sum_{i=j}^{U} \sum_{p \in S'(i)} A_p^D \geq \left\lfloor \sum_{i=j}^{U} \sum_{p \in S'(i)} y_p \right\rfloor \geq \left(\sum_{i=j}^{U} \sum_{p \in S'(i)} y_p \right) - 1. \quad (15)$$

The sum of the sizes of the pages in D is $\sum_{p \in S} w_p A_p^D$. Since $A_p^D = 1$ for $p \in S \backslash S'$, it suffices to show that $\sum_{p \in S'} w_p A_p^D \geq W(S') - k$ for the proof. Consider the following:

$$\begin{aligned}
\sum_{p \in S'} w_p A_p^D &\geq \sum_{p \in S'} \min\{w_p, W(S') - k\} A_p^D \\
&= \sum_{i=0}^{U} \sum_{p \in S'(i)} \min\{w_p, W(S') - k\} A_p^D \\
&\geq \frac{1}{2} \sum_{i=0}^{U} \sum_{p \in S'(i)} \min\{2w_p, W(S') - k\} A_p^D \\
&\geq \frac{1}{2} \sum_{i=0}^{U} \min\{2^{i+1}, W(S') - k\} \sum_{p \in S'(i)} A_p^D \quad (16) \\
&\geq -\frac{1}{2} \min\{2^{U+1}, W(S') - k\} \\
&\quad + \frac{1}{2} \sum_{i=0}^{U} \min\{2^{i+1}, W(S') - k\} \sum_{p \in S'(i)} y_p \quad (17) \\
&\geq -\frac{1}{2} (W(S') - k) \\
&\quad + \frac{1}{2} \sum_{i=0}^{U} \sum_{p \in S'(i)} \min\{w_p, W(S') - k\} y_p \quad (18) \\
&\geq -\frac{1}{2} (W(S') - k) + \frac{3}{2} (W(S') - k) \geq W(S') - k. \quad (19)
\end{aligned}$$

Here, Inequality (16) follows since $w_p \geq 2^i$ for each $p \in S'(i)$. Inequality (17) follows by applying Claim 4.3 with $a_i = \min\{2^{i+1}, W(S') - k\}$, $x_i = \sum_{p \in S'(i)} A_p^D$ and $y_i = \sum_{p \in S'(i)} y_p$, and observing that (15) implies that the conditions of the claim are satisfied. Inequality (18) follows since $w_p < 2^{i+1}$, and finally Inequality (19) follows since by the LP knapsack constraints, and the fact that $y_p = 3x_p$ for each $p \in S'$:

$$\begin{aligned}
&\sum_{i=0}^{U} \sum_{p \in S'(i)} \min\{w_p, W(S') - k\} y_p \\
&= 3 \sum_{p \in S'} \min\{w_p, W(S') - k\} x_p \geq 3(W(S') - k).
\end{aligned}$$

□

We show how to maintain the bounded cost property using both the LHS and RHS of condition (14).

LEMMA 4.4. *Let μ be any distribution on balanced sets that is consistent with y. Then the cost property holds with $\beta = 10$. That is, if y changes to y' while incurring a fractional cost of d, then the distribution μ can be modified to another distribution μ' over balanced sets such that μ' is consistent with y' and the cost incurred while modifying μ to μ' is at most $10d$.*

PROOF. By considering each page separately, it suffices to show that the property holds whenever y_p increases or decreases for some page p. Assume first that the weight y_p of page p for $p \in S(i)$ is increased by ϵ. The argument when y_p is decreased is analogous. Page p belongs to $S(i)$, and so $w_p \geq 2^i$. Thus, the fractional cost is at least $\epsilon 2^i$.

We construct μ' as follows. To ensure the consistency with y',i.e., Equation (13), we add page p to ϵ measure of the sets D that do not contain p. Since this is the Bit model, this incurs a cost of at most $2^{i+1}\epsilon$. However this may violate condition (14) for classes $j \leq i$. We iteratively fix condition (14) starting with class i. Consider class i. Let $s = \lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y_p \rceil$ and suppose first that $\lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y'_p \rceil$ remains equal to s. Then in μ', let ϵ' be the measure of sets that have $s+1$ pages in classes i or higher. Note that $\epsilon' \leq \epsilon$. Consider the sets with $s-1$ pages in classes i or higher and arbitrarily choose ϵ' measure of these (this is possible since $s = \lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y'_p \rceil$). Arbitrarily pair the sets with $s+1$ pages to those with $s-1$ pages. Consider any pair of sets (D, D'). Since μ' satisfies condition (14) is for class $i+1$, the number of pages in D and D' that lie in classes $i+1$ or higher differ by at most 1. Hence, $D \setminus D'$ contains some class i page. We move this page from D to D'. Note that (14) is satisfied for i after this procedure. Now, consider the case when $\lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y'_p \rceil$ increases to $s+1$. Note that in this case, the condition (14) is be violated for class i for at most $\epsilon' \leq \epsilon$ of sets that have precisely $s-1$ pages in classes i or higher. We arbitrarily pair the classes with $s-1$ to pages to those with $s+1$ pages and apply the argument above. The total cost incurred in this step is at most $(2\epsilon') \cdot 2^{i+1} \leq 2^{i+2}\epsilon$.

After applying the above procedure to fix class i, condition (14) might be violated for class $i-1$ for at most ϵ measure of sets. We apply the matching procedure sequentially to $i-1$ and lower classes incurring an additional cost of $\sum_{j=0}^{i-1} 2\epsilon \cdot 2^{j+1} < 4\epsilon 2^i$. Thus the total cost incurred is at most $10\epsilon 2^i$. □

THEOREM 4.5. *There is an $O(\log k)$-competitive algorithm for the general caching problem in the Bit model.*

4.2 The General Cost Model

In this section we study the General cost model and show how to obtain an $O(\log^2 k)$-competitive randomized caching algorithm for this model. Let $U \triangleq \lfloor \log_2 k \rfloor$. Let $C = \lfloor \log_2 C_{\max} \rfloor$. For $i = 0$ to U, and $j = 0$ to C, we define $S(i,j)$ to be the set of pages of sizes at least 2^i and less than 2^{i+1}, and fetching cost between 2^j and less than 2^{j+1}. Formally, $S(i,j) = \{p \mid 2^i \leq w_p < 2^{i+1} \text{ and } 2^j \leq c_p < 2^{j+1}\}$. Let $x_1, \ldots, x_n$ be the LP solution at the current time step that satisfies the knapsack cover inequalities for all subsets. Let $\gamma = U+3$. Thus, for each page p, $y_p = \min\{1, (U+3) \cdot x_p\} = O(\log k) \cdot x_p$.

DEFINITION 4.6. *A set D of pages is balanced with respect to y if the following two conditions hold:*

1. *If $y_p = 1$ then p is evicted in all cache states, i.e. $A_p^D = 1$ for all D with $\mu(D) > 0$.*
2. *For each size class $0 \leq i \leq U$, it holds that for each $0 \leq j \leq \lfloor \log C_{\max} \rfloor$:*

$$\left\lfloor \sum_{z=j}^{C} \sum_{p \in S(i,z)} y_p \right\rfloor \leq \sum_{z=j}^{C} \sum_{p \in S(i,z)} A_p^D \leq \left\lceil \sum_{z=j}^{C} \sum_{p \in S(i,z)} y_p \right\rceil . \quad (20)$$

We first show that the size property follows from the requirement that the sets are balanced.

LEMMA 4.7. *Let x and y be defined as above. Then, for any subset D that is balanced with respect to y, the sum of sizes of all pages in D is at least $W(B(t)) - k$.*

PROOF. For the proof it suffices to use the LHS of condition (20) (i.e., the lower bound). Let S' denote the subset of pages with $y_p < 1$. As $y_p = 1$ whenever $A_p^D = 1$, it suffices to show that $\sum_{p \in S'} w_p A_p^D \geq W(S') - k$. Moreover, condition (20) implies that for any $0 \leq i \leq U$:

$$\sum_{z=0}^{C} \sum_{p \in S'(i,z)} A_p^D \geq \lfloor \sum_{z=0}^{C} \sum_{p \in S'(i,z)} y_p \rfloor \geq -1 + \sum_{z=0}^{C} \sum_{p \in S'(i,z)} y_p. \quad (21)$$

Thus, the total size of pages from S' that are in D can be lower bounded as follows:

$$\begin{aligned}
\sum_{p \in S'} w_p A_p^D &\geq \sum_{p \in S'} \min\{w_p, W(S') - k\} A_p^D \\
&= \sum_{i=0}^{U} \sum_{j=0}^{C} \sum_{p \in S'(i,j)} \min\{w_p, W(S') - k\} A_p^D \\
&\geq \frac{1}{2} \sum_{i=0}^{U} \sum_{j=0}^{C} \sum_{p \in S'(i,j)} \min\{2w_p, W(S') - k\} A_p^D \\
&\geq \frac{1}{2} \sum_{i=0}^{U} \min\{2^{i+1}, W(S') - k\} \sum_{j=0}^{C} \sum_{p \in S'(i,j)} A_p^D \quad (22) \\
&\geq \frac{1}{2} \sum_{i=0}^{U} \min\{2^{i+1}, W(S') - k\} \left(-1 + \sum_{j=0}^{C} \sum_{p \in S'(i,j)} y_p \right) \quad (23) \\
&\geq -\frac{U+1}{2}(W(S') - k) \\
&\quad + \frac{1}{2} \sum_{i=0}^{U} \sum_{j=0}^{C} \sum_{p \in S'(i,j)} \min\{w_p, W(S') - k\} y_p \quad (24) \\
&\geq -\frac{U+1}{2}(W(S') - k) + \frac{U+3}{2}(W(S') - k) \\
&= W(S') - k. \quad (25)
\end{aligned}$$

Inequality (22) follows since $w_p \geq 2^i$ for each page $p \in S'(i,j)$, and Inequality (23) follows from (21). Inequality (24) follows since $w_p \leq 2^{i+1}$ for each page $p \in S'(i,j)$. Finally, Inequality (25) follows by the knapsack constraints:

$$\begin{aligned}
&\sum_{i=0}^{U} \sum_{j=0}^{C} \sum_{p \in S'(i,j)} \min\{w_p, W(S') - k\} y_p \\
&= \sum_{p \in S'} \min\{w_p, W(S') - k\} y_p \\
&= (U+3) \sum_{p \in S'} \min\{w_p, W(S') - k\} x_p \\
&\geq (U+3)(W(S') - k).
\end{aligned}$$

Here we use the fact that $y_p = (U+3)x_p$ for $p \in S'$. □

We now show how to maintain the bounded cost property with $\beta = 10$. For this we need to use both the LHS and RHS of condition (20), and we use an argument similar to the one used in the proof of Lemma 4.4.

LEMMA 4.8. *Give any distribution μ over balanced sets that is consistent with y. If y changes to y' incurring a*

fractional cost of d, then the distribution μ can be modified to another distribution μ' over balanced sets consistent with y' such that total cost incurred is at most $10d$.

PROOF. Suppose that y_p increases by ϵ and p lies in the class $S(i,j)$. Note that the balance condition (20) holds for every size class different from i, and moreover for size class i the condition also holds for all cost classes higher than j. We apply the procedure used in Lemma 4.4 to size class i. Note that applying this procedure does not have any effect on size classes different from i, and we can thus iteratively balance cost classes starting from j down to 0 in size class i. To bound the cost, observe that the analysis in the proof of Lemma 4.4 only used the fact that the cost of the classes are geometrically decreasing. Thus, a similar analysis implies that the cost incurred is no more than $10\epsilon \cdot 2^j$. □

We conclude with the next theorem:

THEOREM 4.9. *There is an $O(\log^2 k)$-competitive algorithm for the caching problem in the General model.*

4.3 The Fault Model

In this section we study the Fault model and show how to obtain an $O(\log k)$-competitive randomized caching algorithm for this model. Note that an $O(\log^2 k)$-competitive algorithm follows directly from the result for the General model. Recall that in the proofs for the Bit model and the General model we crucially used the fact that the cost in the different classes is geometrically decreasing. However, this is not the case for the Fault model, making the proof significantly more involved and requiring the use of a potential function so as to perform an amortized analysis.

We sort the n pages with respect to their size, i.e., $w_1 \leq w_2 \leq \ldots \leq w_n$. Let $x_1, \ldots, x_n$ be the LP solution at the current time step that satisfies the knapsack cover inequalities for all subsets. For each page p, let $y_p = \min\{1, 15 \cdot x_p\}$. Let S' denote the set of pages with $y_p < 1$. During the execution of the algorithm we maintain a grouping $\mathcal{G}$ of pages in S' into groups $G(i)$, $1 \leq i \leq \ell$. Each group $G(i)$ contains a sequence of consecutive pages in S'. As the pages are ordered in non-decreasing order with respect to size, for any i the largest page size in group $G(i)$ is at most the smallest page size in $G(i+1)$.

DEFINITION 4.10 (GOOD GROUPING). *A grouping $\mathcal{G}$ of pages in S' is called* good *if it satisfies the following properties.*

1. *For each i, $1 \leq i \leq \ell$, we have $\sum_{p \in S(i)} y_p \leq 12$.*
2. *If $\sum_{p \in S'} y_p \geq 3$, then for each group i, $1 \leq i \leq \ell$, we have $\sum_{p \in G(i)} y_p \geq 3$. If $\sum_{p \in S'} y_p < 3$, then there is exactly one group $G(1)$ containing all the pages in S'.*

We define $\sum_{p \in G(i)} y_p$ to be the *weight* of group $G(i)$.

DEFINITION 4.11 (BALANCED SET). *Given a good grouping G, a set of pages D is called balanced if the following two properties hold.*

1. *If $y_p = 1$, then $A_p^D = 1$.*
2. *For each i, the number of pages $|D \cap G(i)| = \sum_{p \in G(i)} A_p^D$ satisfies*

$$\left\lfloor \sum_{p \in G(i)} y_p \right\rfloor \leq \sum_{p \in G(i)} A_p^D \leq \left\lceil \sum_{p \in G(i)} y_p \right\rceil . \qquad (26)$$

The simulation procedure works as follows. At any time the algorithm maintains a good grouping $\mathcal{G}$ of the pages. It also maintains a probability distribution μ on balanced sets D which is consistent with y. At each step of the algorithm, as the value of y changes, the algorithm modifies the distribution μ to be consistent with y. Additionally, as y changes, the grouping $\mathcal{G}$ may also possibly change (so as to remain good), in which case a previously balanced set need not remain balanced anymore. In such a case, we also modify μ since only balanced sets can belong to the support of μ.

We first show that the size property holds for balanced sets D, and then show how to update $\mathcal{G}$ and μ as y changes, such that the cost property holds with $\beta = O(1)$ in an amortized sense.

LEMMA 4.12. *Let y be as defined above and let $\mathcal{G}$ be a good grouping with respect to y. Then any balanced set D with respect to $\mathcal{G}$ has size at least $W(S) - k$.*

PROOF. Let S' be the set of pages p for which $y_p < 1$. As D is balanced, each page with $y_p = 1$ belongs to D and hence it suffices to show that $\sum_{p \in S'} w_p A_p^D \geq W(S') - k$. If $W(S') - k \leq 0$, then we are already done. Henceforth we assume that $W(S') - k > 0$.

The linear program constraint for the set S' implies that $\sum_{p \in S'} \min\{w_p, W(S') - k\} x_p \geq W(S') - k$. This implies that $\sum_{p \in S'} x_p \geq 1$ and so $\sum_{p \in S'} y_p \geq 15$. Hence by the second condition for a good grouping, each group $G(i)$ has weight at least 3.

For each group $G(i)$ let $w_i(\min)$ and $w_i(\max)$ denote the smallest and largest page size in $G(i)$. Recall that for each i, we have that $w_i(\min) \leq w_i(\max) \leq w_{i+1}(\min)$. (Define $w_{\ell+1}(\min) = w_\ell(\max)$.) Let $m_i = \min(w_i(\min), W(S') - k)$ for $i = 1, \ldots, \ell + 1$. We lower bound the total size of pages in $D \cap S'$ as follows.

$$\begin{aligned}
\sum_{p \in S'} w_p A_p^D &\geq \sum_{p \in S'} \min\{w_p, W(S') - k\} A_p^D \\
&= \sum_{i=1}^{\ell} \sum_{p \in G(i)} \min\{w_p, W(S') - k\} A_p^D \\
&\geq \sum_{i=1}^{\ell} m_i \sum_{p \in G(i)} A_p^D \geq \sum_{i=1}^{\ell} m_i (-1 + \sum_{p \in G(i)} y_p) \\
&\geq \frac{2}{3} \sum_{i=1}^{\ell} m_i \sum_{p \in G(i)} y_p \qquad (27) \\
&= \frac{2}{3} \left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p \in G(i)} y_p \right) \\
&\quad - \frac{2}{3} \left(\sum_{i=1}^{\ell} (m_{i+1} - m_i) \sum_{p \in G(i)} y_p \right) \\
&\geq \frac{2}{3} \left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p \in G(i)} y_p \right) - 8 \left(\sum_{i=1}^{\ell} (m_{i+1} - m_i) \right) \qquad (28) \\
&= \frac{2}{3} \left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p \in G(i)} y_p \right) - 8 m_{\ell+1} + 8 m_1 \\
&\geq \frac{2}{3} \left(\sum_{i=1}^{\ell} \sum_{p \in G(i)} \min\{w_p, W(S') - k\} y_p \right) \\
&\quad - 8(W(S') - k) \qquad (29)
\end{aligned}$$

$$\geq \quad 2(W(S') - k).$$

Here inequality (27) follows since D is balanced, and hence for each $1 \leq i \leq \ell$,

$$\sum_{p \in G(i)} A_p^D \geq \left\lfloor \sum_{p \in G(i)} y_p \right\rfloor \geq -1 + \sum_{p \in G(i)} y_p,$$

and by observing that $\mathcal{G}$ is good and hence $\sum_{p \in G(i)} y_p \geq 3$ for each $1 \leq i \leq \ell$ and thus

$$-1 + \sum_{p \in G(i)} y_p \geq \frac{2}{3} \left(\sum_{p \in G(i)} y_p \right).$$

Inequality (28) follows since $m_{i+1} - m_i \geq 0$ for each $1 \leq i \leq \ell$, and since $\mathcal{G}$ is good, for each $1 \leq i \leq \ell$ we have that $\sum_{p \in G(i)} y_p \leq 12$: Finally, Inequality (29) follows by considering the knapsack cover inequality for the set S' and observing that $y_p = 15x_p$ for each $p \in S'$:

$$\begin{aligned} & \sum_{i=1}^{\ell} \sum_{p \in G(i)} \min\{w_p, W(S) - k\} y_p \\ = & \sum_{p \in S'} \min\{w_p, W(S') - k\} 15 x_p \geq 15(W(S') - k). \end{aligned}$$

□

LEMMA 4.13. *As the solution y changes over time we can maintain a good grouping $\mathcal{G}$ and a consistent distribution on balanced sets with amortized cost at most a constant times the fractional cost.*

PROOF. The online fractional algorithm has the following dynamics. After a page p is requested variable y_p can only increase (the page is gradually evicted). This process stops when page p is requested again and y_p is set to zero. Whenever y_p changes, we need to modify the distribution μ on balanced sets D to remain consistent. Moreover, a change in y_p may change the structure of the groups. This happens if either the weight of $G(i)$ exceeds 12, or if it falls below 3, or if y_p becomes 1 and leaves the group $G(i)$ (recall that groups only contain pages q with $y_q < 1$). We view a change in y_p as a sequence of steps where y_p changes by an infinitesimally small amount ϵ. Thus at each step exactly one of the following events happens.

Event 1: Variable $y_p < 1$ of page p increases or decreases by ϵ.

Event 2: The weight of group $G(i)$ reaches 12 units.

Event 3: The weight of group $G(i)$ drops to 3 units.

Event 4: The value of y_p for page p reaches 1 and p leaves the set $S(i)$.

We prove that in all cases the amortized cost of the online algorithm is at most $O(1)$ times the fractional cost. For amortization we use the following potential function:

$$\Phi = 13 \sum_{p \in S'} y_p + 11 \sum_{i=1}^{\ell} \left| 6 - \sum_{p \in G(i)} y_p \right|.$$

In each possible event let C_{on} be the total cost of the online algorithm. Let C_f be the fractional cost, and let $\Delta\Phi$ be the change in the potential function. We show that in each of the events:

$$\Delta C_{on} + \Delta\Phi \leq 405 \Delta C_f \tag{30}$$

Since Φ is always positive, this will imply the desired result.

Event 1.

Assume first that y_p such that $p \in G(i)$ is increased by ϵ. If y_p increases by ϵ it must be that x_p is increased by at least $\frac{\epsilon}{15}$. Thus, in the fault model the fractional cost is at least $\frac{\epsilon}{15}$.

To maintain consistency, we add p to ϵ measure of the sets D that do not contain p. However this might make some of these sets unbalanced by violating (26). Suppose first that $s = \lceil \sum_{p \in G(i)} y_p \rceil$ does not change when y_p is increased by ϵ. In this case, we match the sets with $s+1$ pages in $G(i)$ (the measure of these is at most ϵ) arbitrarily with sets contains $s-1$ pages, and transfer some page from the larger set (that does not lie in the smaller set) to the smaller set. An analogous argument works when s increases as y_p is increased. Note that after this step, the sets become balanced.

The total online cost is 3ϵ. Moreover, the potential change $\Delta\Phi$ is at most $13\epsilon + 11\epsilon = 24\epsilon$ and hence (30) holds. An analogous argument works if y_p is decreased (in fact it is even easier since the potential only decreases).

Event 2.

Consider an event in which the total weight of a group $G(i)$ reaches 12 units. In this case we split $G(i)$ into two sets such that their weight is as close to 6 as possible. Suppose one set is of size $6 + x$ and the other is of size $6 - x$ where $0 \leq x \leq 1/2$. Let $\Phi(s)$ and $\Phi(e)$ denote the potential function before and after the change respectively. The contribution of the first term does not change. The second term corresponding to $G(i)$ initially is at least $11(12 - 6) = 66$ and the final contribution is $11(|6 - (6 - x)| + |6 - (6 + x)|) = 22x \leq 11$. Thus $\Delta\Phi = \Phi(e) - \Phi(s) = 11 - 66 \leq -55$.

Next, we redistribute the pages in the original group $G(i)$ among the sets D such that they are balanced with respect to the two new groups. Observe that in the worst case, each set D might need to remove all the 12 pages it previously had and bring in at most $\lceil 6 + x \rceil + \lceil 6 - x \rceil \leq 13$ new pages. Since the measure of sets is D, the total cost incurred is at most 25. Again, (30) holds as the fractional cost C_f is 0 and the decrease in potential more than offsets the cost C_{on}.

Event 3.

Consider the event when the weight of a group $G(i)$ decreases to 3 units. If $G(i)$ is the only group (i.e. $\ell = 1$) then all properties of a good grouping still hold. Otherwise, we merge $G(i)$ with one of its neighbors (either $G(i-1)$ or $G(i+1)$). If $G(i)$ has a neighbor with weight at most 9, then we merge $G(i)$ with this neighbor. Note that before the merge, each balanced set D has exactly 3 pages from $G(i)$ and hence it also remains balanced after the merge. Also, since $|6 - 3| + |6 - x| \geq |6 - (x + 3)|$ for all $3 \leq x \leq 9$, and hence the potential function does not increase in this case. Thus (30) holds trivially.

Now suppose that all neighbors of $G(i)$ have weight greater 9. Consider any such neighbor and let $x > 9$ be its weight. We merge $G(i)$ with this neighbor to obtain a group with weight $3 + x$ which lies in the range $(12, 15]$. Then as in the

handling of Event 4.3, we split this group into two groups with as close weight as possible. Since the weight is at most 15, the cost of balancing the sets D is at most $16 + 15 = 31$ (using argument similar to that in Event 4.3). We now consider the change in potential. The only change is due to second terms corresponding to $G(i)$ and its neighbor (the first term does not matter since total weight of pages in S' does not change upon merging or splitting). Before the merge, the contribution was $11 \cdot 3 + 11 \cdot (x-6) = 11x - 33 \geq 66$. After the merge (and the split) the maximum value of the potential is obtained for the case when the size of the merged group is 15 which upon splitting leads to sets of size $7 + y$ and $8 - y$ where $y \leq 0.5$, in which case its value is $11(1+y+2-y) = 33$. Thus, the potential function decreases by at least 33 while the online cost is at most 31, and hence (30) holds.

Event 4.

Suppose some y_p increases to 1 and exits the group $G(i)$. Note that if $y_p = 1$, then all balanced sets D contain p. Thus, removing p from $G(i)$ keeps the sets balanced.

Let us first assume that the weight of $G(i)$ does not fall below 3 when p is removed. In this case, the groups and the balanced sets remain unchanged. Thus the online algorithm incurs zero cost. The first term of the potential decreases by 13, and the second term increases by at most 11, and hence (30) holds. Now consider the case when the weight of $G(i)$ falls below 3. We apply an argument similar to that for Event 4.3. If $G(i)$ can be merged with some neighbor without weight exceeding 12, then we do so. This merge may cause some sets D to become imbalanced. However, this imbalance is no more than one page and can be fixed by transferring one page from each set to another appropriate set. The total cost incurred in this case is at most 2. We now consider the change in potential. The first term decreases by 13. For the second term, the original group $G(i)$ contributes function $11(6 - (3 + x)) = 11(3 - x)$, with $x < 1$ and its neighbor contributes $11(|6 - z|)$ where $3 \leq z \leq 9$ is its weight. After the merge, the second term corresponding to the merged group contributes $11(|6 - (z + 2 + x)|)$ which is at most $11(|6 - z| + (2 + x))$. Overall, $\Delta\Phi \leq -13 + 11(2 + x) - 11(3 - x) = 22x - 24 < -2$. Thus (30) holds.

If we need to split the merged set, we note that the above analysis, showing that (30) holds, is also valid when $9 \leq z \leq 12$. Next, when this merged set is split, we can apply the analysis in Event 4.3, and then the potential function decreases by at least 33 units, while the cost incurred is at most 31, and hence (30) holds. □

We conclude with the next theorem:

THEOREM 4.14. *There is an $O(\log k)$-competitive algorithm for the caching problem in the Fault model.*

5. REFERENCES

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

[2] S. Albers, S. Arora, and S. Khanna. Page replacement for general caching problems. In *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 31–40, 1999.

[3] N. Bansal, N. Buchbinder, and J. Naor. A primal-dual randomized algorithm for weighted paging. In *Proc. of the 48th annual IEEE Symposium on Foundations of Computer Science*, pages 507-517, 2007.

[4] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

[5] Reuven Bar-Yehuda and Dror Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM J. Discrete Math.*, 19(3):762–797, 2005.

[6] A. Borodin and R. El-Yaniv. Online computation and competitive analysis. *Cambridge University Press*, 1998.

[7] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proceedings of the 13th Annual European Symposium on Algorithms*, pages 689–701, 2005.

[8] P. Cao and S. Irani. Cost-aware www proxy caching algorithms. In *USENIX Symposium on Internet Technologies and Systems*, pages 193–206, 1997.

[9] R. Carr, L. Fleischer, V. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Symposium on Discrete Algorithms*, pages 106–115, 2000.

[10] M. Chrobak, H. J. Karloff, T. H. Payne, and S. Vishwanathan. New results on server problems. *SIAM J. Discrete Math*, 4(2):172–181, 1991.

[11] E. Cohen and H. Kaplan. LP-based analysis of greedy-dual-size. In *Proceedings of the 10th Annual ACM-SIAM symposium on Discrete algorithms*, pages 879–880, 1999.

[12] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.

[13] S. Irani. Competitive analysis of paging: a survey. In *Proceedings of the Workshop on Online Algorithms, Dagstuhl, Germany*. Springer Verlag Lecture Notes in Computer Science.

[14] S. Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings of the 29th Annual ACM Symposium on Theory of computing*, pages 701–710, 1997.

[15] S. Irani. Randomized weighted caching with two page weights. *Algorithmica*, 32(4):624–640, 2002.

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

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

[18] N. E. Young. On-line caching as cache size varies. In *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–250, 1991.

[19] N. E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.

[20] N. E. Young. On-line file caching. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 82–86, 1998.