Randomized Competitive Algorithms for Generalized Caching[*]

Nikhil Bansal [†] Niv Buchbinder [‡] Joseph (Seffi) Naor [§]

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. Our main result is an online algorithm with competitive ratio $O(\log^2 k)$, which gives the first $o(k)$ competitive algorithm for the problem. We also give improved $O(\log k)$-competitive algorithms for the special cases of the bit model and fault model, improving upon the previous $O(\log^2 k)$ guarantees due to Irani [17]. Our algorithms are based on an extension of the online primal-dual framework introduced by Buchbinder and Naor [8], and are based on two steps. First, we obtain an $O(\log k)$-competitive fractional algorithm based on solving online a LP formulation strengthened with exponentially many knapsack-cover constraints. Second, we design a suitable online rounding procedure to convert this online fractional algorithm into a randomized algorithm. Our techniques provide a unified framework for caching algorithms and are substantially simpler than those previously used.

1 Introduction

Caching is one of the earliest and most effective techniques of accelerating the performance of computing systems. Thus, vast amounts of effort 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 basic caching 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 correspond to web pages that could have possibly different 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

[*]A preliminary version of this paper appeared in the 40th Annual ACM Symposium on Theory of Computing, 2008

[†]Eindhoven University of Technology, The Netherlands. E-mail: *n.bansal@tue.nl*

[‡]Computer Science Department, Open University of Israel. Supported by ISF grant 954/11 and BSF grant 2010426. E-mail: *niv.buchbinder@gmail.com*

[§]Computer Science Department, Technion, Haifa 32000, Israel. Supported by ISF grant 954/11 and BSF grant 2010426. E-mail: *naor@cs.technion.ac.il* some of the pages locally, and also by internet providers that maintain proxy servers for exactly the same purpose.

The generalized caching problem models these two extensions and is formally defined as follows. There is a cache of total size k and there are n pages $1, \ldots, n$ which are requested (possibly repeatedly) over time. Each page p is associated with two parameters, w_p and c_p, denoting size and fetching cost, respectively. We assume that $w_p \in [1, k]$ for all p. Note that w_p is not necessarily an integer and that the cache size k can be viewed as the ratio between the cache size and the smallest page size. At each time step t, if the currently requested page p_t is already present in the cache, then the page is served with zero cost. Otherwise, if p_t is not in the cache, then it must be fetched into the cache, possibly evicting other pages to make room for it, and a cost equal to the fetching cost of p_t is incurred. The goal of a paging algorithm is to minimize the total cost over the entire request sequence. In the online setting, the algorithm only knows the pages requested thus far, and hence the main question in designing any online algorithm for caching is determining which pages to evict to make room for the currently requested page, in the absence of knowledge of future requests.

We study here several models of the generalized caching problem from a competitive analysis point of view. In the most general setting, called the *general model*, pages have both non-uniform sizes and costs, i.e. both w_p and c_p are arbitrary. 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, i.e. $c_p = w_p$, and 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 irrespective of their size, i.e. $c_p = 1$ and w_p is arbitrary, 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 in which both size and cost are uniform is very well understood. In their seminal paper, Sleator and Tarjan [19] showed that any deterministic algorithm is at least k-competitive, and also showed that the LRU (least recently used) policy is exactly k-competitive. When randomization is allowed, Fiat et al. [15] 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 [18, 1]. More generally, the (h, k)-version of the problem has also been studied. Here, the online algorithm with cache size k is compared against an offline algorithm with a cache size of h, where $h \leq k$. Here, a tight $k/(k-h+1)$-competitive algorithm is known for the deterministic case [19], and a $2\ln(k/(k-h+1))$-competitive algorithm is also known for the randomized case [20].

For weighted caching (uniform page size, but non-uniform fetching cost), a tight k-competitive deterministic algorithm follows from the more general result of Chrobak et al. [12] for the k-server problem on trees. Subsequently, Young [21] gave a tight $k/(k-h+1)$-competitive deterministic algorithm for the more general (h, k)-caching problem. The randomized case was settled relatively recently by Bansal et al. [4], who designed an $O(\log k)$-competitive algorithm for the problem, and more generally an $O(\ln(k/(k-h+1)))$-competitive algorithm for the (h, k)-version, via the online primal-dual method.

Caching becomes substantially harder when page sizes are non-uniform. Already in the offline case, the generalized caching problem becomes strongly NP-Hard even for the special cases of the bit and the fault models [13]. In contrast, the offline weighted caching problem can be solved in polynomial time using minimum cost network flow. Following a sequence of results

[17, 3, 14], Bar-Noy et al. [5] gave a 4-approximation for the problem based on the local-ratio technique. In fact, the result of Bar-Noy et al. [5] holds in the more general setting where the available cache space may vary with time. This is currently the best known approximation for the offline generalized caching problem. An alternate $O(1)$ approximation using the framework of *knapsack cover inequalities* (which is also what we will use in this paper) can be found in [11]. They refer to this problem as the priority line cover problem. We note that the local ratio algorithm of [5] for generalized caching, coupled with the techniques of [6], also yields a set of knapsack cover inequalities defining a linear program with bounded integrality gap (at most 4).

For the online case it is known that LRU is $(k+1)$-competitive for the bit model and also for the fault model [17]. (Recall that k denotes the ratio between the cache size and the size of the smallest page.) Later on, Cao and Irani [9] and Young [22] gave a $(k+1)$-competitive algorithm for the general model based on a generalization of the greedy-dual algorithm of Young [21]. An alternate proof of this result was obtained by [14]. When randomization is allowed, Irani [17] designed an $O(\log^2 k)$-competitive algorithm for both fault and bit models. For the general model, no $o(k)$ randomized algorithms were known (until our work) and designing such algorithms had been an important open question.

We remark that in general, there has been extensive work on various other variants of caching. We refer the reader to the excellent text by Borodin and El-Yaniv [7] and to the survey by Irani [16] on caching for further details.

1.2 Results and techniques

In this paper, we consider all three models mentioned above and prove the following result:

Theorem 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.*

Recall that any randomized algorithm can be viewed as a probability distribution over deterministic algorithms. This is also the view that we will adopt in designing our randomized algorithms above. In particular, at each time step, we will explicitly specify the distribution on the deterministic states that the algorithm maintains. To do this, for each time step t and each real number $\eta \in [0,1)$, we specify a set $S(\eta, t)$ of pages such that: (i) the total size of the pages in $S(\eta, t)$ is at most k and (ii) the currently requested page p_t is present in $S(\eta, t)$ for all $\eta \in [0,1)$. Upon arrival of a new request at time $t+1$, the sets $S(\eta, t+1)$ can possibly change and the cost incurred by the algorithm at this time step is the expected cost

$$\mathbb{E}_\eta\Big[C(S(\eta, t+1) \setminus S(\eta, t))\Big], \tag{1}$$

where $C(X)$, for a subset of pages X, denotes the total fetching cost of the pages in X.

Now, given this explicit description, consider the randomized algorithm that picks a random number $r \in [0,1)$, once and for all, and sets its cache state at each time t to $S(r,t)$. Clearly, the expected cost (by averaging over r) of this randomized algorithm is exactly equal to the cost given by (1). We remark that our algorithm does not necessarily run in polynomial time (this is not an issue for online algorithms) and we shall not concern ourselves with this aspect here.

Instead of working directly with distributions over deterministic states, it is much simpler to break this down into two steps:

- *Maintaining a fractional solution:* Instead of specifying an explicit distribution on cache states, it is much more convenient to work with a probability distribution on pages in the cache at each time step, i.e. the probability of page p being present in the cache at time t, denoted by x_p^t. We call this the *fractional* view, and first design a *fractional* algorithm in this setting. The fractional algorithm is allowed to keep fractions of pages as long as the total (fractional) size of the pages present in the cache does not exceed the cache size k. The cost of fetching an ε fraction of a page is then defined to be ε times the fetching cost of the whole page.

- *Online rounding of the fractional solution:* While every probability distribution of cache states induces a probability distribution on the pages, the reverse is not true (i.e. there are probability distribution on pages that cannot be extended to a valid distribution on cache states). So, in the second step, we map the above fractional algorithm into an algorithm that works with a distribution over cache states. To this end, we maintain online a distribution over cache states such that (i) it is (almost) consistent with the fractional solution in hand, and (ii) show how to map the changes in the fractional distribution over pages (at every time step t) to changes in the distribution of caches, so that the cost incurred is not much higher than the fractional cost. Note that this mapping is maintained in an online manner.

For a problem at hand, if one can design an α-competitive fractional algorithm, and lose another β factor in the online rounding process, by the discussion above this gives an $\alpha\beta$ competitive randomized algorithm for the problem.

To prove theorem 1, we design an $O(\log k)$ fractional algorithm for generalized caching, and lose an additional $O(\log k)$ factor in the online rounding, implying an $O(\log^2 k)$ competitive algorithm. In the special cases of the bit model and the fault model however, we give an improved rounding procedure that only loses a factor of $O(1)$, implying the $O(\log k)$ competitive algorithm for these cases.

The Fractional Online Algorithm. The fractional algorithm we propose is based on the online primal-dual framework developed by [8] for online packing and covering problems. Roughly speaking, the framework says that if the offline problem can be cast as a covering linear program (possibly with box constraints of the type $x \leq 1$ for variable x), then there is an $O(\log n)$ competitive fractional algorithm for the online version of the problem, where the constraints arrive online over time, and the value of the variables is monotonically non-decreasing over time. Here, n is the maximum number of variables in any constraint. In the context of caching, [4] showed how this guarantee could be refined from $O(\log n)$ to $O(\log k)$.

We use a similar approach. However, as we discuss later on in Section 2, dealing with non-uniform sizes requires additional ideas. Specifically, the natural LP relaxation (where we only require that the total fractional size of the pages be no more than k) for the generalized caching problem is too weak to give a good randomized algorithm upon rounding, and thus we need to strengthen it by adding exponentially many knapsack cover inequalities. We then show how to apply the online primal-dual technique to this strengthened LP to derive an $O(\log k)$-competitive fractional algorithm for generalized caching.

Online Rounding. The online rounding poses many challenges, and it is probably the most technical part of this paper. Not only do we need to crucially use the strength of knapsack cover inequalities to obtain a good integral solution, but we also need to maintain this rounding in an online manner as the fractional solution changes over time, while ensuring that the cost incurred is competitive with the cost of the fractional algorithm. This online rounding is model-dependent and is different for the bit, fault, and general models. In all these cases, the rounding involves maintaining a distribution over cache states that closely mimics the fractional solution, together with several additional properties that are needed to ensure that the solution can be updated in an online manner. The latter issue is non-trivial, even when the pages have uniform size (we refer the interested reader to a simple example in [4] that shows the need for such additional properties).

2 Preliminaries

Let us recall the notation before we begin. There is a cache of size k and there are n pages with sizes $w_1, w_2, \ldots, w_n$, such that $w_p \in [1, k]$ for all p. Throughout the paper, for a subset of pages S, we use $w(S) = \sum_{i \in S} w_i$ to denote the total size of pages in S. Each page p has a fetching cost of c_p, which we assume (by normalizing if needed) is at least 1, i.e. $c_p \geq 1$ for all p. With this terminology, we have the following: for each page p, $c_p = 1$ in the fault model, $c_p = w_p$ in the bit model, and c_p and w_p are arbitrary in the general model. At each time $t = 1, \ldots, T$, where T is the length of the input sequence, let p_t denote the page requested at t. If p_t is not in the cache, the algorithm must fetch it to the cache (possibly evicting other pages to make room for p) and incurring a cost of c_p. Note that evicting a page does not incur any cost.

Cost Accounting. A simple observation is that instead of charging for fetching a page, we can charge for evicting it from the cache. Clearly, over the entire time horizon of the request sequence, the number of times any page is fetched can differ from the number of times it is evicted by at most 1. Thus, the overall costs for fetching versus evicting can differ by at most $k \cdot w_{\max}$, which is an additive constant independent of the length of the request sequence, and hence does not affect the competitive ratio.

2.1 LP formulation for the generalized caching problem.

While it is perhaps more natural to view caching as a packing problem (as the pages must be packed into a cache of size k), it can also be viewed as a covering problem in the following way. For each page p, and integers $j = 1, \ldots,$ let $t(p, j)$ denote the time of the jth request. Consider any two consecutive requests for page p at times $t(p, j)$ and $t(p, j + 1)$. Since p must be present in the cache at both of these times, the algorithm pays for fetching p at time $t(p, j + 1)$ if and only if p was evicted during the time interval $[t(p, j) + 1, t(p, j + 1) - 1]$. Now, in the offline solution, if p is evicted in this interval, then we can assume without loss of generality that it is evicted at time $t(p, j) + 1$ (clearly, this assumption is not necessarily true in the online case).

The above motivates the following covering integer program. Let $x(p, j)$ be an indicator variable for the event that page p is evicted from the cache during $[t(p, j) + 1, t(p, j + 1) - 1]$ (and hence at $t(p, j) + 1$). For each page p, let $r(p, t)$ denote the number of requests for page p 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 ever requested until time t (including t).

The Natural IP Formulation. We need to satisfy that at any time t, the page p_t must be present in the cache, and that the total space used by pages in the cache is at most k. As 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 $B(t) \setminus \{p_t\}$ at time t, i.e.

$$\sum_{p \in B(t) \setminus \{p_t\}} w_p \left(1 - x(p, r(p,t))\right) \leq k - w_{p_t},$$

which upon rearrangement, gives us that

$$\sum_{p \in B(t) \setminus \{p_t\}} w_p \cdot x(p, r(p,t)) \geq w(B(t)) - k. \tag{2}$$

This gives the following exact covering formulation of the offline generalized caching problem (where we charge for evicting a page instead of fetching).

$$\begin{array}{rcll} \min \displaystyle\sum_{p=1}^{n} \sum_{j=1}^{r(p,T)} c_p \cdot x(p,j) & & & \\ \displaystyle\sum_{p \in B(t) \setminus \{p_t\}} w_p \cdot x(p, r(p,t)) & \geq & w(B(t)) - k & \forall t \in \{1, \ldots, T\} \\ x(p,j) & \in & \{0,1\} & \forall \; p, j \end{array}$$

Integrality Gap of the Natural LP Relaxation. The natural LP relaxation is obtained by relaxing the variables $x(p,j)$ in the above IP to take values between 0 and 1. It turns out that this LP relaxation has an arbitrarily large gap and is therefore not suitable for our purposes.

The following example is instructive. Suppose the cache size is $k = 2\ell - \varepsilon$, for some $\varepsilon > 0$, and there are two identical pages, each of size $w_p = \ell$ and cost $c_p = 1$. Consider the request sequence in which the two pages are requested alternately. As any integral solution can keep only one page in the cache at any time, it incurs a cache miss at each request, implying a total cost of $\Omega(T)$. The LP solution on the other hand can keep one unit of the currently requested page, and $1 - \varepsilon/\ell$ units of the other page, as $\ell \cdot 1 + \ell(1 - \varepsilon/\ell) = 2\ell - \varepsilon = k$. Hence, it only needs to fetch an ε/ℓ fraction of a page at each step, incurring a total cost of $O(\varepsilon T/\ell)$. Choosing ε arbitrarily smaller than ℓ, implies an arbitrarily large integrality gap.

Knapsack Cover Inequalities. To get around the integrality gap, we strengthen the above formulation by adding exponentially many *knapsack cover* (KC) inequalities, as in [10]. These constraints are redundant in the integer program, but they dramatically reduce the integrality gap of the LP relaxation.

The KC inequalities are based on two new ideas. First, instead of the single constraints (2), one adds exponentially many such constraints. Consider a subset of pages $S \subset B(t)$ such that $p_t \in S$ and $w(S) \geq k$. As the cache has size k and p_t must be present at time t, the pages in $S \setminus \{p_t\}$ can occupy at most $k - w_{p_t}$ size in the cache at time t. Thus, for any such S and time t the following is a valid constraint: $\sum_{p \in S \setminus \{p_t\}} w_p \cdot (1 - x(p, r(p,t))) \leq k - w_p$, which can be rewritten as

$$\sum_{p \in S \setminus \{p_t\}} w_p \cdot x(p, r(p,t)) \geq w(S) - k. \tag{3}$$

The second idea is that for each such constraint, one can truncate the size of a page to be equal to the right hand size of the constraint, i.e. replace (3) by

$$\sum_{p \in S \setminus \{p_t\}} \min(w(S) - k, w_p) \cdot x(p, r(p,t)) \geq w(S) - k, \tag{4}$$

without affecting the space of feasible integral solutions (while significantly strengthening the LP relaxation). To see this, first note that as $\min(w(S)-k, w_p) \le w_p$, any solution satisfying (4) also satisfies (3). Conversely, we claim that any 0-1 solution satisfying (3) also satisfies (4). To see this, suppose x satisfies (3) for some set S. Now, if $x(p, r(p,t)) = 0$ for some p, then clearly this has no effect in both (3) and (4). So we assume that $x(p, r(p,t)) = 1$. Now, the quantity $\min(w(S)-k, w_p)$ differs from w_p only if $w_p > w(S) - k$, but then as $x(p, r(p,t)) = 1$ and $w_p > w(S) - k$, (4) is also satisfied.

Strengthened Formulation. We can thus define the following strengthened LP relaxation to the problem.

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

For every time t, and all subsets $S \subseteq B(t)$, such that $p_t \in S$ and $w(S) > k$:

$$\sum_{p \in S \setminus \{p_t\}} \min\{w(S)-k, w_p\} \cdot x(p, r(p,t)) \quad \ge \quad w(S) - k \tag{5}$$

$$x(p,j) \quad \in \quad [0,1] \qquad \forall \;\; p, j \tag{6}$$

It is instructive to see how this strengthened LP gets rid of the large integrality gap instance for the natural LP relaxation previously discussed. In that instance, consider constraint (5) for $S = \{1,2\}$. Then, $w(S) - k = 2\ell - k = \varepsilon$, and hence this constraint becomes $\varepsilon x(p, r(p,t)) \ge \varepsilon$ (where p is the page other than p_t). In particular, this captures the fact that p must be evicted completely at time t (as opposed to an ε extent in the natural LP) to make room for p_t.

A Further Simplification. We note another observation about KC inequalities that will allow us to further simplify the above LP by removing the box constraints $x(p,j) \le 1$.

Observation 2. *Given a fractional solution x, if a KC inequality (5) is violated for a set S at time t, then it is also violated for the set $S' = S \setminus \{p : x(p, r(p,t)) \ge 1\}$.*

Proof. Consider some fractional solution x, and suppose that inequality (5) is violated for some S with some $p \in S$ for which $x(p, r(p,t)) \ge 1$. Now, it must necessarily be the case that $\min(w(S)-k, w_p) < w(S)-k$, as the inequality for S would trivially be satisfied if $w_p \ge w(S)-k$ and $x(p, r(p,t)) \ge 1$. Suppose we delete p from S. The right hand size of (4)) decreases by exactly w_p. The left hand side decreases by at least $x(p, r(p,t)) \cdot w_p \ge w_p$ (the decrease could be possibly more, as the term $\min(w(S)-k, w_{p'})$ may also decrease for other pages $p' \in S$, due to the removal of p from S). Thus, (5) is also violated for $S \setminus \{p\}$. The result follows by repeatedly applying the argument. □

Observation 2 implies that in any feasible solution to the constraints given by (5), there is no advantage to having $x(i,j) > 1$. Thus, we can drop the upper bounds on $x(i,j)$ and simplify the formulation to obtain the following covering LP:

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

For every time t, and all subsets $S \subseteq B(t)$, such that $p_t \in S$ and $w(S) > k$:

$$\sum_{p \in S \setminus \{p_t\}} \min\{w(S) - k, w_p\} \cdot x(p, r(p,t)) \geq w(S) - k \tag{7}$$

$$x(p,j) \geq 0 \qquad \forall \ p, j \tag{8}$$

The Dual Program. We now consider the dual of the LP above, which will play a key role in our algorithm. 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:

$$\max \quad \sum_{t=1}^{T} \sum_{S \subseteq B(t), p_t \in S} (w(S) - k) \cdot y(t,S)$$

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

$$\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \ | \ p \in S} \min\{w(S) - k, w_p\} \cdot y(t,S) \leq c_p \tag{9}$$

$$y(t,S) \geq 0 \tag{10}$$

3 Computing a Competitive Fractional Solution

In this section we describe a fractional online caching algorithm that computes a solution to LP-Caching. In the online case, the constraints of LP-Caching corresponding to time t are only revealed at time t. At any step, the online algorithm is required to maintain a feasible (primal) solution subject to the constrains seen thus far. Moreover, these variables can only be increased over time (this models the online nature of the problem, that previous decisions cannot be revoked). The increase in these variables is guided by the dual program.

In the LP above, at any time t, exponentially many new linear knapsack-cover constraints are revealed to the online algorithm. Since there are exponentially many constraints, this process may not run in polynomial-time. For ease of exposition, we will ignore this issue when we first describe the algorithm. Later, we will show how the primal-dual nature of the algorithm can be used to make the running time polynomial.

Informal Algorithm Description. We start with a high level description of the algorithm. Upon arrival of the new constraints at time t, if they are all already satisfied, then the algorithm does nothing. Otherwise, the algorithm needs to satisfy all the current constraints by increasing some of the primal variables (this corresponds to evicting other pages gradually to make room for the page p_t). Let us call a set S minimal if $x(p, r(p,t)) < 1$ for each $p \in S$. By Observation 2, it suffices to consider the constraints corresponding to minimal sets. To this end, the algorithm arbitrarily picks an unsatisfied primal constraint for a minimal set S and starts increasing the corresponding dual variable $y(t,S)$ continuously. The variables $x(p,j)$ increase according to an exponential function of the variable $y(t,S)$. When a variable $x(p,j)$ reaches 1, the set S is no longer minimal, and 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. While this algorithm is stated in a continuous manner, it can be easily implemented in a discrete manner.

Formal Algorithm Description. The algorithm is as follows.

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** primal constraints corresponding to time t are satisfied, do the following:
- Pick some minimal set S for which the primal constraint is unsatisfied.

1. Increase variable $y(t, S)$ continuously.
2. For each $p \in S \setminus \{p_t\}$, increase the variable $x(p, r(p, t))$ at rate
$$\frac{dx(p, r(p,t))}{dy(t,S)} = \frac{1}{c_p} \cdot \ln(k+1) \cdot \min\{w(S) - k, w_p\} \cdot \left(x(p, r(p,t)) + \frac{1}{k}\right) \quad (11)$$
3. If $x(p, r(p, t)) = 1$, then remove p from S, i.e. $S \leftarrow S \setminus \{p\}$.

We show that:

Theorem 3. *The fractional caching algorithm described above is $O(\log k)$-competitive.*

We first note the following property about the variables $x(p, j)$.

Observation 4. *For every p and j, the value of $x(p, j)$ at the end of the algorithm (i.e. at time T) is given by*

$$x(p,j) = \frac{1}{k} \cdot \left(\exp\left(\frac{\ln(k+1)}{c_p} \left(\sum_{t'=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \mid p \in S} \min\{w(S') - k, w_p\} \cdot y(t', S) \right) \right) - 1 \right).$$

Proof. First, we note that $x(p, j)$ is only updated at times $t(p, j) + 1, \ldots, t(p, j + 1) - 1$, and it increases according to (11) whenever some dual variable $y(t, S)$ such that $p \in S$ and $t \in [t(p, j) + 1, \ldots, t(p, j + 1) - 1]$ is increased.

Now, consider the differential equation $dz/dy = c(z + 1/k)$. If y increases from a to b, we have that
$$\ln(z(b) + 1/k) - \ln(z(a) + 1/k) = c(b - a). \quad (12)$$
Now, let us consider the dynamics of $x(p, j)$. It starts at 0 at time $t(p, j)+1$, increases according to (11). As each $y(t, S)$ is only increased at time t and its value starts at 0, by applying (12) at each time step, this implies that at the end of time $t(p, j + 1) - 1$:

$$\ln(x(p,j) + 1/k) - \ln(1/k) = \sum_{t'=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \mid p \in S} \frac{1}{c_p} \cdot \ln(k+1) \cdot \min\{w(S) - k, w_p\} \cdot y(t', S).$$

Now, taking exponents on both sides of this equation gives the claimed result. □

We now prove theorem 3.

Proof. (Theorem 3) It suffices to show the following. First, both primal and dual solutions are feasible, and second, the primal cost is at most $O(\log k)$ times the dual cost. As the value of any feasible dual solution lower bounds the optimum LP value, the result is implied.

Feasibility: The primal solution generated by the algorithm is feasible by design 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 beyond 1, since whenever $x(p,j)$ reaches 1, any set S containing p is not minimal any more, and hence the corresponding variables $y(t,S)$, for which $p \in S$, do not increase anymore.

To see that the dual is not violated, note that as $x(p,j) \le 1$, Observation 4 implies that

$$\frac{1}{k} \cdot \left(\exp\left(\frac{\ln(k+1)}{c_p} \left(\sum_{t'=t(p,j)+1}^{t(p,j+1)-1} \sum_{S \ \mid \ p \in S} \min\{w(S') - k, w_p\} y(t', S) \right) \right) - 1 \right) \le 1,$$

which upon rearranging and taking logarithms equivalently implies that

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

and hence that the dual constraint corresponding to the variable $x(p,j)$ is never violated.

Bounding Primal and Dual Costs: Let us consider an infinitesimal step where variable $y(t,S)$ increases by $dy(t,S)$. Looking at the dual objective function, it follows that the dual profit obtained during this infinitesimal step is

$$dY = (w(S) - k) \cdot dy(t,S).$$

We will show that the primal cost incurred during this infinitesimal step is at most

$$2\ln(k+1)(w(S) - k) \cdot dy(t,S) = 2\ln(k+1) \cdot dY.$$

As the eviction cost for page p is c_p, the primal cost incurred during this infinitesimal step can be written as

$$\begin{aligned} & \sum_p c_p \cdot dx(p, r(p,t)) \\ = \ & \sum_p c_p \cdot \frac{dx(p, r(p,t))}{dy(t,S)} \cdot dy(t,S) \\ = \ & \sum_{p \in S \setminus \{p_t\}} \ln(k+1) \cdot \min\{w(S) - k, w_p\} \cdot \left(x(p, r(p,t)) + \frac{1}{k} \right) \cdot dy(t,S), \end{aligned} \tag{13}$$

where the last step follows from the update rule of our algorithm given by (11).

We bound (13) as follows. First, as $y(t,S)$ is being raised, it must be (according to the algorithm) that the primal constraint corresponding to the variable $y(t,S)$ is still unsatisfied, and hence

$$\sum_{p \in S \setminus \{p_t\}} \min\{w(S) - k, w_p\} \cdot x(p, r(p,t)) \le w(S) - k. \tag{14}$$

Second, we claim that

$$\sum_{p \in S \setminus \{p_t\}} \min\{w(S) - k, w_p\} \cdot \frac{1}{k} \le w(S) - k. \tag{15}$$

To see why this holds, consider the following two cases, depending on whether $w(S) \ge k+1$, or not.

- If $w(S) \geq k+1$, then:

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

  In the second step, the first inequality follows as $w(p_t) \geq 1$ (the minimum page size is 1) and the second inequality follows as the inequality $(x-1)/k \leq x-k$ holds for every real number $x \geq k+1$ (and $w(S) \geq k+1$).

- If $w(S) < k+1$, then the set S contains at most k pages (since the minimum page size is 1). In this case we get that:

$$\begin{aligned}\sum_{p \in S\setminus\{p_t\}} \frac{1}{k} \cdot \min\{w(S)-k, w_p\} &\leq \frac{1}{k} \sum_{p \in S\setminus\{p_t\}} (w(S)-k) \\ &\leq \frac{k-1}{k} \cdot (w(S)-k) < w(S) - k.\end{aligned}$$

Now, plugging the bounds from (14) and (15) into (13), it follows that the primal cost is at most

$$2\ln(k+1) \cdot (w(S)-k) \cdot y(t,S),$$

which is equal to $2\ln(k+1)$ times the dual profit dY. □

3.1 The (h,k)-generalized caching problem

The above algorithm and the analysis can be directly extended to obtain an $O(\ln(k/(k-h+1)))$ competitive fractional algorithm for the (h,k)-generalized caching problem. Here an online cache of size k is compared with an offline cache of size h. The only change to the algorithm is the use of the dual linear program with a cache of size h to lower bound the optimum, and the update of variable $x(p,j)$ in the online algorithm using the exponential function:

$$\frac{dx(p,r(p,t))}{dy(t,S)} = \frac{1}{c_p} \cdot \ln(\eta+1) \cdot \min\{w(S)-k, w_p\} \cdot \left(x(p,r(p,t)) + \frac{1}{\eta}\right), \tag{16}$$

where $\eta = \frac{k}{k-h+1}$. Note that if $h = k$, then $\eta = k$, and this update rule is identical to (11).

3.2 Polynomial Running Time

Since there are exponentially many primal constraints in each iteration, the running time of our algorithm may not be polynomial. However, we can use the properties of our algorithm to make the running time polynomial. In particular, in the online rounding process (described in Section 4), we will see that one does not necessarily require the fractional solution to satisfy all primal constraints. Specifically, for each of the three costs model we consider, we show that there exists a (different) value $\gamma > 1$, such that the algorithm only 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 can actually consider only that set. Now, in general, in an offline setting, all possible subsets may be needed for the knapsack cover constraints, since it is not clear a-priori which set S will have the latter property (more details on this issue can be found in [10]). Fortunately for us, the requirement that variables can only increase monotonically (in the online primal-dual framework) 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 continues for at most n rounds. Thus the fractional algorithm described above can be executed in polynomial time.

4 Rounding the Fractional Solution Online

In this section we show how to obtain a randomized online algorithm, for each of the three cost models, from the fractional algorithm described in Section 3.

High level idea and basic definitions. As discussed previously, in order to obtain an actual randomized algorithm we need to specify a probability distribution, μ, over the various possible cache states and specify how this probability distribution evolves over time. More precisely, at each time step t and real number $\eta \in [0,1)$, the probability distribution μ associates a subset $S(\eta, t)$ of pages that lie in the cache. When a new request arrives at time $t+1$, the sets $S(\eta, t+1)$ change and the cost incurred by the online algorithm is the cost of the fetched pages, averaged over η, i.e.,

$$\mathbb{E}_\eta\Big[C(S(\eta, t+1) \setminus S(\eta, t))\Big] \tag{17}$$

where $C(X) = \sum_{p \in X} c_p$. Of course, the distribution μ and its evolution over time will be closely related to the evolution of the fractional caching solution x.

Relating μ and fractional caching solution. Instead of specifying a probability distribution over cache states, it turns out to be more convenient to (equivalently) work with the probability distribution over sets of pages missing from the cache. This is because in the fractional caching solution the variable $x(p)$ indicates the probability that p is missing from the cache. So at time t, we maintain for each $D \subseteq B(t)$ the probability $\mu(D)$ that exactly the pages in D are missing from the cache.

At any time step t, let $x_p := x(p, r(p,t))$ denote the fraction of page p that is absent from the cache as specified by the fractional solution. Let $\gamma \geq 1$ be a parameter and set $y_p = \min(\gamma x_p, 1)$[1]. All the distributions, μ, that we maintain later will be *consistent* with the fractional solution in the sense that the marginal probability that a page p is not in the cache will be exactly y_p. Formally,

Definition 5 (Consistent Distribution). *A distribution μ is consistent with respect to y, if μ induces the probability distribution on the cache states such that,*

$$\forall p : \qquad \sum_{D \subseteq B(t)} \chi_D(p) \cdot \mu(D) = y_p, \tag{18}$$

where χ_D denotes the characteristic vector of the subset D, i.e., for each page p, $\chi_D(p) = 1$ iff $p \in D$.

The support of the probability distributions that we maintain should only consist of *valid* cache states (or in our notation valid complements of caches). As the total size of pages in a cache can be at most k, this implies the following natural definition,

[1]Note that in this section we only care about the fractional primal solution generated in the previous section. In particular, the term y_p is unrelated to the dual variables $y(t, S)$.

Definition 6 (Valid Distribution). *A probability distribution μ is* valid, *if for any set $D \subseteq B(t)$ with $\mu(D) > 0$, it holds that $\sum_{p \in B(t)} w_p(1 - \chi_D(p)) \leq k$, or alternatively,*

$$\sum_{p \in B(t)} w_p \cdot \chi_D(p) \geq w(B(t)) - k.$$

Online Maintenance. We now discuss the issue of maintaining the distribution μ over time as the fractional solution changes. First, throughout this section we consider the (equivalent) cost version in which the fractional solution pays c_p for both fetching and evicting a page p. As the total amount of evictions and fetches can differ by at most 1 for any page, over the entire time horizon, this increases the cost of the fractional solution by a factor of two. So, in this accounting, the fractional cost of changing the fractional distribution from x to x' is $\sum_p c_p |x'_p - x_p|$.

As we would like to ensure that the cost of modifying μ from time t to $t+1$ is not much more than the fractional cost incurred during that step, it motivates the following definition.

Definition 7 (β-simulation). *Let t be some time during the execution, and suppose the fractional solution changes from y to y' while incurring a fractional cost of d. Let μ be a distribution that is consistent with respect to y and valid. A β-*simulation *is a procedure of updating the probability distribution μ to a probability distribution μ' that is consistent with respect to y' and valid, while incurring a (possibly amortized) cost of at most βd, where $\beta > 0$. In particular, for each $\eta \in [0, 1)$, the simulation procedure specifies the way the set $S(\eta)$ changes.*

The existence of a β-simulation procedure thus implies that

$$\sum_t \mathbb{E}_\eta[C(S_{\eta,t} \oplus S_{\eta,t+1})] \leq \beta \cdot \sum_t \sum_p |y_p^t - y_p^{t+1}|, \tag{19}$$

where $A \oplus B$ denotes the symmetric difference of sets A and B.

The following observation is immediate.

Observation 8. *If there exists a β-simulation procedure with respect to y, and γ is the scaling factor as defined above, and if there exists a c-competitive fractional algorithm, then there exists a randomized algorithm that is $\beta\gamma c$-competitive.*

Proof. As $y_p^t = \min(\gamma x_p^t, 1)$ for every time t and page p, together with (19) the cost of the randomized algorithm can be bounded as

$$\begin{aligned}
\sum_t \mathbb{E}_\eta[C(S_{\eta,t} \oplus S_{\eta,t+1})] &\leq \beta \cdot \sum_t \sum_p |y_p^t - y_p^{t+1}| \\
&\leq \beta\gamma \cdot \sum_t \sum_p c_p |x_p^t - x_p^{t+1}| \\
&= \beta\gamma \cdot C_f \leq \beta\gamma c \cdot C^*,
\end{aligned}$$

where C_f and C^* denote the costs of the fractional solution and the offline optimal solution, respectively. □

By the above observation, our goal is now is to design a β-simulation for the generalized caching problem with β and γ as small as possible. However, designing such a procedure is rather subtle. Indeed, [4] describes an example in which the rounding can get "stuck" if μ is not chosen carefully. In particular, even if μ is consistent with y at time t, if y changes to y' with fractional cost d, it could be the case that moving to any probability distribution μ' from μ, such that μ' is consistent with y', incurs a cost much higher than d. Thus, our approach is to guarantee that the probability distribution μ is not only valid and consistent with y, but rather satisfies an additional property which is model-specific. That is, this property will be different for each of the three models (bit, fault and general) under consideration, and the first lemma in each of the following subsections deals with this new property.

We then construct a β-simulation procedure for each model with the following properties: (i) at each time t, the distribution μ is consistent, valid, and satisfies the stronger (model-specific) property with respect to y; (ii) when y changes to y', incurring a fractional cost d, the distribution μ can be modified to μ' which is consistent with y', valid, satisfies the stronger (model-specific) property, and also incurs a (possibly amortized) cost of at most βd, where $\beta > 0$. Designing such a simulation guarantees that the stronger (model-specific) property can always be maintained, which gives us the desired randomized algorithm by observation 8.

4.1 The Bit Model

In this section we show how to obtain an $O(\log k)$-competitive randomized algorithm for the generalized 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. $\gamma = 3$). We first define the stronger property on the distribution μ that we are going to maintain.

Definition 9 (Balanced Subsets). *A distribution μ has* the balanced subsets *property with respect to y, if for any set $D \subseteq B(t)$ with $\mu(D) > 0$, the following holds for all $0 \leq j \leq U$:*

$$\left\lfloor \sum_{i=j}^{U} \sum_{p \in S(i)} y_p \right\rfloor \leq \sum_{i=j}^{U} \sum_{p \in S(i)} \chi_D(p) \leq \left\lceil \sum_{i=j}^{U} \sum_{p \in S(i)} y_p \right\rceil . \tag{20}$$

Note that the middle sum is simply the number of pages in D with size (and cost) at least 2^j.

Lemma 10. *Let μ be any distribution that is consistent with y and satisfies the balanced subset property, then any set D such that $\mu(D) > 0$ is a valid complement of a cache .*

We will need the following simple mathematical claim.

Claim 11. *Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be arbitrary non-negative real numbers, and let $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n$ be a non-decreasing sequence. If for every $1 \leq j \leq n$,*

$$\sum_{i=j}^{n} x_i \geq -1 + \left(\sum_{i=j}^{n} y_i \right) , \tag{21}$$

then it holds that $\sum_{i=1}^{n} a_i x_i \geq -a_n + \sum_{i=1}^{n} a_i y_i$.

Proof. Let us define $a_0 = 0$. For each $j \in [n]$, multiplying inequality (21) by $(a_j - a_{j-1})$, and noting that $a_j - a_{j-1} \geq 0$, we get

$$(a_j - a_{j-1}) \sum_{i=j}^{n} x_i \geq -(a_j - a_{j-1}) + (a_j - a_{j-1}) \sum_{i=j}^{n} y_i. \tag{22}$$

Summing (22) over j, the left and right hand sides evaluate, respectively, to

$$\sum_{j=1}^{n}(a_j - a_{j-1})\sum_{i=j}^{n} x_i = \sum_{i=1}^{n} x_i \sum_{j=1}^{i}(a_j - a_{j-1}) = \sum_{i=1}^{n} x_i(a_i - a_0) = \sum_{i=1}^{n} a_i x_i,$$

and

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

Hence, we obtain that $\sum_{i=1}^{n} a_i x_i \geq -a_n + \sum_{i=1}^{n} a_i y_i$. □

Proof. (Lemma 10). We prove that under the conditions of the lemma, each subset D with $\mu(D) > 0$ satisfies

$$\sum_{p\in B(t)} w_p \cdot \chi_D(p) \geq w(B(t)) - k.$$

By the consistency property, if $y_p = 1$ then p must be missing from every cache under the distribution μ, and hence in this case $\chi_D(p) = 1$ whenever $\mu(D) > 0$. For any such set D, the condition (20) remains unchanged if we ignore the pages with $y_p = 1$. Formally, let $S' \subseteq B(t)$ be the set of pages with $y_p < 1$, and let $S'(i) = S' \cap S(i)$ be the class i pages in S'. Condition (20) implies that for every $0 \leq j \leq U$:

$$\sum_{i=j}^{U} \sum_{p\in S'(i)} \chi_D(p) \geq \left\lfloor \sum_{i=j}^{U} \sum_{p\in S'(i)} y_p \right\rfloor \geq \left(\sum_{i=j}^{U} \sum_{p\in S'(i)} y_p \right) - 1. \tag{23}$$

Recall that our goal is to show that $\sum_{p\in B(t)} w_p \cdot \chi_D(p) \geq w(B(t)) - k$. As $\chi_D(p) = 1$ for $p \in B(t) \setminus S'$, it suffices to show that $\sum_{p\in S'} w_p \chi_D(p) \geq w(S') - k$. To this end, consider the following:

$$\begin{aligned}
\sum_{p\in S'} w_p \chi_D(p) &\geq \sum_{p\in S'} \min\{w_p, w(S') - k\}\chi_D(p) \\
&= \sum_{i=0}^{U} \sum_{p\in S'(i)} \min\{w_p, w(S') - k\}\chi_D(p) \\
&\geq \frac{1}{2}\sum_{i=0}^{U} \sum_{p\in S'(i)} \min\{2w_p, w(S') - k\}\chi_D(p) \\
&\geq \frac{1}{2}\sum_{i=0}^{U} \min\{2^{i+1}, w(S') - k\} \sum_{p\in S'(i)} \chi_D(p) && (24) \\
&\geq \frac{1}{2}\left(- \min\{2^{U+1}, w(S') - k\} + \sum_{i=0}^{U} \min\{2^{i+1}, w(S') - k\} \sum_{p\in S'(i)} y_p \right) && (25) \\
&\geq -\frac{1}{2}(w(S') - k) + \frac{1}{2}\sum_{i=0}^{U} \sum_{p\in S'(i)} \min\{w_p, w(S') - k\} y_p && (26) \\
&\geq -\frac{1}{2}(w(S') - k) + \frac{3}{2}(w(S') - k) \geq w(S') - k. && (27)
\end{aligned}$$

Here, (24) follows as $w_p \geq 2^i$ for each $p \in S'(i)$. Inequality (25) follows by applying Claim 11 with $a_i = \min\{2^{i+1}, w(S') - k\}$, $x_i = \sum_{p\in S'(i)} \chi_D(p)$ and $y_i = \sum_{p\in S'(i)} y_p$, and observing that

(23) implies that condition (21) needed in Claim 11 holds for our choice of x_i and y_i. Inequality (26) follows as $w_p < 2^{i+1}$. Finally, to see (27), we recall that $y_p < 1$ for each $p \in S'$ and by definition $y_p = \min(3x_p, 1)$, which implies that $y_p = 3x_p$ for each $p \in S'$. As the solution x_p satisfies all the knapsack cover inequalities, and hence in particular satisfies the knapsack constraint corresponding to S', it must satisfy $\sum_{p \in S'} \min\{w_p, w(S') - k\} x_p \geq 3(w(S') - k)$. This gives us that

$$\sum_{i=0}^{U} \sum_{p \in S'(i)} \min\{w_p, w(S') - k\} y_p \quad = \quad 3 \sum_{p \in S'} \min\{w_p, w(S') - k\} x_p \geq 3(w(S') - k).$$

□

The β-simulation: We now describe a β-simulation procedure with $\beta = 10$ for the bit model. This amounts to showing the following:

Lemma 12. *Let y be any fractional caching distribution, and let μ be an arbitrary probability distribution on cache states which is both consistent and has the balanced subsets property with respect to y. Then, given any other fractional caching distribution y', there is a procedure to transform μ into a probability distribution μ' which is both consistent and has the balanced subsets property with respect to y', such that the cost between μ and μ' is at most β times the fractional cost between y and y', for $\beta = 10$.*

Before we prove the above lemma, we note that it readily implies our main result for the bit model. In particular, as the LP solution gives a fractional $O(\log k)$ competitive algorithm, as $\gamma = O(1)$ in Lemma 10, and as $\beta = O(1)$ in Lemma 12, together with Observation 8, we get that:

Theorem 13. *There is a randomized $O(\log k)$-competitive algorithm for the generalized caching problem in the bit model.*

We now prove Lemma 12.

Proof. (Lemma 12) Let us decompose the change from y to y' into a sequence of steps where the marginal y_p of exactly one page p changes from y_p to y'_p. As the fractional cost of changing y to y' is equal to the sum of the fractional costs incurred at each step, it suffice to describe and prove the lemma for a single step.

So, let us fix some page p and assume that $y'_p = y_p + \varepsilon$ for some $\varepsilon > 0$ (the case when $y'_p = y_p - \varepsilon$ is analogous). Let i be the class of p, i.e. $w_p \in [2^i, 2^{i+1})$. Thus, the fractional cost of the change from y to y' is at least $\varepsilon 2^i$.

Construction of μ': We construct μ' as follows. Recall that μ is consistent with the distribution y, and hence for page p, $\sum_{D \subseteq B(t)} \chi_D(p) \cdot \mu(D) = y_p$. To be consistent with y', we need to transform μ to μ' such that $\sum_{D \subseteq B(t)} \chi_D(p) \cdot \mu'(D) = y'_p = y_p + \varepsilon$. To this end, choose an arbitrary collection[2] of sets $D \subseteq B(t)$ with total measure ε under μ such that $p \notin D$, and add the page p to them (i.e, $D \leftarrow D \cup \{p\}$). The resulting distribution μ' is consistent with y'. The cost incurred is at most $2^{i+1} \cdot \varepsilon$, as we evict p from an ε fraction of caches.

Fixing the Balanced Set Property: However, the above distribution μ' may violate the balanced set property. We now fix this. Let us inspect the balance condition (20) more closely.

[2]As μ associates a set D with each real number $\eta \in [0, 1)$, we must specify this choice explicitly. To do this, we can pick the least indices η with total measure ε corresponding to sets that do not contain p.

First observe that it may only be violated for classes $\tilde{i}$ such that $\tilde{i} \leq i$. Moreover, this violation can occur for two reasons. First, the subsets D that changed to $D \cup \{p\}$ may cause the middle term of (20) to increase for some classes $\tilde{i}$ and the left and right terms remain unchanged. Second, the increase of y_p to $y_p + \varepsilon$ may cause the right and left terms to increase for some classes $\tilde{i}$, but the middle term remains unchanged for some sets D. We consider each of these two cases separately.

First Case: Let us consider the (harder) first case. We will fix the unbalanced property one class at a time starting from class i and going down to class 0. Let us consider class i. Let $s = \lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y_p \rceil$ denote the upper bound on class $\geq i$ pages. Note that (in this first case) we are assuming that s does not increase when we replace y_p by $y'_p = y_p + \varepsilon$. As μ satisfied the balanced set property with respect to y, each D with $\mu(D) > 0$ contained exactly s or $s-1$ pages of classes $\geq i$. So, in the distribution μ', each subset D can contain exactly $s-1, s$ or $s+1$ pages belonging to classes $\geq i$. Let ε' be the measure of sets D in μ' that have exactly $s+1$ pages in classes $\geq i$. Note that $\varepsilon' \leq \varepsilon$ since we only added page p to ε measure of the distribution μ. As μ' is consistent by our assumption above,

$$\sum_{D \subseteq B(t)} \mu'(D) \sum_{k=i}^{U} \sum_{p \in S(k)} \chi_D(p) = \sum_{k=i}^{U} \sum_{p \in S(k)} y'_p \leq s.$$

As ε' measure of the sets in μ' have $s+1$ pages of classes $\geq i$, this implies that at least ε' measure of sets in μ' that contain $s-1$ pages of classes $\geq i$.

We arbitrarily[3] choose ε' measure of these sets and match them with the sets containing $s+1$ class $\geq i$ pages (again for explicitness we can choose the smallest lexicographic matching). Consider any pair of sets (D, D') that are matched. Since μ' satisfies condition (20) for class $i+1$, the number of pages in D and D' that lie in classes $i+1$ or higher differs by at most 1 (while the number of pages in class i or higher differs by 2). Hence, $D \setminus D'$ contains at least one page, p' of class i (if there are several such p', we can choose the page with the least index). We move this page from D to D' (i.e., $D' \leftarrow D' \cup \{p'\}$, $D \leftarrow D \setminus \{p'\}$). Note that both D and D' satisfy the balanced sets property (20) of class i after this procedure (they both have exactly s pages of class i or higher). As this procedure transfers one page of class i from ε' measure of the subsets the total cost incurred here is at most $(2\varepsilon') \cdot 2^{i+1} \leq 2^{i+2} \cdot \varepsilon$.

Second Case: Let us now consider the case when $\lceil \sum_{j=i}^{U} \sum_{p \in S(j)} y'_p \rceil$ increases to $s+1$. This is relatively easy. First, we split ε as $\varepsilon' + \varepsilon''$, where ε'' is such that $\sum_{j=i}^{U} \sum_{p \in S(j)} y_p + \varepsilon' = s$ (i.e. the fractional number of pages in classes $\geq i$ just reaches s). We apply the transformation of the previous case with $y'_p = y_p + \varepsilon'$, which by the balanced set condition and consistency ensures that every cache state with $\mu(D) > 0$ has exactly s class $\geq i$ pages. Now, we add p to ε'' measure of the sets D in μ, guaranteeing that cache states contains either s or $s+1$ of class $\geq i$ pages and guarantees that the distribution is consistent with y'.

Handling Smaller Classes: We now show that applying the above steps to class i results in an analogous scenario for classes $i-1$ and lower. In the above procedure for class i, page p is added to at most ε measure of sets D, and some page p' of class i is removed from a subset of the above sets and transferred to sets containing $s-1$ pages from classes $\geq i$. As the resulting distribution satisfies the balanced property for classes $\geq i$, and exactly ε measure of sets D gain a class i page, the resulting μ' may violate condition (20) with respect to class $i-1$ for at most ε measure of subsets D. This results in the same situation with respect to class $i-1$ that we

[3]To make this choice explicit, we can choose the relevant sets with the smallest indices η

previously considered for class i. More specifically, μ' satisfies the balanced subset property for classes $\geq i$, and at most ε measure of D in μ' violate (by at most 1) the balanced property for class $i-1$. So, we apply the above procedures sequentially to fix classes $i-1, i-2, \ldots, 0$ resulting in an additional cost of at most $\sum_{j=0}^{i-1} 2\varepsilon \cdot 2^{j+1} < 4\varepsilon 2^i$.

Thus, the total cost incurred by all the steps (i.e., adding page p to ε fraction of states and fixing the balanced property for classes $i, \ldots, 0$) is at most $(2+4+4)\varepsilon 2^i = 10\varepsilon 2^i$. □

4.2 The General Cost Model

In this section we show how to obtain an $O(\log^2 k)$-competitive randomized algorithm for the caching problem in the general cost model. Let $U \triangleq \lfloor \log_2 k \rfloor$. Let $C = \lfloor \log_2 c_{\max} \rfloor$, where $c_{\max}$ denotes the maximum fetching cost. For $i = 0$ to U, and $j = 0$ to C, we define $S(i,j)$ 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. Recall that it satisfies the knapsack cover inequalities for all subsets. Let $\gamma = U+3$. Thus, for each page p, $y_p = \min\{1, (U+3) \cdot x_p\} = O(\log k) \cdot x_p$. We first define a stronger property on the probability distribution μ that we are going to maintain.

Definition 14 (Balanced size/cost)**.** *A probability distribution μ has* the balanced size/cost *property with respect to y, if for any set $D \subseteq B(t)$ with $\mu(D) > 0$ the following is satisfied. For each size class $0 \leq i \leq U$ and for each $0 \leq j \leq C$:*

$$\left\lfloor \sum_{z=j}^{C} \sum_{p \in S(i,z)} y_p \right\rfloor \leq \sum_{z=j}^{C} \sum_{p \in S(i,z)} \chi_D(p) \leq \left\lceil \sum_{z=j}^{C} \sum_{p \in S(i,z)} y_p \right\rceil. \tag{28}$$

Note that the middle sum is simply the number of pages in D that are of size approximately 2^i, and cost at least 2^j.

We first show that any consistent distribution with the balanced size/cost property is a valid one.

Lemma 15. *A distribution μ that is consistent and has the balanced size/cost property with respect to some fractional solution y, satisfies that D is a valid cache complement state whenever $\mu(D) > 0$.*

Proof. For the proof it suffices to use the left hand side of condition (28) (i.e., the lower bound). Let S' denote the subset of pages with $y_p < 1$. As $y_p = 1$ whenever $\chi_D(p) = 1$, it suffices to show that $\sum_{p \in S'} w_p \chi_D(p) \geq w(S') - k$. Moreover, condition (28) implies that for any $0 \leq i \leq U$:

$$\sum_{z=0}^{C} \sum_{p \in S'(i,z)} \chi_D(p) \geq \lfloor \sum_{z=0}^{C} \sum_{p \in S'(i,z)} y_p \rfloor \geq -1 + \sum_{z=0}^{C} \sum_{p \in S'(i,z)} y_p. \tag{29}$$

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

$$\sum_{p \in S'} w_p \chi_D(p) \quad \geq \quad \sum_{p \in S'} \min\{w_p, w(S') - k\} \chi_D(p)$$

$$
\begin{aligned}
&= \sum_{i=0}^{U}\sum_{j=0}^{C}\sum_{p\in S'(i,j)} \min\{w_p, w(S')-k\}\chi_D(p) \\
&\geq \frac{1}{2}\sum_{i=0}^{U}\sum_{j=0}^{C}\sum_{p\in S'(i,j)} \min\{2w_p, w(S')-k\}\chi_D(p) \\
&\geq \frac{1}{2}\sum_{i=0}^{U}\min\{2^{i+1}, w(S')-k\}\sum_{j=0}^{C}\sum_{p\in S'(i,j)}\chi_D(p) \qquad (30)\\
&\geq \frac{1}{2}\sum_{i=0}^{U}\min\{2^{i+1}, w(S')-k\}\left(-1+\sum_{j=0}^{C}\sum_{p\in S'(i,j)} y_p\right) \qquad (31)\\
&\geq -\frac{U+1}{2}(w(S')-k)+\frac{1}{2}\sum_{i=0}^{U}\sum_{j=0}^{C}\sum_{p\in S'(i,j)}\min\{w_p, w(S')-k\}y_p \qquad (32)\\
&\geq -\frac{U+1}{2}(w(S')-k)+\frac{U+3}{2}(w(S')-k) = w(S')-k. \qquad (33)
\end{aligned}
$$

Inequality (30) follows since $w_p \geq 2^i$ for each page $p \in S'(i,j)$, and Inequality (31) follows from Inequality (29). Inequality (32) follows since $w_p \leq 2^{i+1}$ for each page $p \in S'(i,j)$. Finally, Inequality (33) holds due to the knapsack constraints for S' as follows:

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

Here we use the fact that $y_p = (U+3)x_p$ for each $p \in S'$, and the last inequality follows as x_p satisfies the knapsack constraint for S'. □

We now show how to construct a β-simulation for the general cost model with $\beta = 10$. For this we need to use both left hand and right hand sides of condition (28), and we use an argument similar to the one used in the proof of Lemma 12.

Lemma 16. *Let y and y' be two fractional solutions. There exists a β-simulation procedure with $\beta = 10$ that transforms any distribution μ that is consistent and has the balanced size/cost property with respect to y into a distribution μ' that is consistent and has the balanced size/cost property property with respect to y'.*

Proof. Again, as in Lemma 12, consider page $p \in S(i,j)$ for which y_p is increased by ε. Note that adding and removing page p from cache states can only violate condition (28) for classes $S(i,k)$ for $k \leq j$. We would like to apply the same β-simulation designed for the bit model to fix condition (28). Note that the proof of Lemma 12 is oblivious to the sizes of the pages, and it only used the fact that the costs of the classes are geometrically decreasing. In the bit model, this fact resulted from the sizes of pages in $S(k)$, while here it holds by definition of class $S(i,k)$. Therefore, we can use the same simulation procedure developed for the bit model to transform μ into μ'. Note again that the procedure only fetches and evicts pages that are in classes $S(i,k)$ for $k \leq j$. Thus, classes $S(i',j)$ for $i' \neq i$ are not affected and condition (28) remains valid. □

Since, here, $\gamma = O(\log k)$ and $\beta = 10$, we conclude with our main theorem for this section:

Theorem 17. *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 show how to obtain the improved $O(\log k)$-competitive randomized algorithm for the caching problem in the fault 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.

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 18 (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 $\sum_{p \in G(i)} y_p \geq 3$.*
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 19 (Balanced grouping)**.** *Given a good grouping G, a distribution μ has* the balanced grouping *property with respect to y if, for any set $D \subseteq B(t)$ with $\mu(D) > 0$ the following property holds. For each i, the number of pages $|D \cap G(i)| = \sum_{p \in G(i)} \chi_D(p)$ satisfies*

$$\left\lfloor \sum_{p \in G(i)} y_p \right\rfloor \leq \sum_{p \in G(i)} \chi_D(p) \leq \left\lceil \sum_{p \in G(i)} y_p \right\rceil . \tag{34}$$

At any time the algorithm maintains a good grouping $\mathcal{G}$ of the pages with respect to y. It also maintains a probability distribution μ that has the balanced grouping property with respect to $\mathcal{G}$. We first show that indeed a probability distribution that has the balanced grouping property is a valid probability distribution. Alternatively, every subset D that satisfies the balanced grouping property also satisfies the total size requirement.

Lemma 20. *A probability distribution that has the balanced grouping property with respect to a good grouping $\mathcal{G}$ is a valid probability distribution.*

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 $\sum_{p \in S'} w_p \chi_D(p) \geq w(S') - k$. 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 sizes in $G(i)$. Recall that for each i, we have that $w_i(\min) \leq w_i(\max) \leq w_{i+1}(\min)$. (Define $w_{\ell+1}(\min) = w_\ell(\max)$.) We define $m_i = \min(w_i(\min), w(S')-k)$ for $i = 1, \ldots, \ell+1$. We lower bound the total size of pages in $D \cap S'$ as follows.

$$\begin{aligned}
\sum_{p\in S'} w_p \chi_D(p) &\geq \sum_{p\in S'} \min\{w_p, w(S')-k\}\chi_D(p) = \sum_{i=1}^{\ell}\sum_{p\in G(i)} \min\{w_p, w(S')-k\}\chi_D(p) \\
&\geq \sum_{i=1}^{\ell} m_i \sum_{p\in G(i)} \chi_D(p) \geq \sum_{i=1}^{\ell} m_i(-1 + \sum_{p\in G(i)} y_p) \geq \frac{2}{3}\sum_{i=1}^{\ell} m_i \sum_{p\in G(i)} y_p \qquad (35) \\
&= \frac{2}{3}\left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p\in G(i)} y_p\right) - \frac{2}{3}\left(\sum_{i=1}^{\ell}(m_{i+1}-m_i)\sum_{p\in G(i)} y_p\right) \\
&\geq \frac{2}{3}\left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p\in G(i)} y_p\right) - 8\left(\sum_{i=1}^{\ell}(m_{i+1}-m_i)\right) \qquad (36) \\
&= \frac{2}{3}\left(\sum_{i=1}^{\ell} m_{i+1} \sum_{p\in G(i)} y_p\right) - 8m_{\ell+1} + 8m_1 \\
&\geq \frac{2}{3}\left(\sum_{i=1}^{\ell}\sum_{p\in G(i)} \min\{w_p, w(S')-k\}y_p\right) - 8(w(S')-k) \qquad (37) \\
&\geq 2(w(S')-k)
\end{aligned}$$

Here inequality (35) follows since D is balanced, and hence for each $1 \leq i \leq \ell$,

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

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 (36) 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 (37) follows by considering the knapsack cover inequality for the set S' and observing that $y_p = 15x_p$ for each $p \in S'$:

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

$\square$

The simulation step in the case of the fault model is more involved, since whenever y changes to y' we should also maintain a good grouping, and also update the distribution. At each step, as the value of y changes, the algorithm modifies the probability 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 set that previously satisfied Inequality (34) may not satisfy it anymore. In such a case, we also modify μ, since only sets that satisfy property (34) can belong to the support of μ. Finally, we only bound the amortized cost in each step. We prove the following lemma.

Lemma 21. *Let y and y' be two fractional solutions. As the solution y changes over time we can maintain a good grouping $\mathcal{G}$ and a consistent probability distribution μ that satisfies the balanced grouping property 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 probability 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[4]:

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

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:

$$\Delta C_{on} + \Delta\Phi \leq 405 \Delta C_f. \tag{38}$$

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

Event 1: Assume first that y_p, for $p \in G(i)$, is increased by ε. If y_p increases by ε it must be that x_p is increased by at least $\frac{\varepsilon}{15}$. Thus, in the fault model the fractional cost is at least $\frac{\varepsilon}{15}$.

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 (34). 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 sets with $s+1$ pages in $G(i)$ (the measure of these is at most ε) arbitrarily with sets contains $s-1$ pages,

[4]We have not tried to optimize the constants.

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 $13\varepsilon + 11\varepsilon = 24\varepsilon$ and hence (38) 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 $G(i)$ is initially 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. The total cost incurred is at most 25. Again, (38) holds, as the fractional cost is 0, and the decrease in potential more than offsets the redistribution cost.

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 had 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 (38) holds trivially.

Now suppose that all neighbors of $G(i)$ have weight greater than 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, similarly to the handling of Event 4.3, we split this group into two groups with weight as close as possible. Since the weight is at most 15, the cost of balancing the sets D is at most $16 + 15 = 31$ (using an argument similar to the one 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 the 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 $y \leq 1/2$, in which case its value is $11(1 + y + 2 - y) = 33$. Thus, the potential function decreases by at least 33, while the cost of this step is at most 31, and hence (38) holds.

Event 4: Suppose that for some page $p \in G(i)$, y_p increases to 1, and exits 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 (38) holds. Now consider the case when the weight of $G(i)$ falls below 3. We apply an argument similar to the one 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 (38) holds.

If we need to split the merged set, we note that the above analysis, showing that (38) 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 (38) holds. □

We conclude with the next theorem:

Theorem 22. *There is an $O(\log k)$-competitive algorithm for the caching problem in the fault model.*

5 Concluding Remarks

Very recently, Adamaszek et al. [2] have improved the main result of this paper and obtained an $O(\log k)$-competitive algorithm for the general cost model. Their improvement is based on designing a better rounding scheme for the general cost model that loses only an $O(1)$ factor on top of the $O(\log k)$ factor lost by the fractional algorithm.

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] Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. An o(log k)-competitive algorithm for generalized caching. In *Symposium on Discrete Algorithms, SODA, to appear*, 2012.

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

[4] N. Bansal, N. Buchbinder, and J. Naor. A primal-dual randomized algorithm for weighted paging. In *Proceedings of the 48th annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[5] 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.

[6] 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.

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

[8] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. *Mathematics of Operations Research*, 34:270–286, 2009.

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

[10] 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.

[11] Deeparnab Chakrabarty, Elyot Grant, and Jochen Könemann. On column-restricted and priority covering integer programs. In *IPCO*, pages 355–368, 2010.

[12] 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.

[13] Marek Chrobak, Gerhard Woeginger, Kazuhisa Makino, and Haifeng Xu. Caching is hard - even in the fault model. In *ESA 2010*, pages 195–206, 2010.

[14] 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.

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

[16] 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.

[17] 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.

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

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

[20] 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.

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

[22] N. E. Young. On-line file caching. In *SODA '98: Proceedings of the 9th annual ACM-SIAM symposium on Discrete algorithms*, pages 82–86, 1998.