Don't Melt Your Cache: Low-Associativity with Heat-Sink

Michael A. Bender
Stony Brook University
USA
bender@cs.stonybrook.edu

Alex Conway
Cornell Tech
USA
me@ajhconway.com

Daniel DeLayo
Stony Brook University
USA
ddelayo@cs.stonybrook.edu

Martin Farach-Colton
New York University
USA
martin@farach-colton.com

Jaehyun Han
The University
of North Carolina at Chapel Hill
USA
jaehyun@cs.unc.edu

Linfeng He
Rutgers University
USA
lh818@scarletmail.rutgers.edu

Rob Johnson
VMware Research, USA
USA
rob@robjohnson.io

Sudarsun Kannan
Rutgers
USA
sudarsun.kannan@rutgers.edu

William Kuszmaul
Carnegie Mellon University
USA
william.kuszmaul@gmail.com

Donald Porter
The University
of North Carolina at Chapel Hill
USA
porter@cs.unc.edu

Evan West
Stony Brook University
USA
etwest@cs.stonybrook.edu

ABSTRACT

Perhaps the most influential result in the theory of caches is the following theorem due to Sleator and Tarjan: With $O(1)$ resource augmentation, the basic LRU eviction policy is guaranteed to be $O(1)$-competitive with the optimal offline policy.

Sleator and Tarjan's result applies to LRU on a *fully associative* cache, but does not tell us how to think about caches with low associativity, i.e., caches where each page has only d positions in which it is capable of residing. This means that many modern caches cannot directly apply the result.

It is widely believed that to implement a cache with low associativity, one should still use LRU, but restricted to the d positions that are eligible for eviction. However, this low-associativity version of LRU has never been analyzed.

We show that low-associativity implementations of LRU are often *actually not* constant-competitive algorithms. On the other hand, we give randomized eviction algorithms that *are constant-competitive*, and even a d-associative algorithm that, using any $d = \omega(1)$, and using $1 + o(1)$ resource augmentation, is $1 + o(1)$ competitive with the fully-associative LRU algorithm. Combined, our algorithms suggest a new way of thinking about the design of low-associativity caches, in which one intentionally designs randomized mechanisms that allow parts of the cache which are "overheating" to naturally cool down.

SPAA '25, July 28–August 1, 2025, Portland, OR, USA

ACM ISBN 979-8-4007-1258-6/25/07…$15.00
https://doi.org/10.1145/3694906.3743303

CCS CONCEPTS

• **Theory of computation** → **Design and analysis of algorithms**; **Randomness, geometry and discrete structures**; • **Hardware**; • **Computer systems organization** → Architectures; • **Mathematics of computing** → **Probability and statistics**; *Discrete mathematics*;

ACM Reference Format:
Michael A. Bender, Alex Conway, Daniel DeLayo, Martin Farach-Colton, Jaehyun Han, Linfeng He, Rob Johnson, Sudarsun Kannan, William Kuszmaul, Donald Porter, and Evan West. 2025. Don't Melt Your Cache: Low-Associativity with Heat-Sink. In *37th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '25), July 28–August 1, 2025, Portland, OR, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3694906.3743303

1 INTRODUCTION

In the ***paging problem*** [18], one must maintain a ***cache*** capable of storing up to n pages at a time. The goal is to support a sequence $x_1, x_2, \ldots$ of accesses to pages. If a page x_i is not in cache when accessed, then the access is said to be a ***cache miss***, and the paging algorithm is required to put x_i in cache. This may force the algorithm to ***evict*** another page y from cache. The choice of which page to evict, also known as the ***eviction policy***, is the algorithmic knob that distinguishes different solutions to the paging problem.

Given the sequence $x_1, x_2, \ldots$, the ***offline optimal algorithm*** OPT is any algorithm that minimizes the overall number of cache misses. An online algorithm ALG is said to be α-***competitive*** with OPT, using β ***resource augmentation***, if the total (expected) number of cache

misses that ALG incurs using a cache of size n is at most α times the total number of cache misses that OPT incurs on a cache of size n/β.

The canonical solution to the online paging problem is to use **LRU *eviction***, which is the policy of always evicting the least recently accessed page out of those in cache. In a seminal 1985 paper, Sleator and Tarjan proved that LRU eviction is a provably good policy: with resource augmentation 2, it is 2-competitive with OPT. This result is essentially optimal in the sense that any $O(1)$-competitive policy *must* use $1 + \Omega(1)$ resource augmentation [18].

Sleator and Tarjan's LRU analysis is arguably the most influential result in the theory of caching. The original paper has amassed more than 3000 citations, from both theoreticians and practitioners. Nearly four decades after the analysis was first performed, the LRU policy remains the baseline policy on which almost all real-world cache-eviction policies are based [1, 6, 19].

Caches with Low Associativity. LRU itself, however, it not implementable on most modern caches. This is because most caches [11, 13] are designed with an additional restriction known as ***d-way associativity*** or d-associativity for short, in which each page x has a set of only d positions $h_1(x), h_2(x), \ldots, h_d(x) \in [n]$ where it is allowed to reside in cache. When a page x is brought into cache, the eviction algorithm must respect the associativity: the page y that we evict must be one of the d pages in positions $h_1(x), h_2(x), \ldots h_d(x)$.

Notice that low-associativity caches come in many flavors. For example, a natural choice is to have each h_i be a uniform random hash function. As nomenclature, we will refer to $h_1(x), h_2(x), \ldots, h_d(x)$ as the ***hashes*** of x. Another example is where the cache is partitioned into n/d disjoint sets of size d, each x is hashed to one of those sets via some uniformly random function $h(x)$ and each h_i maps each page x to the ith page of the set $h(x)$. The competitive ratio of an eviction rule depends not only on d but on the design of the underlying low-associativity cache.

This paper studies the ***d-associative paging problem***, which consists of designing a d-associative cache by both specifying the hash functions h_i and designing a page-eviction algorithm that respects the associativity. Recent work [10], implements $d = \Omega(\log n/\log\log n)$ associative first-level caches using hardware hashing. They find that these hardware hashes have sufficiently low latency to be practical.

For $d < n$, d-associative caches cannot directly implement LRU, because LRU does not respect the d-associativity. The folklore solution, which is also the one that is widely used in the practical literature [8, 10, 17], is to use LRU but just on the d pages that are eligible for eviction.

Given the centrality of low-associativity caches, it is surprising that some basic questions about low-associativity paging have not been addressed, for example:

- For what values of d can we design a d-associative cache for which d-LRU has low competitive ratio with low resource augmentation?
- For what values of d is there a d-associative design that admits an eviction rule with a low competitive ratio and low resource augmentation?

In this paper, we address these problems and show the surprising results that d-LRU can be a poor eviction rule (such as for $d = o(\log n/\log\log n)$) but that there exist other good eviction rules, even for $d = 2$.

Competitive Analysis for low-associativity Caches. We need to take care in how we define the competitive ratio of low-associativity caches. Consider a cache where each h_i is a uniformly random hash function. With some probability, it will not be possible to put certain sets of pages into cache together (e.g., if $d + 1$ pages all have the same d hashes as each other). When $d = O(1)$, this can force a $1/\text{poly}(n)$ cache-miss rate even in situations where fully associative offline OPT would not miss at all. Thus, in order to discuss competitive ratios versus fully associative OPT, we will adopt the following modified notion of α-competitiveness: We say that an algorithm ALG is α-competitive with OPT using β-resource augmentation (or ***(α, β)-competitive*** for short), if the miss rate r_{ALG} for ALG on a cache of size n and the miss rate r_{OPT} for OPT on a cache of size n/β satisfy

$$\mathbb{E}[r_{\text{ALG}}] \leq \alpha r_{\text{OPT}} + 1/\text{poly}(n).$$

In other words, we will allow for an *additive* error of $1/\text{poly}(n)$ on the miss rate when comparing ALG and OPT. (In all of our results, the $1/\text{poly}(n)$ term will be concretely bounded by $O(1/n)$).

Finally, we note that we will sometimes consider the competitive ratio (and resource augmentation) of an algorithm versus OPT, but sometimes we will also consider the competitive ratio (and resource augmentation) versus (fully associative) LRU. Such a comparison is meaningful because many paging algorithms, even ones with poor performance in practice, are known to be $(O(1), O(1))$-competitive with OPT. Establishing that an algorithm is $(1 + \varepsilon, 1 + \varepsilon)$-competitive with LRU also implies that it is $(O(1), O(1))$-competitive with OPT, but says something stronger: it is competitive beyond worst-case workloads, and may therefore be a useful paging algorithm.

1.1 Results

We break the results of the paper into three parts. In part 1, we show that LRU is actually a surprisingly poor policy for low-associativity caches, failing to achieve a constant competitive guarantee even for moderately large values of d. In part 2, we show that *random eviction* does much better, achieving an $(O(1), O(1))$-competitive guarantee even for $d = 2$. The surprising effectiveness of random eviction is due to an interesting heat-dissipation effect in which pages naturally migrate away from hot-spots within the cache over time. Finally, in part 3, we show how to leverage this heat-dissipation phenomenon further with an algorithm that we call HEAT-SINK LRU. HEAT-SINK LRU is capable of matching (fully-associative!) LRU's performance up to $(1 + o(1), 1 + o(1))$, even when d is very small. Our algorithmic techniques in Parts 2 and 3 suggest a new approach for designing low-associativity caches: rather than prioritizing LRU-type behavior, one should prioritize the use of randomized mechanisms through which the hot spots in the cache can naturally dissipate their load.

Part 1: The Downfall of LRU. We begin by considering the simplest low-associativity caching policy: each page x hashes to 2 independent random positions $h_1(x), h_2(x)$ in cache, and each eviction performs LRU on the two eligible positions. We call this **2-LRU**. Our first result is a lower bound: We show that 2-LRU is *not* (α, β)-competitive for any $\alpha, \beta \in O(1)$. This is surprising, because 2-LRU has been proposed as the gold standard for how to manage 2-associative caches [8, 17].

One might hope to get around this lower bound by increasing the number of hashes d for each page. It turns out, however, that the

lower bound extends even to moderately large values of d: For any $d \in o(\log n/\log\log n)$, the lower bound holds.

In fact, the constraint that the hashes $h_1(x), \ldots, h_d(x)$ are uniformly random is also not necessary. The lower bound continues to hold even if we allow for arbitrary dependencies between the hashes $h_1(x), \ldots, h_d(x)$ and even if we allow for each individual $h_i(x)$ to be drawn from an arbitrary distribution satisfying a mild distributional assumption that we call *semi-uniformity*. What this means is that, for $d = o(\log n/\log\log n)$, almost all natural variations of d-associative LRU cannot asymptotically match the performance of fully-associative LRU.

Part 2: The Power of Randomized Choice. Our second result is a positive one. Even though LRU performs poorly, there is another policy, which we call **2-RANDOM**, that we prove is $(O(1), O(1))$-competitive with OPT. As in 2-LRU, in 2-RANDOM each page hashes to two random positions, but rather than evicting the least recently used of the two, 2-RANDOM simply evicts a *random* one.

It may seem counter-intuitive that random eviction would outperform LRU. However, 2-RANDOM performs well because of a *heat-dissipation* phenomenon. Consider a page x choosing between two positions $h_1(x), h_2(x)$. Although x does not know it, one of these positions $h_1(x)$ is a "hot spot" (a position in cache that many pages accessed in the near future would like to use) and the other ($h_2(x)$) is a "cold spot". What makes random eviction powerful is that poor decisions (such as using $h_1(x)$) do not have lasting consequences on the state of the cache; if x makes the wrong decision by using $h_1(x)$, then, because $h_1(x)$ is a hot spot, x will soon be evicted from $h_1(x)$ and the decision will therefore be short-lived. The next time that x is accessed, x will get a second chance at making a good decision. Likewise, if x makes a good decision by using $h_2(x)$, then, because $h_2(x)$ is a cold spot, the good decision is likely to be long-lived. What this means is that, even though x's random choice is equally likely to be good or bad, there is nonetheless a tendency for items to naturally make their way into cooler regions of the hash table over time and therefore decrease the amount of contention.

Part 3: HEAT-SINK LRU. Our final result is a d-associative algorithm that we call **HEAT-SINK LRU**. Even with very low associativity, HEAT-SINK LRU is able to compete with *fully associative* LRU up to $1 + o(1)$ factors. More precisely, for any $\epsilon \in (0, 1)$, if we implement HEAT-SINK LRU with associativity $d = \text{poly}(\epsilon^{-1})$, then the algorithm is guaranteed to be $(1 + \epsilon, 1 + \epsilon)$-competitive with LRU. This means that, up to low-order terms, we can get something just as good as LRU even on caches with very low associativity.

The high-level structure of HEAT-SINK LRU is as follows (see also Figure 1 and Section 5). Most of the cache consists of bins, each of which has capacity $d - 2$. Each page x hashes to $\mathbf{Bin}(x)$: a random bin where it is capable of residing. When a new page is brought into a bin and an old page is evicted, the eviction is performed using LRU within the bin.

In addition to the bins, a small portion of the cache is set aside to act as a "heat-sink". The HEAT-SINK constitutes only a $1/\text{poly}(d)$ fraction of the cache overall, and performs evictions using 2-RANDOM. The main challenge is how to "connect" the HEAT-SINK to the bins so that, on one hand, the HEAT-SINK gets only a small fraction of the pages (as it is very small) but so that, on the other hand, whenever a bin is too "hot", it naturally sheds its heat onto the HEAT-SINK.

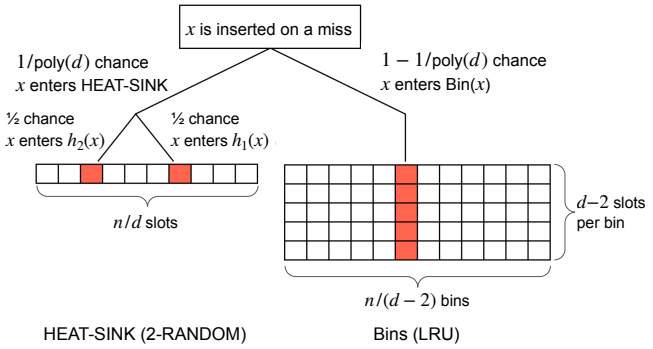

Figure 1: The design of HEAT-SINK LRU. The element x may be stored in any slot in $\text{Bin}(x)$ or in one of 2 HEAT-SINK slots. Thus, these are the only slots we need to check on an access of x. Eviction in the HEAT-SINK is governed by 2-RANDOM and eviction in the bin is governed by LRU. When x is brought into cache, we flip a biased coin to determine if x is to be placed in the HEAT-SINK or $\text{Bin}(x)$. Intuitively, the HEAT-SINK dissipates the heat of the hot (overfilled) bins.

This connection is achieved via a simple probabilistic mechanism. Whenever a page x is brought into cache, it flips a coin. With probability $1 - 1/\text{poly}(d)$, the page is placed into $\text{Bin}(x)$; and with probability $1/\text{poly}(d)$, it is placed in the HEAT-SINK. Note that these coin-flips are not per *page* but instead per *miss*: if the same page experiences multiple cache misses, it makes independent coin flips for each one.

The goal of the HEAT-SINK is to absorb load from any bins that are behaving badly (i.e., too many hot pages hash to the bin). The basic idea is that if a bin is behaving badly, then, by definition, it is incurring many misses. Each miss flips a coin and, with some small probability, abandons the bin for the HEAT-SINK. Over time, if a bin continues to behave badly, then the pages that hash to the bin will slowly migrate away from the bin and into the HEAT-SINK, releasing the pressure on the bin. Perhaps surprisingly, this heat-dissipation mechanism is all that one needs in order to construct a low-associativity cache that is nearly as good as fully-associative LRU.

1.2 Prior Work on Low Associativity

Prior to our work, the known results on low associativity have focused either on directly simulating a fully-associative algorithm using a relatively large value of d [2, 4]; or on settings in which the caching algorithm is permitted to rearrange pages internally within the cache for free (or cheaply), and where the goal is to compete with an OPT that itself has associativity restrictions [5, 7, 15, 16].

Recent work by Bender, Das, Farach-Colton, and Tagliavini [4] considers *set-associative* caches, i.e., caches in which each item hashes to a random bucket of size d. They show that, if $d = \omega(\log n)$, then such a cache is $(O(1), 1 + o(1))$-competitive using LRU (in fact, w.h.p., it contains all of the contents that the OPT it is being compared against contains, at any given moment). On the other hand, when $d = o(\log n)$, they show that no eviction policy is (α, β)-competitive for any $\alpha, \beta \in O(1)$.[1]

[1]Technically, the lower bound in [4] applies only to the LRU policy. However, the lower bound naturally extends to any policy as follows: Suppose we repeatedly access

A similar approach was also taken in earlier work by Bender et al. [2], who showed how to use balls-and-bins techniques in order to achieve a simulation result: so long as $d \geq \text{poly}\log\log n$, it is possible to construct a d-associative cache that, with $1+o(1)$ resource augmentation, perfectly mimics an arbitrary fully associative policy. The style of balls and bins result that they use hits a fundamental barrier at $d = \text{poly}\log\log n$, which might seem to suggest, a priori, that one cannot hope for good results using smaller d. One of the takeaways of the current paper is that it is possible to achieve strong results for any d, not by trying to directly simulate a fully-associative caching policy, but by designing new eviction policies that work well for low-associativity topologies.

Peserico [16] presents yet another perspective on low-associativity caches. He allows each page x to have an arbitrary set of pages $F(x)$ where it is capable of residing in cache (note that $F(x)$ is *not* necessarily drawn from any probability distribution). He then performs a competitive analysis, not with a fully associative OPT (which would be impossible in his setting), but instead with an OPT that is subject to essentially the same associativity restrictions as the online algorithm being designed. Additionally, so that nontrivial competitive ratios are possible to achieve in his setting, Peserico allows for pages within cache to be *rearranged* without first being evicted. Within this model, Peserico gives nearly tight competitive ratios based on the so-called *pseudo-associativity* of a cache [16]. Subsequent work by Buchbinder, Chen, and Naor [7] considered a similar model without resource augmentation, and showed how to design a randomized matroid-based algorithm that obtains an $O(\log^2 n)$ competitive ratio, again against an OPT that is subject to the same associativity constraints as the online algorithm.

Finally, there has also been a great deal of work on *companion caches* [5, 7, 15], which are generalizations of set-associative caches. Companion caches are a combination of a set-associative cache (the *main* cache) with a, typically smaller, fully associative cache (the *companion*).[2] As in [7, 16], the cache is permitted to rearrange pages internally after they are brought into cache, and the goal is to be competitive against an optimal offline algorithm *for the specific cache organization at hand*. Such algorithms do not necessarily offer any competitive guarantee when compared to a fully associative OPT.

2 PRELIMINARIES

Definitions. Given an ***access sequence*** $x_1, x_2, \ldots, x_\ell$, a ***cache*** of size n process the sequence as follows: each time a page x is accessed, it incurs a ***cache miss*** if it is not in cache. When a cache miss occurs, x must be placed into cache, and another page may be evicted. In the paging problem, our goal is to design an eviction policy that incurs as few misses as possible.

A cache is said to have ***associativity*** d if each page x is restricted to d positions $h_1(x), \ldots, h_d(x) \in [n]$ (these are called x's ***hashes***), where the tuple $(h_1(x), \ldots, h_d(x))$ is drawn independently for each page x from some ***hash distribution*** P. When x is brought into cache, if the caching algorithm wishes to places x in position $h_i(x)$, and if $h_i(x)$ is occupied, then the algorithm must evict the page currently in that position.

Given a hash distribution P, we define ***P*-LRU** to be the policy that performs evictions as follows: If, when x is brought into cache, all of x's hashes are occupied, then we evict the least recently accessed page out of the pages in those positions. If P is the distribution that samples each $h_i(x)$ independently and uniformly at random from $[n]$, then we refer to P-LRU simply as ***d*-LRU**. We will also study the **2-RANDOM** policy, which is the policy in which each page hashes to two independent and uniformly random positions $h_1(x), h_2(x)$; and in which, whenever a page x is brought into cache, it chooses a random position out of $h_1(x), h_2(x)$ to go to, evicting any page that was previously there.

We will evaluate policies with competitive analysis. We say that a policy ALG_1, using β ***resource augmentation***, is ***α-competitive*** with a policy ALG_2 (or that ALG_1 is ***(α, β)-competitive*** with ALG_2, for short) if the following is true. Consider any access sequence $x_1, \ldots, x_\ell$, and compare the number of misses M_1 that ALG_1 incurs on a cache of size n with the number M_2 that ALG_2 incurs on a cache of size n/β. The requirement for (α, β)-competitiveness is that, if $M_2 = \omega(n)$, then

$$\mathbb{E}[M_1] \leq (1+\alpha)M_2 + O(\ell/n).$$

We remark that, in general, one cannot help having a term of the form $\ell/\text{poly}(n)$. This can be seen, for example, by observing that, for $d = O(1)$, there is a $1/\text{poly}(n)$ probability that a given set of $d+1$ pages all have their hashes in a set of just d positions, in which case the pages cannot reside in cache together, and any sequence that repeatedly access those pages will incur an $\Omega(1)$ miss rate.

Throughout our analyses, whenever we perform competitive analysis, ALG_2 will either be the offline optimal solution OPT or the fully-associative LRU policy.

Conventions. When performing a competitive analysis to show that ALG_1 is competitive with ALG_2, our default perspective will be that of ALG_1. So, for example, if we refer to a page x experiencing a miss, but do not specify which algorithm we are talking about, the default is ALG_1.

In general, we will not concern ourselves with optimizing constants in our analyses. In fact, in some cases where it is expositionally helpful, we will choose constants that are (clearly) unnecessarily large/small, as this will allow us to make it clear how the different constants should relate to each other (i.e., which constants should be larger/smaller than others).

3 A LOWER BOUND FOR d-ASSOCIATIVE LRU

In this section, we prove a negative result on the performance of 2-LRU:

THEOREM 1. *The 2-LRU policy is not (α, β)-competitive for any $\alpha, \beta \in O(1)$.*

In fact, what we will actually prove is a stronger result. For any associativity $d \in o(\log n/\log\log n)$, and for any distribution P for hashes $(h_1(x), h_2(x), \ldots, h_d(x)) \sim P$, so long as P satisfies a mild distributional assumption that we call *semi-uniformity*, then the lower bound continues to hold: There do not exist any constants α, β such that P is (α, β)-competitive.

pages $x_1, \ldots, x_{n/\beta}$ over and over in rounds. OPT incurs n/β total cache misses. Any d-set-associative policy, with $d = o(\log n)$, can be shown to incur an $2^{-O(d)} = n^{-o(1)}$ miss rate (indeed, a $2^{-O(d)}$ fraction of buckets have more than d keys that has to them, and therefore incur at least one miss per round). By performing enough rounds, one can straightforwardly argue that OPT is achieving asymptotically fewer misses than the set-associative policy.

[2] In the hardware architecture literature, a common use of companion caches is to implement *victim caches* [12].

Formally, a distribution P over $[n]^d$ is ***semi-uniform*** if: Given a sample $(h_1, h_2, \ldots, h_d) \sim P$, an index $j \in [d]$, and a position $i \in [n]$, we have that $\Pr[h_j = i] \leq \text{polylog}(n)/n$. That is, each individual position $i \in [n]$ has probability density for h_j that is at most a polylog n factor larger than the uniform distribution would have. Note that semi-uniformity does not say anything about how the hashes $h_1, h_2, \ldots, h_d$ relate to each other—they can have arbitrary dependencies.

The main theorem of the section is the following:

THEOREM 2. *Let $d = o\left(\frac{\log n}{\log\log n}\right)$ and P be any distribution over $[n]^d$ that is semi-uniform. Let $K \geq 1$ be a parameter, and define* OPT *to be the optimal caching policy on a cache of size $n/\log n$. For any $K \leq n$, there exists an oblivious sequence of $Kn(\log n)^{O(d)} = Kn^{1+o(1)}$ accesses such that OPT incurs cost $O(n)$ and P-*LRU *incurs expected cost $\omega(Kn)$.*

Corollary 1. *For any $d = o\left(\frac{\log n}{\log\log n}\right)$ and for any semi-uniform distribution P over $[n]^d$, there do not exist any positive constants $\alpha, \beta \in O(1)$ such that P-*LRU *is (α, β)-competitive.*

In the rest of the section, we will prove Theorem 2. Let $l = O(n)$ be a large constant multiple of n. We begin by accessing a sequence of $l = 10^6 n$ distinct pages $a_1, a_2, \ldots, a_l$. We refer to this part of the access sequence as ***populating the cache*** and use t_0 as the moment after the l-th access. The effect of this population step is the following:

Lemma 1. *Consider a page $a \notin \{a_1, a_2, \ldots, a_l\}$. The probability that all of a's hashes $h_1(a), \ldots, h_d(a)$ are occupied slots at time t_0 is at least 0.99.*

PROOF. Let p_i be the probability that slots $h_1(a_i), \ldots, h_d(a_i)$ are occupied *conditioned* on the state of the cache after accesses $1, \ldots, i-1$ (note that p_i is a random variable!). Let J be the largest j such that $p_j \leq 0.999$ (J is also a random variable). Since inserting pages cannot decrease the number of occupied slots, p_i is monotonically increasing and $1 - p_i$ is monotonically decreasing. It follows that, for each access a_i, if we condition on $i \leq J$, then the access increases the number of occupied slots with probability at least $1 - p_i \geq 0.001$. This means that, if we define Q to be the number of slots occupied when the J-th access occurs, then

$$\begin{aligned}\mathbb{E}[Q] &= \sum_{i \geq 1} \Pr[i \leq J] \cdot \Pr[a_i \text{ fills an unoccupied slot} \mid i \leq J] \\ &\geq 0.001 \sum_{i \geq 1} \Pr[i \leq J] \geq 0.001 \cdot \mathbb{E}[J].\end{aligned}$$

Since Q can never exceed n, it follows that $\mathbb{E}[J] \leq 1000n$. Thus, by Markov's inequality, we have that

$$\Pr[J > l] \leq \mathbb{E}[J]/l = \mathbb{E}[J]/(10^6 n) \leq 0.001.$$

Now consider a page $x \notin \{a_1, a_2, \ldots, a_l\}$. The probability that all of $h_1(x), \ldots, h_d(x)$ are occupied at time t_0 is at least $1 - p_l$. And the probability that $1 - p_l < 0.999$ is at most $\Pr[J \geq l]$. Since $l \geq 10^6 n$, this probability is at most 0.001. Therefore, the overall probability that $h_1(x), \ldots, h_d(x)$ are all occupied is at least $\Pr[1 - p_l \leq 0.999] \cdot 0.999 \geq 0.999 \cdot 0.999 \geq 0.99$, as desired. □

The rest of the access sequence will access only three sets of pages H, A, and B. The sets A and B are each of size $n/\log^\gamma n$ for some sufficiently large positive constant γ, and are chosen arbitrarily such that $A, B, \{a_1, \ldots, a_l\}$ are all disjoint. The set H is constructed by sampling each page from $\{a_1, \ldots, a_l\}$ independently with probability $1/\log^\gamma n$. Note that, with high probability in n (probability at least, say, $1 - O(n^{-10})$), this results in $|H| \leq O(n/\log^\gamma n)$. We will call the elements of H ***heavy pages*** and the elements of A and B ***light pages***.

After populating the cache (i.e. from t_0 onwards), we access pages in the following pattern: we access H, then A, then H, then B, then H, then A, the H, then B, and so on. This is the entire access sequence.

Call a slot $i \in [n]$ ***negligible*** if it is unoccupied at time t_0. We know from Lemma 1 that most pages $x \in A \cup B$ do not hash to any negligible slots. Now consider a page x in set A or B. Call page x ***promising*** if it satisfies the following three conditions:

(1) All of x's hashes $h_1(x), \ldots, h_d(x)$ are non-negligible.
(2) The pages that were in slots $h_2(x), \ldots, h_d(x)$ at time t_0 were all selected as heavy. Call these pages Y_x.
(3) Every heavy page $z \in H$ is either in Y_x or has disjoint hashes $h_1(z), \ldots, h_d(z)$ from x's hashes $h_1(x), \ldots, h_d(x)$.

The next step in our analysis is to show that each $x \in A$ has a reasonably large probability of being promising.

Lemma 2. *For each $x \in A \cup B$, $\Pr[x \text{ is promising}] \geq 1/(\log n)^{O(d)}$.*

PROOF. For convenience, we associate the three conditions above (that determine whether x is promising) with the events C_1, C_2, and C_3 respectively.

By Lemma 1, the probability that any of $h_1(x), \ldots, h_d(x)$ are negligible is at most 0.01. That is, $\Pr[C_1] \geq 0.99$. Moreover, if the slots are non-negligible, then each page stored in slots $h_1(x), \ldots, h_d(x)$ independently has a $1/\text{polylog}\, n$ chance of being selected as heavy. Thus, $\Pr[C_2 \mid C_1] \geq 1/(\log n)^{O(d)}$, which implies that

$$\Pr[C_1 \wedge C_2] \geq 0.99/(\log n)^{O(d)} = 1/(\log n)^{O(d)}.$$

Next we wish to analyze $\Pr[C_3 \mid C_1 \wedge C_2]$. For any outcome of $h(x) = (h_1(x), \ldots, h_d(x))$, the number Q of pages in $\{a_1, \ldots, a_l\}$ that collide with at least one of x's hashes satisfies

$$\begin{aligned}\mathbb{E}[Q \mid h(x)] &= \sum_{i \in [l]} \sum_{j \in [d]} \sum_{k \in [d]} \Pr[h_j(a_i) = h_k(x)] \\ &= l \cdot d^2 \cdot \frac{\text{polylog}\, n}{n} \leq \text{polylog}\, n,\end{aligned}$$

where the second-to-final step makes use of semi-uniformity. Moreover, since $Q \mid h(x)$ is a sum of independent indicator random variables, we can apply a Chernoff bound to deduce that,

$$\Pr[Q \leq \text{polylog}\, n \mid h(x)] \geq 1 - 1/n^2.$$

As this is true for any value of $h(x)$, we can remove the conditional to get that

$$\Pr[Q \leq \text{polylog}\, n] \geq 1 - 1/n^2.$$

Combining this with the fact that $\Pr[C_1 \wedge C_2] \geq 1/(\log n)^{O(d)}$, we can conclude that

$$\begin{aligned}\Pr[(Q \leq \text{polylog}\, n) \wedge C_1 \wedge C_2] &\geq 1/(\log n)^{O(d)} - 1/n^2 \\ &\geq 1/(\log n)^{O(d)}\end{aligned}$$

where the final step makes use of the fact that $d = o(\log n/\log\log n)$. Finally, if we condition on $(Q \leq \text{polylog}\, n) \wedge C_1 \wedge C_2$, the probability

that any of the $Q - |Y_x|$ pages $z \in H \setminus Y_x$ satisfying $h(z) \cap h(x) \neq \emptyset$ are selected in H is at most

$$\frac{\text{polylog}\, n}{\log^\gamma n} \leq 0.1,$$

where the final step uses of the fact that γ is a sufficiently large positive constant. This means that

$$\Pr[C_3 \mid (Q \leq \text{polylog}\, n) \wedge C_1 \wedge C_2] \geq 0.9,$$

which implies that

$$\begin{aligned}\Pr[C_3, C_2, C_1] &\geq 0.9 \cdot \Pr[(Q \leq \text{polylog}\, n) \wedge C_2 \wedge C_1] \\ &\geq 0.9/(\log n)^{O(d)} = 1/(\log n)^{O(d)}.\end{aligned}$$

□

Call two light pages $a \in A$ and $b \in B$ a ***happy pair*** if:

(1) Both a and b are promising.
(2) $h_1(a) = h_1(b)$.
(3) Every light page $z \notin \{a, b\}$ has disjoint hashes $h_1(z), \ldots, h_d(z)$ from a's and b's hashes $h_1(a), \ldots, h_d(a)$ and $h_1(b), \ldots, h_d(b)$.

The final technical lemma that we need to prove is to show that there are (in expectation) many happy pairs. Before we can do this, we will need to prove the following helper lemma:

Lemma 3. *Let x be a page and condition on an arbitrary outcome for $h_1(x), h_2(x), \ldots, h_d(x)$. Let C be a set of $O(n/\log^\gamma n)$ pages with $x \notin C$. The probability that there exists a page $c \in C$ with hashes $h_1(c), \ldots, h_d(c)$ that collide with $h_1(x), \ldots, h_d(x)$ is at most* 0.01.

Proof. Since distribution P is semi-uniform, we can union bound the probability that a page $c \in C$ has hashes which collide with $h_1(x), \ldots, h_d(x)$ to be at most $d\,\text{polylog}\, n/n$. By a union bound over all $O(n/\log^\gamma n)$ pages in C, the probability that any page collides with x is at most

$$\frac{n}{\log^\gamma n} \cdot \frac{d\,\text{polylog}\, n}{n}.$$

Recalling that γ is sufficiently large, we can conclude that the $\log^\gamma n$ term dominates, so that the overall probability is at most, say, 0.01. □

We can now prove that every pair of light pages $a \in A$ and $b \in B$ has a reasonably large probability of being a happy pair.

Lemma 4. *Given two light pages $a \in A$ and $b \in B$, then*

$$\Pr[(a, b) \text{ is a happy pair}] \geq \frac{1}{n \cdot (\log n)^{O(d)}}.$$

Proof. Let p_a and p_b be the indicator random variables for the events that a and b are promising, respectively. Let $q_{a,b}$ be the indicator random variable for the event that $h_1(a) = h_1(b)$. Let C_1 be the event that $p_a p_b = 1$, let C_2 be the event $q_{a,b} = 1$, and let C_3 be the event that every light page $z \notin \{a, b\}$ has disjoint hashes from a and b. The pair (a, b) is happy if all of C_1, C_2, C_3 happen concurrently.

Recall from Lemma 2 that the probability that a given $x \in A \cup B$ is promising is greater than or equal to $\frac{1}{(\log n)^{O(d)}}$. Thus, we have that $\mathbb{E}[p_a] = \mathbb{E}[p_b] \geq \frac{1}{(\log n)^{O(d)}}$.

Unfortunately, p_a and p_b are not independent. Nonetheless, with a careful argument, we can still lower bound $\Pr[C_1] = \mathbb{E}[p_a p_b]$.

We begin by considering some fixed outcome for $S(t_0)$, the state of the cache (and therefore the distribution of heavy pages) at time t_0. Let $p_a \mid S(t_0)$ and $p_b \mid S(t_0)$ be the conditional values of p_a, p_b, and $q_{a,b}$ given $S(t_0)$. Recall that p_a and p_b rely on the three conditions from the definition of promising; these conditions are determined by $S(t_0)$ and the d-tuple of hashes for the potentially promising page. As the d-tuples for a and b are independent from each other (and even independent of $S(t_0)$), we have for any $S(t_0)$ that the conditional random variables $p_a \mid S(t_0)$ and $p_b \mid S(t_0)$ are independent.

Since p_a and p_b when conditioned on $S(t_0)$ are independent,

$$\Pr[(p_a \mid S(t_0)) \wedge (p_b \mid S(t_0))] = \mathbb{E}[(p_a \mid S(t_0))^2].$$

By convexity (and, specifically, by Jensen's inequality),

$$\begin{aligned}\mathbb{E}_{S(t_0)}[(p_a \mid S(t_0))^2] &\geq \mathbb{E}_{S(t_0)}[p_a \mid S(t_0)]^2 \\ &\geq \left(\frac{1}{(\log n)^{O(d)}}\right)^2 = \frac{1}{(\log n)^{O(d)}}.\end{aligned}$$

It follows that

$$\begin{aligned}\Pr[C_1] = \mathbb{E}[p_a p_b] &= \mathbb{E}_{S(t_0)}[\Pr[(p_a \mid S(t_0)) \wedge (p_b \mid S(t_0))]] \\ &\geq \frac{1}{(\log n)^{O(d)}}.\end{aligned}$$

Next we argue that $\Pr[C_1 \wedge C_2] \geq \frac{1}{n}\Pr[C_1]$. To do so, we will need the following standard claim:

Claim 1. *Let x, y be iid random variables over $[n]$. Then $\Pr[x = y] \geq 1/n$.*

Proof. Let $q_i = \Pr[x = i] = \Pr[y = i]$. From the Cauchy-Schwarz Inequality

$$\left(\sum_{i\in[n]} q_i\right)^2 \leq \left(\sum_{i\in[n]} q_i^2\right)\left(\sum_{i\in[n]} 1\right).$$

It follows that

$$\frac{\left(\sum_{i\in[n]} q_i\right)^2}{n} \leq \sum_{i\in[n]} q_i^2.$$

Since the sum of the probabilities must be equal to 1 (i.e. $\sum_{i\in[n]} q_i = 1$), it follows that

$$\sum_{i\in[n]} q_i^2 \geq 1/n,$$

which completes the proof. □

If we condition on any outcome of $S(t_0)$ and on p_a and p_b being 1, then the conditional distributions for a and b are iid. It follows from Claim 1 that

$$\Pr[q_{a,b} = 1 \mid p_a = 1, p_b = 1, S(t_0)] \geq 1/n.$$

This means that

$$\Pr[q_{a,b} p_a p_b = 1 \mid S(t_0)] \geq \frac{1}{n} \cdot \Pr[p_a p_b = 1 \mid S(t_0)].$$

Taking an expected value over $S(t_0)$, it follows that

$$\Pr[q_{a,b} p_a p_b = 1] \geq \frac{1}{n} \cdot \Pr[p_a p_b = 1],$$

meaning that $\Pr[C_1 \wedge C_2] \geq \frac{1}{n}\Pr[C_1]$.

Now condition on C_1 and C_2. By Lemma 3, even if we condition on arbitrary outcomes for $S(t_0)$ and for the hashes for a and b (which

is stronger than conditioning on C_1 and C_2), the probability that any light page $c \notin \{a, b\}$ collides with either a or b is at most 0.02. Thus $\Pr[C_3 \mid C_1, C_2] \geq 0.98$.

Putting the pieces together,

$$\Pr[(a,b) \text{ is happy pair}] \geq \Pr[C_3 \mid C_1, C_2] \cdot \Pr[C_1 \wedge C_2] \geq 0.98 \cdot \frac{1}{n(\log n)^{O(d)}} = \frac{1}{n \cdot (\log n)^{O(d)}}.$$

□

Having lower-bounded the probability of a given pair $(a, b) \in A \times B$ being happy, we can now complete the proof of Theorem 2.

Proof of Theorem 2. Recall that, after populating the cache, we access pages in the following pattern: we access all of H, A, H, B, H, A, H, B, etc. By design, heavy pages are never evicted from the cache by light pages. This means that each happy pair contends for their first hash slot. So, if (a, b) are happy, then each access to a will evict b and each access to b will evict a. Critically, this implies that every access to a or b is a miss. By Lemma 4, the expected number of happy pairs is $\Theta(\frac{n^2}{\text{polylog}\, n}) \cdot \frac{1}{n \cdot (\log n)^{O(d)}} = \frac{n}{(\log n)^{O(d)}}$. Since each page can be a member of at most one happy pair, it follows that the expected number of pages in A that are a happy pair is

$$\frac{n}{(\log n)^{O(d)}} \geq \frac{|A|}{(\log n)^{O(d)}}.$$

This implies the expected miss rate for P-LRU is at least $1/(\log n)^{O(d)}$ each time that it accesses H, A, H, B.

Recall that $|A|, |B| = n/\log^\gamma n$ and that, with probability at least $1 - O(n^{-10})$, we have $|H| \leq O(n/\log^\gamma n)$. It follows that, with probability $1 - O(n^{-10})$, the set of accessed pages after t_0 is of size $O(n/\log^\gamma n) \leq n/\log n$, and thus that OPT can store all the pages in cache simultaneously. This means that, with high probability in n, OPT incurs a total of $O(n)$ cache misses. That is, with probability $O(n^{-10})$, OPT's misses are bounded only by the access length. Since the total length of the access sequence $Kn^{1+o(1)}$ is bounded by $O(n^3)$, it follows that OPT's overall expected number of misses is $O(n)$.

On the other hand, after t_0, P-LRU incurs misses on a $\frac{1}{(\log n)^{O(d)}}$ expected fraction of the remaining accesses. It follows that within $Kn(\log n)^{O(d)} = Kn^{1+o(1)}$ accesses P-LRU incurs $\omega(Kn)$ expected misses. This completes the proof of Theorem 2.

□

4 AN UPPER BOUND FOR 2-RANDOM

In this section, we prove that the 2-RANDOM policy (with an associativity of only 2!) is $(O(1), O(1))$-competitive. In the previous section, we found that d-LRU with associativity $d \in o(\log n/\log\log n)$ is *not* $(O(1), O(1))$-competitive (Corollary 1). That is, surprisingly, trying to make informed eviction decisions with d-LRU is worse than making random choices even when choosing between only 2 slots.

Recall that the 2-RANDOM policy maps each page x to two uniformly random and independent hashes $h_1(x), h_2(x) \in [n]$. Moreover, when an item x is brought into cache, the policy simply selects a random $i \in \{1, 2\}$, places x in position $h_i(x)$, and evicts any page that previously resided there.

Theorem 3. *There exist positive constants α, β such that, with β resource augmentation,* 2-RANDOM *is α-competitive with* OPT.

Throughout the section, we will define OPT to be the (offline) optimal caching policy on a cache of size n/β. We will break OPT's access sequence into ***phases***, where in each phase OPT incurs $2n/\beta$ cache misses. We will then establish Theorem 3 by arguing that, within the same phase, 2-RANDOM incurs $O(n)$ cache misses.

Consider some phase, and let S be the set of pages accessed during it. As OPT incurs n/β cache misses and initially has at most n/β pages in cache, we know that $|S| \leq 2n/\beta$.

The first issue that we must consider is whether 2-RANDOM is even *capable* of storing all of S in cache at once. For this, we make use of the following standard technical lemma about orientations of random graphs (see, e.g., . [9, Theorem 4]).

Lemma 5. *Let G be a random (multi) graph on n vertices with n/β edges (sampled uniformly) for some constant $\beta > 2$. Then, with probability $1 - O(1/n)$, it is possible to assign each edge to one of its two vertices so that each vertex is assigned at most one edge in total.*

Later in the paper, it will also be helpful to have the following corollary:

Corollary 2. *Let G be a random (multi) graph on n vertices with n/β edges (sampled uniformly) for some (potentially super-constant) $\beta > 2$. Then, with probability $1 - O(1/(\beta n))$, it is possible to assign each edge to one of its two vertices so that each vertex is assigned at most on edge in total.*

Proof. The interesting case is $\beta = \omega(1)$. Suppose we were to generate $r = \lfloor \beta/3 \rfloor$ random edge sets $E_1, E_2, \ldots, E_r$, each with n/β random edges. Each has some probability $1 - p$ of being orientable for some p, and the probability that their union $E = \bigcup E_i$ is orientable is at most $(1-p)^r$. On the other hand, we have by Lemma 5 that E is orientable with probability at least $1 - O(1/n)$. Thus $(1-p)^r \geq 1 - O(1/n)$, implying that $p \leq O(1/(rn)) = O(1/(\beta n))$, as desired. □

Having established that 2-RANDOM is capable of storing S, we can now turn to the interesting part of the analysis: assuming that 2-RANDOM is capable of storing S, how many cache misses should we expect it to incur during the current phase?

For this, we will need one more lemma about random graphs:

Lemma 6. *Consider a set of $n/(4e^2)$ pages, each of which hashes to two random positions in cache. Let G be the (multi) graph on n vertices, where edge (j, k) indicates that some page satisfies $\{h_1(x), h_2(x)\} = \{j, k\}$. For any given page x, the size i of the connected component C that contains the edge $\{h_1(x), h_2(x)\}$ satisfies*

$$\Pr[|C_x| \geq i] \leq \frac{1}{4^{i-2}}$$

for all $i \geq 3$.

Proof. For a component C to be size i, three necessary conditions are:

(1) The component C consists of i positions.
(2) Both of x's hashes are in the component.
(3) $i - 2$ other pages R also need to have all of their hashes in C

There are $\binom{\frac{n}{4e^2}}{i-2}$ options for R and $\binom{n}{i}$ options for C. For any given option, the probability that all of the hashes for R and x are in C is $(\frac{i}{n})^{2(i-1)}$. Thus,

$$\Pr[|C_x| \geq i] \leq \binom{\frac{n}{4e^2}}{i-2} \cdot \binom{n}{i} \cdot \left(\frac{i}{n}\right)^{2(i-1)}$$

Using the identity $\binom{m}{i} = (m \cdot (m-1) \cdots (m-i+1))/i! \leq m^i/i!$, we get

$$\Pr[|C_x| \geq i] \leq \frac{(n/4e^2)^{i-2}}{(i-2)!} \cdot \frac{n^i}{i!} \cdot \left(\frac{i}{n}\right)^{2(i-1)},$$

which implies that

$$\Pr[|C_x| \geq i] \leq \frac{1}{(4e^2)^{i-2}} \cdot \frac{i^{2i-2}}{(i!)^2}.$$

Next, using the identity that $i! \geq (i/e)^i$ (i.e., Stirling's bound), we can conclude that

$$\Pr[|C_x| \geq i] \leq \frac{e^{2i}}{e^{2i-2}4^{i-2}} \cdot \frac{i^{2i-2}}{i^{2i}} = \frac{e^2}{4^{i-2}} \cdot \frac{1}{i^2}.$$

Since $i \geq 3$,

$$\Pr[|C_x| \geq i] \leq \frac{1}{4^{i-2}},$$

as desired. □

We want to show that, for a given $y \in S$, the expected number of times that y gets evicted (during the current phase) is $O(1)$. We will begin by making a weaker claim: if y is contained in a connected component C of some size r, then the number of misses that y incurs during the phase is bounded above by a geometric random variable with mean 2^r. The high-level idea here is that bad choices for where to put the pages in C are temporary (as the pages will promptly get evicted), but good choices are forever (as the pages will never get evicted again). In particular, if all of the pages in C happen to at some point place themselves into cache in a way that is compatible with each other, then they will experience no more misses for the rest of the phase.

Lemma 7. *Let E be the event that* 2-RANDOM *is capable of storing all of S and condition on E. Let G be the (multi) graph on n vertices with edges $(h_1(x), h_2(x))$ for each $x \in S$. Let $y \in S$, and condition on y's edge $\{h_1(y), h_2(y)\}$ being in a connected component C of some size $|C| = r$. Then the number M of misses that* 2-RANDOM *incurs on y during the phase satisfies*

$$\Pr[M > i] \leq (1 - 2^{-|C|})^i = (1 - 1/2^r)^i.$$

Proof. Let $C = \{c_1, c_2, \ldots\}$ be the set of pages whose edges form y's connected component in G. Because we are conditioning on E, there exists at least one injection (i.e., valid configuration) $f : C \to [n]$ such that each c_i satisfies $f(c_i) \in \{h_1(c_1), h_2(c_2)\}$. We're going to split our analysis into *mini-phases*: We begin a mini-phase whenever some $c \in C$ gets placed into a position that is *not* $f(c)$.

Critically, we have two properties:

- Each page $c \in C$ experiences at most one miss per mini-phase. This is because, if c misses and goes to $f(c)$, and then some c' later evicts c, then c' must have triggered a new phase.
- Each mini-phase has probability at least $1/2^{|C|}$ of being the last one. In particular, if we consider some mini-phase, and let $C' \subseteq C$ be the subset of C that gets accessed between the start of the mini-phase and the rest of the full phase, then we can make the following argument. If, for each page $c \in C'$, the next time that c experiences a miss, c is placed in position $f(c)$, then all of the pages $c \in C'$ will get placed into cache together. From this point forward, none of the pages will get evicted during the rest of the phase, so none of them will incur any additional misses. The probability of every $c \in C'$ choosing $f(c)$ is at least $1/2^{|C'|} \geq 1/2^{|C|}$.

It follows that the number of misses M that y incurs is bounded by

$$\Pr[M > i] \leq (1 - 2^{-|C|})^i = (1 - 1/2^r)^i.$$

□

Combining Lemmas 6 and 7, we can now bound the expected number of misses that any $y \in S$ incurs during the current phase.

Lemma 8. *Let E be the event that* 2-RANDOM *is capable of storing all of S. For a given $y \in S$, let M be the number of times that y gets evicted during the current phase. Then $\mathbb{E}[M \mid E] = O(1)$.*

Proof. By Lemma 6, we can bound $\mathbb{E}[M \mid E]$ by

$$\mathbb{E}[M \mid E] = \sum_{i \geq 0} \Pr[M > i \mid E] \leq \sum_{i \geq 0} \mathbb{E}\left[\left(1 - 2^{-|C|}\right)^i \mid E\right]$$
$$\leq \mathbb{E}\left[\sum_{i \geq 0} \left(1 - 2^{-|C|}\right)^i \mid E\right] = \mathbb{E}\left[2^{|C|} \mid E\right].$$

Recall that $|S| \leq 2n/\beta$. Supposing $\beta \geq 8e^2$, it follows from Lemma 6 that $\Pr[|C| \geq i] \leq 1/4^{i-2}$ for all $i \geq 3$. Since E occurs with probability more than $1/2$, this implies that $\Pr[|C| \geq i \mid E] \leq 2/4^{i-2}$. This means that

$$\mathbb{E}\left[2^{|C|} \mid E\right] \leq \sum_j 2^j \Pr[|C| = j \mid E]$$
$$\leq O(1) + \sum_{j \geq 3} 2^j \cdot \frac{2}{4^{j-2}}.$$

Critically, the 2^j term is dominated by the 4^{j-2} term, so the entire sum evaluates to $O(1)$, as desired.

It is worth remarking that, in the calculation of $\mathbb{E}[2^{|C|}]$, given that $|C|$ itself is a geometric random variable, there is a risk that the expectation could be infinite. It is only because of the fact that $|C|$ has a geometric ratio *less* than $1/2$ that the expectation comes out to $O(1)$. □

Putting the pieces together, we can prove Theorem 3.

Proof of Theorem 3. Consider an access sequence in which OPT incurs $M_{\text{OPT}} = \omega(n)$ misses, $x_1, x_2, \ldots, x_\ell$. Break the sequence into phases in which OPT incurs n/β misses (or fewer for the final phase), and let S be the set of items accessed during a given phase. Since OPT incurs n/β misses, we know that $|S| = O(n)$. By Lemma 5, we have with probability $1 - O(1/n)$ that the pages S can be placed into cache together; and conditioned on this, we have by Lemma 8 that the expected number of misses during the phase is $O(|S|) = O(n)$. If there are q phases overall, and if ℓ is the total number of accesses overall, then it follows that 2-RANDOM expects to incur a total of at most $O(\ell/n) + O(qn)$ misses. Since OPT incurs at least $(q-1) \cdot \beta/n$, and since $q > 1$, it follows that 2-RANDOM is $O(1)$-competitive with OPT. □

5 HEAT-SINK LRU: AS GOOD AS LRU, BUT WITH LOW ASSOCIATIVITY

We now present and analyze HEAT-SINK LRU. Recall that our goal is to be $(1 + O(\epsilon), 1 + O(\epsilon))$ competitive with fully-associative LRU for some parameter ϵ. As a small abuse of notation, we will define HEAT-SINK LRU on a cache of size $(1 + \epsilon)n$ (rather than n) and compare to LRU on a cache of size $(1 - 2\epsilon)n$. Of course, up to a change of parameters, this is equivalent to having HEAT-SINK LRU on a cache of size n and LRU on a cache of size $(1 - \Theta(\epsilon))n$.

We define HEAT-SINK LRU as the following policy:

- Construct $\frac{n}{b}$ bins of size $b = \epsilon^{-3}$ each. Map each page x to a random bin $\mathrm{Bin}(x)$.
- Allocate n/d additional slots to implement what we will call the HEAT-SINK.
- When we incur a miss on a page x, we flip a biased coin: With probability $p = \frac{1}{\mathrm{poly}\, d}$, we place x in the HEAT-SINK, and otherwise we place x into $\mathrm{Bin}(x)$.
- Misses that are routed to bin j implement evictions by performing LRU on the bin, and misses that are routed to the HEAT-SINK implement evictions by performing 2-RANDOM on the HEAT-SINK.

Note that, overall, the total associativity is $d = b + 2 = O(\epsilon^{-3})$.[3] We will prove the following theorem:

THEOREM 4 (HEAT-SINK LRU). *Let $\epsilon \in (0, 1)$ satisfy $\epsilon^{-1} = n^{o(1)}$. HEAT-SINK LRU is an $O(\epsilon^{-3})$-associative replacement algorithm on a cache of size $(1 + \epsilon)n$ that is $(1 + O(\epsilon))$-competitive with LRU on a cache of size $(1 - 2\epsilon)n$.*

Of course, by a change of variables (so that HEAT-SINK LRU is on a cache of size n and so that ϵ is adjusted by a constant factor), this implies:

Corollary 3. *Let $\epsilon \in (0, 1)$. There exists an $O(\epsilon^{-3})$-associative replacement algorithm that is $(1 + \epsilon, 1 + \epsilon)$-competitive with LRU.*

The rest of the section will be spent proving Theorem 4. Throughout, we will assume without loss of generality that ϵ is at most a small positive constant.

To prove Theorem 4, we will break the access sequence into ***phases***, where in each phase OPT incurs ϵn cache misses (although in the final phase it may incur fewer). We will spend most of the analysis focusing on one such phase $W = [t_0, t_1]$, spanning time t_0 to t_1 (where time is measured in terms of the number of accesses that have occurred so far). Let A be the set of pages resident in LRU's cache at the beginning of W, and let B be the set of pages that LRU misses on during the window.

As part of our analysis, we will use an accounting argument in which pages sometimes receive ***bonus points*** that HEAT-SINK LRU can use to pay for the next time that it misses on that page. The rules for how bonus points are awarded is that, for each page x, it gets a bonus point whenever LRU misses on x but HEAT-SINK LRU hits; and whenever HEAT-SINK LRU misses on x and decides to send x to the HEAT-SINK.

Call a bin j ***cool*** if $|\{x \in A \cup B \mid \mathrm{Bin}(x) = j\}| \leq b$, and ***hot*** otherwise. Similarly, call a page $x \in A \cup B$ ***cool*** if $\mathrm{Bin}(x)$ is cool and ***hot*** if $\mathrm{Bin}(x)$ is hot. We begin by arguing that HEAT-SINK LRU can always use bonus points to pay misses on cool pages.

Lemma 9. *Let j be a cool bin and let $x \in A \cup B$ satisfy $\mathrm{Bin}(x) = j$. Each access to x during W either is a miss for LRU, is a hit for HEAT-SINK LRU, or is paid for by a bonus point.*

PROOF. Let $A_j = \{x \in A \mid \mathrm{Bin}(x) = j\}$ and similarly $B_j = \{x \in B \mid \mathrm{Bin}(x) = j\}$. By construction, $x \in A_j \cup B_j$.

We begin by considering the first access to x during W. If $x \in B_j \setminus A_j$, then the first access to x during window W is a miss for LRU. On the other hand, if $x \in A_j$, we can analyze the first access to x with three cases:

(1) At time t_0, HEAT-SINK LRU has x in bin j. The fact that $x \in A_j$ means that, at time t_0, the only bin-j pages to have been accessed more recently than x are also in A_j. Thus, at all points in time during W, the only bin-j pages to have been accessed more recently than x are in $A_j \cup B_j$. Since $|(A_j \cup B_j) \setminus \{x\}| < b$, we have during all of W that x is always among the b most recently accessed bin-j pages. This means HEAT-SINK LRU will not evict x at any point during window W, which implies that x will not experience any misses
(2) At time t_0, HEAT-SINK LRU has x is in the HEAT-SINK. Then, x has a bonus point that will pay for x's first miss in W.
(3) At time t_0, the HEAT-SINK does not have x is either in bin j nor the HEAT-SINK. The fact that x is not in bin j but *is* among the most recent $|A_j| \leq b$ accessed bin-j items (at time t_0) means that, last time x was accessed (before time t_0), HEAT-SINK LRU must have placed x in the HEAT-SINK. (Indeed, if x had been placed in bin j, then the LRU policy within the bin would ensure that it is still there at time t_0). It follows that x has a bonus point that it can use to pay for its first miss during W.

Having proven the lemma for the first access to x in W, now consider the i-th access for some $i > 1$. If, after the $(i-1)$-th access for HEAT-SINK LRU, x resided in the heat sink, then x is guaranteed to have a bonus point that it can use to pay for any miss it experiences on its i-th access. Otherwise, it must be that after the $(i-1)$-th access for HEAT-SINK LRU, x was in bin j. In this case, x will remain among the $|A_j \cup B_j| \leq b$ most recently accessed pages for bin j for the rest of window W. This means that bin j will not evict x, and that x experiences a hit on its i-th access. □

Later in the section, Lemma 9 will allow for us to bound the misses on cool bins with a simple charging argument. The more difficult challenge is how to handle hot bins. Intuitively, HEAT-SINK LRU dissipates the "heat" from the hot bins by routing a portion of their pages (or heat) to the HEAT-SINK. At a high level, our analysis of hot bins will be broken into three pieces. We will argue that: (1) cool bins do not contribute very much to the HEAT-SINK, meaning that the HEAT-SINK can focus most of its capacity on handling misses from hot bins; (2) the total number of distinct hot pages is small enough that HEAT-SINK could, in principle, store all of them simultaneously; and (3) hot bins are able to shed their "heat" (i.e., overload) to the HEAT-SINK quickly enough that the cache misses incurred along the way are tolerable. Combining these with the fact that HEAT-SINK is implemented using 2-RANDOM, we will be able to obtain a bound of the form $\epsilon^{\omega(1)} n$ on the the total expected number of misses that HEAT-SINK LRU incurs on hot pages.

[3] In fact, the same analysis would work even if we were to reduce b to ϵ^{-2} polylog ϵ^{-1}, for some appropriate polylog, resulting in $d = O(\epsilon^{-2}\,\mathrm{polylog}\,\epsilon^{-1})$.

We begin by arguing that cool bins do not contribute very much to the HEAT-SINK during W.

Lemma 10. *Let k be the number of number of distinct cool pages x such that there is at least one miss for x during W in which* HEAT-SINK LRU *routes x to the heat sink. With probability $1 - n^{-\omega(1)}$, we have $k \in O(d^2 n)$.*

PROOF. Consider a page x that hashes to a cool bin and that is accessed at least once during W. The first time that HEAT-SINK LRU incurs a miss on x during W, x will enter the heat sink with probability ϵ^2. If x does not enter the heat sink at that point, then it must be placed in bin $j = \text{Bin}(x)$. This means that x will not experience any more misses, since, for the rest of W, x will be among the most recent b items accessed for bin j and will therefore not get evicted.

Therefore, the only way that x can ever make it into the heat sink during W is if it gets placed into the heat sink during its first miss. There are deterministically at most n pages that hash to cool bins, and each independently has probability poly $d \le d^2$ of being placed into the heat sink on their first miss in W. The number of pages from cool bins that get placed into the HEAT-SINK during W is therefore a sum of independent indicator random variables with mean at most $d^2 n$. By a Chernoff bound, this sum guaranteed to be $O(d^2 n)$ with probability at least $1 - n^{-\omega(1)}$. □

Next we bound the total number of hot pages.

Lemma 11. *With probability $1 - n^{-\omega(1)}$, the total number Q of hot pages $x \in A \cup B$ is $\epsilon^{\omega(1)} n$.*

PROOF. We begin by bounding $\mathbb{E}[Q] = \sum_{x \in A \cup B} \Pr[x \text{ is hot}]$. In order for $x \in A \cup B$ to be hot, at least $b-1$ other balls from $A \cup B$ must hash to $\text{Bin}(x)$. The expected number of other balls that hash to the bin is at most $(|A| + |B|)/(n/b) \le ((1-2\epsilon)n + \epsilon n)/(n/b) \le (1-\epsilon)b$. As $b = \epsilon^{-3}$, the only way for the bin to receive at least $b-1$ other balls is if a sum of independent 0-1 random variables with mean $\le \epsilon^{-3} - \epsilon^{-2}$ takes a value at least $\epsilon^{-3} - 1$. By a Chernoff bound, this occurs with probability at most $\epsilon^{\omega(1)}$. Thus, we have that $\Pr[x \text{ is hot}] \le \epsilon^{\omega(1)}$, which implies that

$$\mathbb{E}[Q] \le |A \cup B| \cdot \epsilon^{\omega(1)} \le \epsilon^{\omega(1)} n.$$

Having bounded $\mathbb{E}[Q]$, we next show that, with very high probability, Q is not too far from its mean. We apply McDiarmid's inequality [14], which says that for any real-valued function $f(X_1, \ldots, X_n)$ on n independent random variables, if changing a given X_i is guaranteed to change f by at most Δ, then $\Pr[|f - \mathbb{E}[f]| \ge k\Delta] \le e^{-\Omega(k^2)}$. Given the pages $A \cup B$, we create the function f that takes as input the bin-hashes of the pages in $A \cup B$ and returns the total number of hot pages. Changing a single input to $f(A \cup B)$ (i.e. a single page's bin hash value) can change the number of hot pages by at most b (the worst case is that the input change toggles the hot/cool state of a bin with b pages). We can therefore apply McDiarmid's Inequality to conclude that

$$\Pr\left[f(A \cup B) - \mathbb{E}[f(A \cup B)] > kb\sqrt{|A \cup B|}\right] \le e^{-\Omega(k^2)}.$$

Recalling that $|A \cup B| \le O(n)$, that $\mathbb{E}[f(A \cup B)] \le \epsilon^{\omega(1)} n$, and that $b = n^{o(1)}$ gives

$$\Pr\left[f(a,b) > \epsilon^{\omega(1)} n + k n^{1/2 + o(1)}\right] \le e^{-\Omega(k^2)}.$$

Setting k to be, say $\log n$, completes the proof of the lemma. □

Lemmas 10 and 11 combine to imply that the *total* number of distinct pages (both hot and cool) that incur misses on the heat sink during W is quite small (much smaller than the capacity of the heat sink). This, in turn, allows us to employ our analysis of 2-RANDOM in order to conclude that each individual hot page should only expect to incur a constant number of misses on the heat sink during W.

Lemma 12. *There is a random event E that occurs with probability $1 - O(1/n)$, and conditioned on which the following is true:*

- *Using* HEAT-SINK LRU*, and for a given hot page x, the expected number of times that x is routed to the heat sink during W is $O(1)$.*
- *There are at most $\epsilon^{\omega(1)} n$ hot pages.*

Moreover, the event E is independent of the coin flips used by hot pages to decide whether to go to the heat sink during misses.

PROOF. We know from Lemma 11 that, with probability $1 - 1/n^{\omega(1)}$, the number of hot pages is at most $\epsilon^{\omega(1)} n$. We also know from Lemma 10 that, with probability $1 - 1/n^{\omega(1)}$, the total number of distinct cool pages that get sent to the heat sink during W is $O(\epsilon^2 n)$. It follows that, with probability $1 - 1/n^{\omega(1)}$, the total number of distinct pages that get sent to the heat sink during W is $O(\epsilon^2 n)$. Condition on these events (which occur together with probability $1 - 1/n^{\omega(1)}$) and further condition on the event that these hot pages are capable of residing simultaneously in the heat sink (this occurs with probability $1 - O(1/n)$ by Corollary 2 applied to a graph of size ϵn nodes with $O(\epsilon^2 n)$ edges). Then, we can apply Lemma 8 to deduce that each hot page is placed into the heat sink at most $O(1)$ expected times in W. As the events that we have conditioned on are independent of the heat-sink-coin-flips used by the hot pages, the proof of the lemma is complete. □

As an immediate corollary of the previous lemma (and conditioning on the event E from the lemma), the total expected number of times that hot pages experience misses and get sent to the heat sink (during W) is $\epsilon^{\omega(1)} n$. On the other hand, each time that a page experiences a miss, it has an ϵ^2 probability of being sent to the heat sink. Using this fact, we can recover a bound on the total expected number of times that hot pages experience misses during W (regardless of whether they are sent to the heat sink).

Lemma 13. *Conditioned on the random event E from Lemma 12, the expected number of misses that* HEAT-SINK LRU *incurs on hot pages during W is $\epsilon^{\omega(1)} n$.*

PROOF. Recall that, conditioned on E, there are at most $\epsilon^{\omega(1)} n$ hot pages, each of which expects to be sent to the heat sink $O(1)$ times. Let J be the total number of times that HEAT-SINK LRU incurs misses on these hot pages, and let K be the total number of times that HEAT-SINK LRU incurs misses on these hot pages and flips a coin that sends the hot page to the heat sink. Since conditioning on E does not reveal anything about the coin flips used for sending hot pages to the heat sink, we have that $\mathbb{E}[K] = \mathbb{E}[J] \cdot \epsilon^{-2}$. We have already shown that $\mathbb{E}[J] = \epsilon^{\omega(1)} n$, and it follows that $\mathbb{E}[K] \le \epsilon^{\omega(1)} n$ as well. □

At this point, we are ready to put all of the pieces together. During each phase, we will use Lemma 13 to analyze the misses incurred by hot pages. Combining this with Lemma 9, we can perform an accounting argument that proves Theorem 4.

PROOF OF THEOREM 4. Across all time, let $c_{0,1}$ the the number of accesses on which HEAT-SINK LRU misses but LRU hits; let $c_{0,0}$ be the number of accesses on which both miss; and let $c_{1,0}$ be the number of accesses on which HEAT-SINK LRU hits but LRU misses. Finally, let p be the total number of bonus points that are awarded, and let ℓ be the overall length of the access sequence.

First observe that

$$\mathbb{E}[p] = \epsilon^2\mathbb{E}[c_{0,1}] + \mathbb{E}[c_{1,0}],$$

because, in order for a bonus point to be created, either LRU misses and HEAT-SINK LRU hits (this happens $c_{1,0}$ times); or LRU hits, HEAT-SINK LRU misses, and we flip a coin that sends the missed page page to the HEAT-SINK (this happens $\epsilon^2\mathbb{E}[c_{0,1}]$ expected times).

Let C_{hot} be the total number of misses that HEAT-SINK LRU incurs on hot pages (i.e., pages that are hot during the phase in which the miss occurs), and let L_{LRU} be the total cost incurred by LRU. If there are q phases overall, then we can apply Lemma 13 to each phase to bound $\mathbb{E}[C_{\text{hot}}] \le \epsilon^{\omega(1)}n \cdot q + O(\ell/n)$. On the other hand, by definition, $C_{\text{LRU}} \ge (q-1)\cdot \epsilon n = \Omega(q\epsilon n)$. It follows that $\mathbb{E}[C_{\text{Hot}}] \le \epsilon^{\omega(1)}C_{\text{LRU}} + O(\ell/n)$.

On the other hand, Lemma 9 tells us that the total number of misses that HEAT-SINK LRU incurs on cool pages is at most $p + c_{0,0}$. It follows that the total number of misses C incurred by HEAT-SINK LRU satisfies

$$\mathbb{E}[C] \le \epsilon^{\omega(1)}C_{\text{LRU}} + \mathbb{E}[p] + c_{0,0}.$$

Plugging in our expression for $\mathbb{E}[p]$, we get that

$$\begin{aligned}\mathbb{E}[C] &\le \epsilon^{\omega(1)}C_{\text{LRU}} + \epsilon^2\mathbb{E}[c_{0,1}] + \mathbb{E}[c_{1,0}] + \mathbb{E}[c_{0,0}] + O(\ell/n)\\ &\le \epsilon^{\omega(1)}C_{\text{LRU}} + \epsilon^2\mathbb{E}[C] + C_{\text{LRU}} + O(\ell/n)\end{aligned}$$

Pulling the $\mathbb{E}[C]$ terms to one side, it follows that

$$\mathbb{E}[C] \le \epsilon^{\omega(1)}C_{\text{LRU}} + (1+\epsilon^2)C_{\text{LRU}} + O(\ell/n),$$

which completes the proof. □

6 CONCLUSION

LRU with low-associativity, which is commonly used in hardware caches, does not asymptotically match the performance of fully-associative LRU. By using randomness instead of LRU (such as with the policy 2-RANDOM), low-associativity caching algorithms can match the asymptotic performance of fully associative LRU.

We mix low-associativity LRU with 2-RANDOM to obtain HEAT-SINK LRU, an even better although more complex caching procedure. It makes decisions informed by the relative recency of pages and uses randomness to dissipate the "heat" when many pages contend for the same slot. We prove that HEAT-SINK LRU performs within a small constant factor of fully associative LRU.

Our algorithms suggest a new way of thinking about the design of low-associativity caches. That is, allowing even small amounts of randomness can cause "overheating" parts of the cache to naturally cool down and obtain asymptotically better performance.

Future work could determine whether semi-uniformity is necessary for the LRU lower bound. That is, whether or not the lower bound holds for all possible implementations of low-associativity LRU. We also leave to future work experiments determining how HEAT-SINK LRU and similar policies perform on real-world workloads or in hardware.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CCF-2423105, CCF-2420942, CCF-2247576, and CCF-2230742.

Michael Bender was supported in part by the John L. Hennessy Chaired Professorship.

Martín Farach-Colton was supported in part by the Leonard J. Shustek professorship.

William Kuszmaul was partially supported by a Harvard Rabin Postdoctoral Fellowship and by a Harvard FODSI fellowship under NSF grant DMS-2023528.

Evan West was supported in part by NSF under Grant NRT-HDR 2125295.

REFERENCES

[1] Andreas Abel and Jan Reineke. Reverse engineering of cache replacement policies in intel microprocessors and their evaluation. pages 141–142. IEEE, 3 2014.

[2] Michael A. Bender, Abhishek Bhattacharjee, Alex Conway, Martín Farach-Colton, Rob Johnson, Sudarsun Kannan, William Kuszmaul, Nirjhar Mukherjee, Don Porter, Guido Tagliavini, Janet Vorobyeva, and Evan West. Paging and the address-translation problem. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 105–117, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Michael A. Bender, Richard Cole, Erik D. Demaine, and Martin Farach-Colton. Scanning and traversing: Maintaining data for traversals in a memory hierarchy. In *ESA*, volume 2461 of *Lecture Notes in Computer Science*, pages 139–151. Springer, 2002.

[4] Michael A. Bender, Rathish Das, Martín Farach-Colton, and Guido Tagliavini. An associativity threshold phenomenon in set-associative caches, 2023.

[5] Mark Brehob, Richard Enbody, Eric Torng, and Stephen Wagner. On-line restricted caching. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 374–383, USA, 2001. Society for Industrial and Applied Mathematics.

[6] Samira Briongos, Pedro Malagón, José M Moya, and Thomas Eisenbarth. Reload+refresh: abusing cache replacement policies to perform stealthy cache attacks. USENIX Association, 2020.

[7] Niv Buchbinder, Shahar Chen, and Joseph (Seffi) Naor. Competitive algorithms for restricted caching and matroid caching. In Andreas S. Schulz and Dorothea Wagner, editors, *Proceedings of the 22nd Annual European Symposium on Algorithms (ESA)*, pages 209–221, Berlin, Heidelberg, September 2014. Springer-Verlag.

[8] A. Djordjalian. Minimally-skewed-associative caches. pages 100–107. IEEE Comput. Soc, 2002.

[9] Michael Drmota and Reinhard Kutzelnigg. A precise analysis of cuckoo hashing. *ACM Transactions on Algorithms*, 8:1–36, 4 2012.

[10] Krishnan Gosakan, Jaehyun Han, William Kuszmaul, Ibrahim N. Mubarek, Nirjhar Mukherjee, Karthik Sriram, Guido Tagliavini, Evan West, Michael A. Bender, Abhishek Bhattacharjee, Alex Conway, Martin Farach-Colton, Jayneel Gandhi, Rob Johnson, Sudarsun Kannan, and Donald E. Porter. Mosaic pages: Big tlb reach with small pages. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 433–448, New York, NY, USA, 2023. Association for Computing Machinery.

[11] Giovani Gracioli, Ahmed Alhammad, Renato Mancuso, Antônio Augusto Fröhlich, and Rodolfo Pellizzoni. A survey on cache management mechanisms for real-time embedded systems. *ACM Computing Surveys*, 48:1–36, 11 2015.

[12] Norman P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. *ACM SIGARCH Computer Architecture News*, 18, 1990.

[13] Swadhesh Kumar and P K Singh. An overview of modern cache memory and performance analysis of replacement policies. pages 210–214. IEEE, 3 2016.

[14] Colin McDiarmid. *On the method of bounded differences*, pages 148–188. Cambridge University Press, 8 1989.

[15] M. Mendel and Steven S. Seiden. Online companion caching. *Theoretical Computer Science*, 324(2–3):183–200, September 2004.

[16] Enoch Peserico. Online paging with arbitrary associativity. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 555–564, USA, 2003. Society for Industrial and Applied Mathematics.

[17] André Seznec. A case for two-way skewed-associative caches. *ACM SIGARCH Computer Architecture News*, 21:169–178, 5 1993.

[18] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 2 1985.

[19] Wenjie Xiong and Jakub Szefer. Leaking information through cache lru states. pages 139–152. IEEE, 2 2020.