A New Competitive Analysis of Randomized Caching

by

Ching Law

S.B., Massachusetts Institute of Technology (1998)

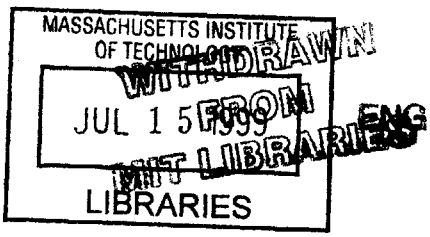
Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999



Author .....
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by .....
Charles E. Leiserson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A New Competitive Analysis of Randomized Caching

by

Ching Law

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 1999, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We provide new competitive upper bounds on the performance of the memoryless, randomized caching algorithm RANDOM. Our bounds are expressed in terms of the ***inherent hit rate*** α of the sequence of memory references, which is the highest possible hit rate that any algorithm can achieve on the sequence for a cache of a given size. Our results show that RANDOM is $(1 - \alpha e^{-1/\alpha})/(1 - \alpha)$-competitive on any reference sequence with inherent hit rate α. Since our new competitive bound does not scale up with the size k of the cache, it beats the putative $\Omega(\lg k)$ lower bound on the competitiveness of caching algorithms.

The research in this thesis represents joint work with Charles E. Leiserson.

Thesis Supervisor: Charles E. Leiserson
Title: Professor of Computer Science and Engineering

Acknowledgements

First and foremost, I would like to thank my advisor Professor Charles Leiserson. Charles motivated, supported, and guided me through the research presented in this thesis. I would like to thank him for his enormous patience with me, in helping to clarify my thoughts and writings. The research in this thesis represents joint work with Charles.

Thanks to all the members of the Supercomputing Technologies Group for giving me a stimulating and supportive environment. In particular, I would like to thank Professor Mingdong Feng, now of National University of Singapore, for supervising my undergraduate project in the group.

Thanks to Professors Tomas Lozano-Perez, Michael Sipser, Tom Leighton, Silvio Micali, and Mor Harchol-Balter for their assistance and inspiration in the past four years.

I would like to thank my teachers Tham Siew Thong, Tan Tiow Choo, Tai Thiam Hoo, and Joan Fong at Raffles Junior College in Singapore. Without their guidance, I probably would not have been admitted to MIT.

One of the most critical events of my life was to have been taught by Ernest Ying-Wai Kwong at St. Joan of Arc Secondary School in Hong Kong. I was first introduced to the Turing Test by Ernest during a grade 10 Chinese Language class. Ernest's influence on my intellectual development is unsurpassed.

My friend Kenneth Hon helped me a lot during my early years at MIT. We have many views in common about the way of studying. Kayi Lee has been my best partner through our adventures as Course 6 majors. My experience at MIT would have not been as good as it was if Kayi had chosen another college. Kayi read a draft of this thesis and provided valuable suggestions. I would also like to thank Hubert Chen, Derek Cheung, Xuxia Kuang, Kenneth Lau, Chris Leung, Jervis Lui, Kenneth Ng, Kenneth Pang, Albert Wong, and June Yiu. Each, in his or her unique way, has made my past four years at MIT memorable.

This research was supported by an IBM Cooperative Fellowship during the academic year 1998-1999. Thanks to Ashwini Nanda, my summer mentor at IBM Watson Research Center. I am also grateful to Professor Kai-Yeung Siu, for his patience with me when the completion of this thesis was delayed.

Finally, I would like to thank my parents and my sister Cindy for their love and encouragement. In particular, it was my father who started my interest in computers by teaching me Applesoft BASIC when I was in grade 5.

1 Introduction

Fiat *et al.* [11] show that no on-line cache replacement algorithm[1] on size-k caches can be better than H_k-competitive, where $H_k = \Theta(\lg k)$ is the kth harmonic number. Moreover, subsequent research [1, 23] has demonstrated the existence of H_k-competitive algorithms for this problem. Despite the apparent meeting of these upper and lower bounds, we show in this thesis that much better competitive upper bounds can be obtained.

Before discussing our new upper bounds, we first present some background definitions. A cache-replacement algorithm is said to be ***on-line*** if at each point in time, the algorithm responds to a memory request based only on past information and with no knowledge whatsoever about any future requests. An ***off-line*** cache-replacement algorithm, on the other hand, assumes the availability of an entire input sequence of memory requests. In this thesis, replacement algorithms will be denoted in Sans Serif font, for example, A, sometimes subscripted with the size of its cache, as in A_k. We say that an algorithm A_k ***ρ-competes*** with another algorithm B_h if the number of cache misses incurred by A_k on any input is at most ρ times the number of cache misses incurred by B_h. We say an algorithm A_k is ***ρ-competitive*** if A_k ρ-competes with OPT_k, where OPT_k is the optimal off-line algorithm.

Fiat *et al.*'s H_k lower bound for size-k caches uses an adversarial argument to construct a sequence of memory requests that causes a given randomized caching algorithm to be at least H_k-competitive on the sequence. Their construction produces sequences whose ***inherent miss rate*** β, the fraction of requests on which the optimal off-line algorithm OPT_k misses, is at most $1/k$. Consequently, for sequences of requests with $\beta > 1/k$, their argument provides no lower bound on how efficiently a caching algorithm can serve these sequences. Indeed, we show in this thesis that for a constant miss rate, an $O(1)$-competitive upper bound can be obtained by the memoryless, randomized caching algorithm RANDOM introduced by Raghavan and Snir [24].

As with Fiat *et al.*'s lower bound, previous upper bounds on the competitiveness of caching algorithms apply most aptly to low miss rates. For example, Raghavan and Snir's analysis of RANDOM, which shows that RANDOM_k is k-competitive, leads to a trivial upper bound for $\beta \geq 1/k$. Analysis of the least-recently used algorithm LRU [27] likewise shows that LRU_k is k-competitive, which is a trivial upper bound if $\beta \geq 1/k$. The $\mathsf{MARKING}_k$ algorithm [11] is $2H_k$-competitive, which offering trivial upper bounds for $\beta \geq 1/2H_k$; and the $\mathsf{PARTITION}_k$ [23] and $\mathsf{EQUITABLE}_k$ [1] algorithms are H_k-competitive, providing trivial upper bounds for $\beta \geq 1/H_k$.

In comparison, our new analysis of RANDOM provides nontrivial upper bounds for all $0 < \beta < 1$. In particular we show that RANDOM_k is $(1 - (1-\beta)e^{-1/(1-\beta)})/\beta$-competitive on request sequences with inherent miss rate β. This result, because of its derivation, is more naturally expressed in terms of the ***inherent hit rate***

[1] Fiat *et al.* actually discuss paging algorithms, instead of cache replacement algorithms, but the basic issues are the same. We use caching terminology, because our results are more appropriate for the domain of caches than for virtual memory.

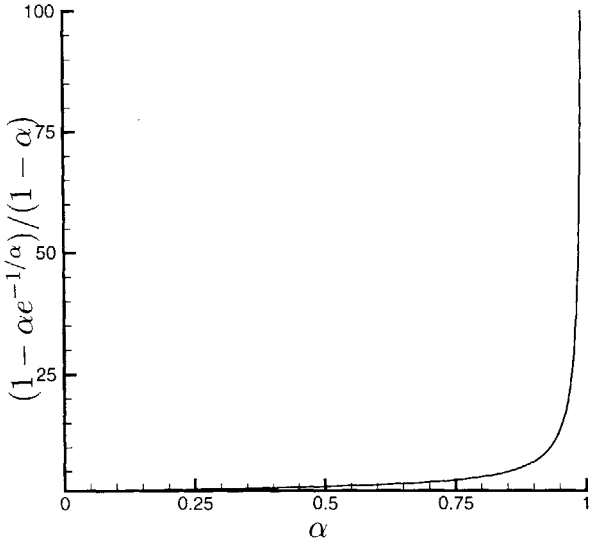


Figure 1: The graph of $(1 - \alpha e^{-1/\alpha})/(1 - \alpha)$ for $\alpha \in (0, 1)$.

Parameter	*L1 cache*	*L2 cache*	*Virtual Memory*
Block size (bytes)	12–128	32–256	4096–65,536
Hit time (cycles)	1–2	6–15	40–100
Miss time (cycles)	8–100	30–200	0.7×10^6–6×10^6
Miss rate (%)	0.5–10	15–30	0.00001–0.001
Size (bytes)	1–128K	0.25–16M	16M–8G

Figure 2: Typical ranges of parameters for cache and virtual-memory systems [14, p. 471].

$\alpha = 1 - \beta$; so RANDOM$_k$ is $(1 - \alpha e^{-1/\alpha})/(1 - \alpha)$-competitive on a request sequences with inherent hit rate α. Figure 1 graphs $(1-\alpha e^{-1/\alpha})/(1-\alpha)$ for $\alpha \in (0, 1)$. Although the competitive ratio approaches infinity as the inherent hit rate approaches 1, it is reasonably small for moderate hit rates. For example, when the inherent hit rate α is 90%, the competitive ratio $(1 - \alpha e^{-1/\alpha})/(1 - \alpha)$ is 7.04. Thus, for a 90% hit rate, our competitive bound for RANDOM is better than Raghavan and Snir's bound of k for any cache with size larger than 7.

Our new bounds do not subsume previous upper bounds, however. In particular, the previous bounds work well for miss rates in ranges that are normally associated with virtual-memory paging (and for which these algorithms were designed), whereas our new bounds are more applicable to the typical miss rates of certain hardware caches. Figure 2 shows typical parameters for level-1 (L1) caches, level-2 (L2) caches, and virtual memory. Figure 3 shows the previous upper bounds together with our new bounds for RANDOM. As can be seen from Figure 3, the bounds for PARTITION and EQUITABLE are the best to date for small inherent miss rates, while our new bounds for RANDOM are best for larger inherent miss rates. In particular, our new bound gives better numerical miss rate bounds than all previously known results for

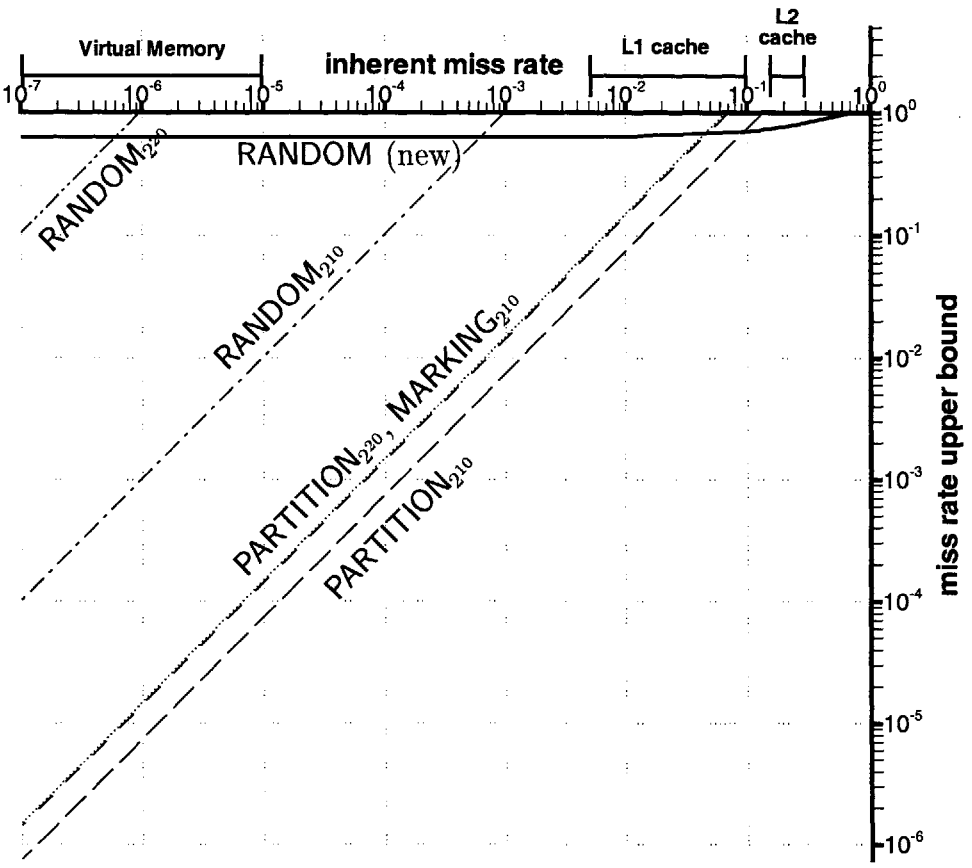


Figure 3: Upper bounds on the miss rate for various on-line algorithms as a function of inherent miss rate across a range from 10^{-7} to 1. Since cache size influences the previous bounds, the upper bounds for cache sizes of 2^{10} and 2^{20} are shown. The upper bound for our new analysis is labeled "RANDOM(new)," and as can be seen from the figure, is the first bound that applies across the entire range of inherent miss rates. The new bounds are stronger than any previous bounds for inherent miss rates of between 10% and 100%.

sequences with inherent miss rates larger than 10%.

The remainder of this thesis is organized as follows. Section 2 reviews related works on competitive analysis for caching algorithms. Section 3 states the essential definitions for describing replacement algorithms. Section 4 introduces a new framework for describing the adversary's strategy. Section 5 gives a a competitive ratio for RANDOM using a convexity argument based on the strategy framework. Section 6 concludes with a discussion of possible extensions to this work.

2 Related Work

In this section we review some competitive analysis results on on-line deterministic and randomized replacement algorithms. After discussing some criticisms and variations of this competitive analysis framework, we summarize with an observation that that previous lower bounds implicitly assume requests sequences with very low inherent miss rates.

Deterministic Algorithms

Competitive ratios of common deterministic replacement algorithms have been determined. Comparing on-line algorithms running on size-k caches with the optimal off-line algorithm running on size-h caches, where $h \leq k$, Sleator and Tarjan [27] showed that both LRU_k (Least-Recently-Used) and FIFO_k (First-In-First-Out) are $k/(k-h+1)$-competitive with OPT_h. They also showed that is the best possible competitive ratio for any deterministic algorithm. Karlin *et al.* [16] proved that FWF_k (Flush-When-Full) is also $k/(k-h+1)$-competitive with OPT_h.

Randomized Algorithms

Manasse *et al.* [21] extended competitive analysis to study randomized replacement algorithms, for which the expected miss rates are compared to the miss rates of OPT. In this thesis we will only consider ***oblivious adversaries*** [24], which generates a request sequence given only the description of the on-line algorithm, but not the random choices made by during the execution of the algorithm. Fiat *et al.* [11] showed that randomized replacement algorithms are at least H_k-competitive, where H_k is the kth harmonic number. They also gave a simple $2H_k$-competitive algorithm $\mathsf{MARKING}_k$. Algorithms $\mathsf{PARTITION}_k$ [23] and $\mathsf{EQUITABLE}_k$ [1] are found to be H_k-competitive. Considering memoryless algorithms, Borodin and El-Yaniv [4] showed that RANDOM_k $k/(k-h+1)$-competes with OPT_h. Moreover, Raghavan and Snir [24] demonstrated that no memoryless randomized algorithm can be better than k-competitive.

Since on-line algorithms are handicapped with imperfect information, it is useful to investigate how the competitive ratio improves if they are compensated with larger caches. Although such results are known for FIFO, LRU and RANDOM, the corresponding knowledge for other randomized algorithms is limited. Young [29] showed that any randomized algorithm A_k at most roughly $\ln(k/(k-h))$-competes with OPT_h when $k/(k-h) \geq e$. He also showed that $\mathsf{MARKING}_k$ roughly $2\ln(k/(k-h))$-competes with OPT_h under the same condition. It is not known in general how other randomized algorithms perform with varying cache sizes.

Criticisms and Variations

We have also seen some criticisms against competitive analysis as an evaluation tool for online algorithms. For example, Ben-David and Borodin [3] indicate that some

competitive algorithms require unbounded memory, and that finite lookahead is useless for improving the competitive ratio. They suggested the max/max ratio as an alternative measure for online algorithms. An online algorithm's max/max ratio is defined as its worst-case amortized cost over the worst-case amortized cost of the off-line algorithm. Dichterman [8] showed that the algorithm UNIFORM achieves the optimal max/max ratio, but is not competitive at all. (Moreover, strongly competitive randomized algorithms (PARTITION [23] and EQUITABLE [1]) appears to be too complex to be implemented in hardware.)

Some researchers have complained that the adversary is too powerful, thus leading to weak competitive ratios of on-line algorithms. Some have tried to curb the power of the adversary. For example, the access graph model [5, 15, 10] restricts the possible choices of the next request as the function of the current request, so as to model locality of reference. Some have tried to enhance the power of the on-line algorithm, for example, with lookaheads [28, 17].

Summary

Although algorithms matching the competitiveness lower bounds for both deterministic and randomized algorithms have been discovered, the results we have so far are still too weak to lend insights into the observed performances of various on-line algorithms. We, however, observe that the derivations of the lower bounds of competitiveness assume request sequences with extremely low miss rates. For example, in the Raghavan and Snir's proof [24] of the k-competitiveness lower bound for RANDOM_k, a request sequence with inherent miss rate $\beta < 1/mk$ is required to make RANDOM_k miss with a factor of $k(1-(1-1/k)^m)$ over OPT_k. Moreover, Fiat *et al.*'s construction [11] of the H_k-competitiveness lower bound for any randomized algorithms produces sequences whose inherent miss rate β is at most $1/k$. In addition, in the proof of the k-competitiveness lower bound for deterministic algorithms, Goemans [12] used a request sequence with inherent miss rate below $1/k$. In this thesis, we present a new approach to get around these lower bounds by using the inherent miss rate as a parameter.

3 Preliminaries

This section lays the framework for analyzing the RANDOM replacement algorithm. We introduce notation to describe the behavior of cache replacement algorithms, and we define precisely the notions of "cache hit" and "cache miss." Our framework loosely corresponds to the model presented by Coffman and Denning [7, pages 243-246].

We model a two-level memory system composed of a ***(primary) memory*** and a ***cache***. The memory is a set of ***(memory) locations***, where each location is a natural number $r \in \mathbb{N}$. The ***cache state*** $T \in \mathbb{N}^k$ of a size-k cache is a k-tuple of locations, where $T[s] = r$ if memory location r is stored in ***slot*** s of the cache. The special location 0 of memory is never referenced. Instead, we use 0 to denote an empty cache slot. The cache has additional ***control state*** Q in order to make

decisions about which locations it stores at any given time.

When a k-slot cache is required to serve a ***(memory) request*** $r \in \mathbb{N} - \{0\}$, a ***replacement algorithm*** A_k changes its existing cache state T and control state Q to a new cache state T' and control state Q'. Specifically, the replacement algorithm A_k chooses a ***replacement slot*** $\mathsf{A}_k(T, Q, r)$ such that

$$T'[s] = \begin{cases} r & \text{if } \mathsf{A}_k(T, Q, r) = s, \\ T[s] & \text{otherwise.} \end{cases}$$

Moreover, we require that if there exists a slot s such that $T[s] = r$, then $\mathsf{A}_k(T, Q, r) = s$. Thus, the replacement slot is the only slot whose contents may change. If the contents do not change, that is, if $r = T'[s] = T[s]$, then the request is a ***cache hit***. Otherwise, if $r = T'[s] \neq T[s]$, the request is a ***cache miss***.

We now define precisely the number of hits (or misses) that a replacement algorithm A_k incurs on a sequence $R = \langle r_1, r_2, \ldots, r_n \rangle$ of n requests. Let $T_1 = (0, 0, \ldots, 0)$ be the k-tuple representing the empty initial cache state, and let Q_1 be the initial control state. In order to service request r_i for $i = 1, 2, \ldots, n$, the cache algorithm A_k inductively changes cache state T_i and control state Q_i to cache state T_{i+1} and control state Q_{i+1}. Since A_k might be a randomized replacement algorithm, define $\text{hit}(\mathsf{A}_k, r_i)$ to be the event that request r_i is a hit, and overload the notation to define the indicator random variable

$$\text{hit}(\mathsf{A}_k, r_i) = \begin{cases} 1 & \text{if } r_i \text{ is a hit,} \\ 0 & \text{if } r_i \text{ is a miss.} \end{cases}$$

Let $\text{hit}(\mathsf{A}_k, R)$ be the total number of hits incurred by A_k over the entire sequence R, whence

$$\text{hit}(\mathsf{A}_k, R) = \sum_{i=1}^{n} \text{hit}(\mathsf{A}_k, r_i) \ .$$

Likewise, for misses define

$$\text{miss}(\mathsf{A}_k, r_i) = 1 - \text{hit}(\mathsf{A}_k, r_i)$$

for $i = 1, 2, \ldots, n$, and

$$\text{miss}(\mathsf{A}_k, R) = n - \text{hit}(\mathsf{A}_k, R) \ .$$

The focus of this thesis is the analysis of RANDOM, a simple, randomized replacement algorithm. RANDOM is "memoryless," meaning that its control state Q never changes. Suppose RANDOM_k is running on a cache of size k with a cache state $T \in \mathbb{N}^k$. For a request $r \in \mathbb{N} - \{0\}$ that causes a miss ($r \notin T$), the algorithm selects a slot to be replaced uniformly at random. That is, for $s = 1, 2, \ldots, k$, we have

$$\Pr\{\mathsf{RANDOM}_k(T, Q, r) = s\} = 1/k \ .$$

We shall compare RANDOM with OPT, the optimal, offline replacement algorithm [2] that achieves the minimum-possible number of misses on any given request sequence $R = \langle r_1, r_2, \ldots, r_n \rangle$. For each request r_i that causes a miss, the OPT algorithm omnisciently replaces the best possible slot to minimize misses. Specifically, let $f_i(r)$ be the ***forward distance*** of a sequence R defined by

$$f_i(r) = \begin{cases} d & \text{if } r_{i+d} \text{ is the first occurrence of } r \text{ in } \langle r_{i+1}, r_{i+2}, \ldots, r_n \rangle, \\ \infty & \text{if } r \text{ does not appear in } \langle r_{i+1}, r_{i+2}, \ldots, r_n \rangle. \end{cases}$$

For a cache miss $r_i \notin T_i$, the replacement algorithm OPT_h chooses the location in the cache whose forward distance is largest. That is, it chooses $s = \mathsf{OPT}_h(T_i, Q_i, r_i)$ if

$$f_i(T_i[s]) = \max_{1 \le j \le h} f_i(T_i[j]),$$

where ties are broken arbitrarily.

Belady [2] showed that for any request sequence R, the OPT algorithm minimizes the total number of misses.

4 The Adversary's Strategy

Before presenting our analytical results in Section 5, we first develop a framework for describing and analyzing the oblivious adversary. We define formally the notion of the adversary's "strategy," and prove a theorem that shows that specifying a strategy is equivalent to specifying a request sequence.

The power of the oblivious adversary lies in the selection of a request sequence $R = \langle r_1, r_2, \ldots, r_n \rangle$, which is then served by the optimal algorithm. As we shall see, however, the actual request locations in the adversary's strategy do not matter. What is important is the time (if ever) that a location is subsequently accessed. Thus, we shall adopt a representation for the adversary's strategy in which the adversary directly specifies for each time step which cache slot to use and whether the cache hits or misses.

In order to formalize the notion of a strategy, we first define some terminology. Cache behavior can be described by two sequences. A ***slot sequence*** $S = \langle s_1, s_2, \ldots, s_n \rangle$ is a sequence of positive integers such that at time $i = 1, \ldots, n$, a request is brought into slot s_i. We define $\text{slots}(S) = \{s_1, s_2, \ldots, s_n\}$ to be the set of slots actually referenced in the sequence. Since the adversary gains no advantage by omitting reference to a slot, we assume that that $\text{slots}(S) = \{1, \ldots, h\}$, where $h = |\text{slots}(S)|$. An ***outcome sequence*** $Z = \langle z_1, z_2, \ldots, z_n \rangle$ is a sequence of 0's and 1's such that the cache hits at time $i = 1, \ldots, n$ if and only if $z_i = 1$.

We shall often wish to refer to the last time that a particular slot was used. In particular, we let $\text{prev}(S, i)$ be p if $s_p = s_i$ and s_i does not appear in $\langle s_{p+1}, \ldots, s_{i-1} \rangle$. In other words, s_p is the last occurrence of slot s_i in $\langle s_1, \ldots, s_{i-1} \rangle$. If slot s_i does not appear in $\langle s_1, \ldots, s_{i-1} \rangle$, we let $\text{prev}(S, i) = \infty$.

We can now define the the notion of a strategy formally.

Definition 1. *A **strategy** is a pair (S, Z), where S is a slot sequence and Z is a hit sequence of the same length.*

As discussed in Section 3, given any request sequence R, the algorithm OPT determines the outcome and the slot to use at each time $i = 1, \ldots, n$. The following theorem shows that a request sequence can also be deduced from a slot sequence and an outcome sequence. Thus, designing an adversarial strategy is essentially equivalent to designing an adversarial request sequence.

Theorem 2. *For each strategy (S, Z), a request sequence $R = \langle r_1, \ldots, r_n \rangle$ exists such that*

1. *request r_i is a hit if and only if outcome $z_i = 1$, and*
2. *running $\mathsf{OPT}_{|slots(S)|}$ on request sequence R produces the slot sequence S.*

Proof. Let $S = \langle s_1, \ldots, s_n \rangle$, $Z = \langle z_1, \ldots, z_n \rangle$, and $h = |\text{slots}(S)|$. We shall show that a request sequence $R = \langle r_1, \ldots, r_n \rangle$ satisfying both conditions exists. We construct request sequence R by the inductive definition:

$$r_i = \begin{cases} i & \text{if } z_i = 0, \\ r_{\text{prev}(S,i)} & \text{if } z_i = 1. \end{cases} \tag{1}$$

The request sequence R is well defined, because the definition of $\text{prev}(S, i)$ depends on $\langle r_1, r_2, \ldots, r_{i-1} \rangle$ only.

We first show that Condition 1 holds. We observe that if $z_i = 0$, then i does not appear in $\langle r_1, \ldots, r_{i-1} \rangle$, and thus r_i is a miss. If $z_i = 1$, then $r_i = r_{\text{prev}(S,i)}$ is a hit, because slot s_i is not used between time $\text{prev}(S, i)$ and time i.

We next show that Condition 2 holds. We show that s_i is selected by OPT_h at any time $i = 1, 2, \ldots, n$.

- If $z_i = 1$, then by Condition 1, r_i is a hit. In this case, $r_i = r_{\text{prev}(S,i)}$, and thus OPT_h must select $s_{\text{prev}(S,i)}$. But, by the definition of prev, $s_{\text{prev}(S,i)} = s_i$, and therefore s_i is selected.

- If $z_i = 0$, then we shall show that OPT_h chooses slot s_i to replace. Recall that $s = \mathsf{OPT}_h(T_i, Q_i, r_i)$ if $f_i(T_i[s]) = \max_{1 \le j \le k} f_i(T_i[j])$, where the T_i are cache states and the Q_i are control states (both defined in Section 3). If we can show that $f_i(T_i[s_i]) = \infty$, that is, that location $T_i[s_i]$ does not appear in $\langle r_{i+1}, \ldots, r_n \rangle$, then OPT_h is free to choose $s = s_i$.

  First, if $T_i[s_i] = 0$ (the slot content is invalid), then by definition of Equation (1), $r_i \neq 0$ for all i. Therefore, $T_i[s_i]$ does not appear in $\langle r_{i+1}, \ldots, r_n \rangle$.

  Otherwise, if $T_i[s_i] \neq 0$, we show that $T_i[s_i] \neq r_j$ for $j = i + 1, \ldots, n$:

  - If $z_j = 0$, then

    $$r_j = j \neq T_i[s_i],$$

    because $T_i[s_i] = r_{\text{prev}(S,i)} < i < j$.

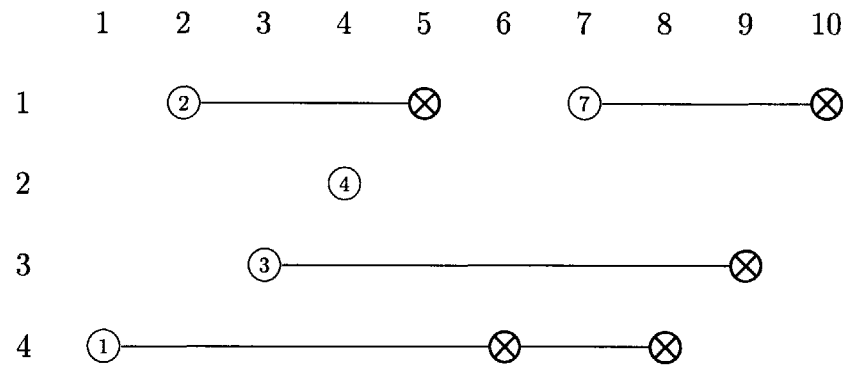


Figure 4: A diagram of the strategy (S, Z), where $S = \langle 4,1,3,2,1,4,1,4,3,1\rangle$ and $Z = \langle 0,0,0,0,1,1,0,1,1,1\rangle$. Time runs from left to right, and slot number from top to bottom. Cache hits are represented by $\otimes$; cache misses are represented by $\bigcirc$. The request sequence $\text{req}(S,Z) = \langle 1,2,3,4,2,1,7,1,3,7\rangle$ for the strategy is illustrated for a miss by the value in the corresponding circle and for a hit by the value in the miss circle at the start of the chain.

- If $z_j = 1$, there are two cases:
  - If $s_j \neq s_i$, then $r_j \neq T_i[s_i]$, because each location stays in one cache slot only.
  - If $s_j = s_i$, then $\text{prev}(S,j) \geq i$. But $T_i[s_i] = r_{\text{prev}(S,i)} < i$, and thus $r_j \neq T_i[s_i]$.

  Since $r_j \neq T_i[s_i]$ for $j = i+1,\ldots,n$, location $T_i[s_i]$ does not appear in $\langle r_{i+1},\ldots,r_n\rangle$.

□

We define the ***request sequence*** of a strategy (S, Z), denoted $\text{req}(S, Z)$, to be the request sequence R provided by Equation (1) of Theorem 2.

As an example, we consider a cache of 4 slots. We let the slot sequence S be $\langle 4,1,3,2,1,4,1,4,3,1\rangle$ and the hit sequence Z be $\langle 0,0,0,0,1,1,0,1,1,1\rangle$. Theorem 2 allows us to construct the request sequence $\text{req}(S,Z) = R = \langle 1,2,3,4,2,1,7,1,3,7\rangle$ such that S is an optimal slot sequence. The strategy (S, Z) is depicted in Figure 4.

5 Analysis of RANDOM

In this section, we derive a new lower bound on the expected number of hits of RANDOM on a request sequence. Specifically, we show that for any request sequence R with inherent hit rate α by OPT_h, the expected hit rate of RANDOM_k is at least $\alpha e^{-(h-1)/(k-1)\alpha}$.

We first outline our plan of proving the lower bound. Any request sequence R has a corresponding optimal strategy (S, Z) by the off-line algorithm OPT_h. According

to the outcome sequence Z, every request in R is either an inherent hit or an inherent miss. We can assume that the on-line algorithm RANDOM_k cannot hit on any inherent misses, because an adversary can, and should, request never-seen-before locations at the inherent misses. On the other hand, RANDOM_k, as a randomized algorithm, should have some chance of hitting on the inherent hits. Our goal is to establish a lower bound for the sum of these probabilities.

Our first step is to derive a bound on the expected number of hits by RANDOM on the inherent hits of a single slot. That is, each inherent hit at time $i = 1, \ldots, n$, is served by a slot $s_i \in \text{slots}(S)$. We focus on one of these slots. We shall later derive bounds for the whole strategy based on these single-slot results.

Lemma 3. *Let (S, Z) be an optimal strategy on a request sequence R with U inherent hits and V inherent misses. Consider any slot $s \in \text{slots}(S)$ having u inherent hits and v inherent misses. Let $t_1, \ldots, t_u$ be the times at which the u inherent hits of slot s occur. Then, there exist nonnegative integers $m_1, m_2, \ldots, m_n$ satisfying $\sum_{i=1}^{u} m_i \leq U + V - u - v$, such that*

$$\sum_{i=1}^{u} \Pr\{\mathit{hit}(\mathsf{RANDOM}_k, r_{t_i})\} \geq \sum_{i=1}^{u} (1 - 1/k)^{m_i}. \tag{2}$$

Proof. We first prove a lower bound on the probability of an individual inherent hit. Consider the ith inherent hit on slot s which occurs at time t_i. Since t_i is an inherent hit, we have $1 \leq \text{prev}(S, t_i) \leq t_i - 1$. If request r_{t_i} is location l, then request $r_{\text{prev}(S,t_i)}$ is the most recent request of l before time t_i. Let $m_i = t_i - \text{prev}(S, t_i) - 1$ be the number of requests between time t_i and time $\text{prev}(S, t_i)$, any of which might miss and cause location l to be evicted from slot s. In the worst case, RANDOM_k misses at all these m_i requests. Each miss evicts location l from the cache with a probability of $1/k$. In other words, the probability that l is not replaced by each request is at least $1 - 1/k$. Since the slot to replace at each miss is selected independently, the probability that location l remains in the cache to be hit at time t_i is at least $(1 - 1/k)^{m_i}$. In other words, $(1 - 1/k)^{m_i}$ is a lower bound of the probability that RANDOM_k hits on request r_{t_i}, the ith inherent hit of slot s:

$$\Pr\{\text{hit}(\mathsf{RANDOM}_k, r_{t_i})\} \geq (1 - 1/k)^{m_i}. \tag{3}$$

Summing Inequality (3) for $i = 1, \ldots, u$, we obtain

$$\sum_{i=1}^{u} \Pr\{\text{hit}(\mathsf{RANDOM}_k, r_{t_i})\} \geq \sum_{i=1}^{u} (1 - 1/k)^{m_i},$$

which is Inequality (2).

We still need to establish the constraints on the m_i's. Because $u + v$ requests are served by slot s, the total number of requests available (that is, not served by slot s) to interfere the inherent hits of slot s is at most $U + V - u - v$, that is, $\sum_i^u m_i \leq U + V - u - v$. Since $t_i > \text{prev}(S, t_i)$, each $m_i = t_i - \text{prev}(S, t_i) - 1$ is nonnegative. □

We next present a technical lemma providing a lower bound for the right-hand side of Inequality (2). We show that an expression of the form $\sum_{i=1}^{u} \gamma^{m_i}$, with a constraint on the $\sum m_i$, is minimized when the m_i's are evenly distributed.

Lemma 4. *Let $m_1, m_2, \ldots, m_u$ be nonnegative real numbers satisfying $\sum_{i=1}^{u} m_i \leq M$ for some nonnegative real number M. Then, for any real number $\gamma \in (0,1)$, we have*

$$\sum_{i=1}^{u} \gamma^{m_i} \geq u\gamma^{M/u}. \tag{4}$$

Proof. Let $\boldsymbol{m} = (m_1, m_2, \ldots, m_u)$. (We use boldface to denote points in a $\mathbb{R}^u$.) Let $f(\boldsymbol{m}) = \sum_{i=1}^{u} \gamma^{m_i}$ be a scalar function on the domain

$$D = \left\{ \boldsymbol{m} \in \mathbb{R}^u \,\middle|\, m_i \geq 0 \text{ for } i = 1, \ldots, u \text{ and } \sum_{i=1}^{u} m_i \leq M \right\}.$$

Our goal is to find the global minimum of $f(\boldsymbol{v})$ on domain D. We first show that any point $\boldsymbol{m}$ in which $m_a \neq m_b$ for some $1 \leq a, b \leq u$ cannot be a minimum point. Given any such point $\boldsymbol{m}$, we consider the point $\boldsymbol{m}' = (m'_1, m'_2, \ldots, m'_u)$ defined by

$$m'_i = \begin{cases} (m_a + m_b)/2 & \text{if } s \in \{a, b\}, \\ m_i & \text{otherwise.} \end{cases}$$

If $\boldsymbol{m} \in D$, then $\boldsymbol{m}' \in D$ as well, because

$$\begin{aligned} (m_a + m_b)/2 &\geq \min(m_a, m_b) \\ &\geq 0, \end{aligned}$$

and

$$\begin{aligned} \sum_{i=1}^{u} m'_i &= \sum_{i=1}^{u} m_i \\ &\leq M. \end{aligned}$$

We show that $f(\boldsymbol{m}') < f(\boldsymbol{m})$ as follows:

$$\begin{aligned} f(\boldsymbol{m}) - f(\boldsymbol{m}') &= \sum_{i=1}^{u} \gamma^{m_i} - \sum_{i=1}^{u} \gamma^{m'_i} \\ &= \gamma^{m_a} + \gamma^{m_b} - \gamma^{m'_a} - \gamma^{m'_b} \\ &= \gamma^{m_a} + \gamma^{m_b} - 2\gamma^{(m_a+m_b)/2} \\ &= (\gamma^{m_a/2} - \gamma^{m_b/2})^2 \\ &> 0, \end{aligned}$$

because $m_a \neq m_b$. Thus, $\boldsymbol{m}$ is not a minimum point.

The maximum-minimum existence theorem [25, page 260] states that a continuous scalar function f must have at least one global maximum point and at least one global minimum point on a compact domain.

We can now find the minimum point. Since domain D is compact, f has a global minimum point on D. We have shown that, however, $f(\boldsymbol{m})$ is not minimized at any point $\boldsymbol{m} = (m_1, \ldots, m_u)$ in which $m_a \neq m_b$ for some $1 \leq a, b \leq u$. Therefore, at the minimum point, we must have $m_1 = m_2 = \cdots = m_u$. And since $\sum_{i=1}^{u} m_i \leq M$, it follows that

$$\begin{aligned} m_i &= \left(\sum_{j=1}^{u} m_j \right) / u \\ &\leq M/u. \end{aligned}$$

Because $\gamma < 1$, we have

$$\gamma^{m_i} \geq \gamma^{M/u},$$

and thus

$$\sum_{i=1}^{u} \gamma^{m_i} \geq u\gamma^{M/u}.$$

□

Equipped with Lemmas 3 and 4 on the expected hit rate of RANDOM_k for a single slot, we can now consider all the slots of a strategy together. We can do so by summing up h instances of Inequality (2), with the constraints that the sum of inherent hits for all the slots equals to the total inherent hits for the entire strategy, and likewise for inherent misses. The following lemma formalizes this argument.

Lemma 5. *Let R be a request sequence. Let (S, Z) be the optimal off-line algorithm OPT_h's strategy on R. Let $u_s = \left|\{t \in \{1, \ldots, n\} \mid s_t = s, z_t = 1\}\right|$ be the number of inherent hits and $v_s = \left|\{ta \in \{1, \ldots, n\} \mid s_t = s, z_t = 0\}\right|$ be the number of inherent misses for each slot $s \in slots(S)$. Let $U = \sum_{s=1}^{h} u_s$ be the total number of inherent hits and $V = \sum_{s=1}^{h} v_s$ be the total number of inherent misses. Then, we have*

$$\mathrm{E}\left[hit(\mathsf{RANDOM}_k, R)\right] \geq \sum_{s=1}^{h} u_s (1 - 1/k)^{(U+V-u_s-v_s)/u_s}. \tag{5}$$

Proof. Let $n = U + V$ be the length of the sequence R. For each slot $s \in \mathrm{slots}(S)$, let $t_{s,1}, t_{s,2}, \ldots, t_{s,u_s}$ be times that the inherent hits of slot s occur. We then apply Inequality (2) of Lemma 3 to obtain

$$\sum_{i=1}^{u_s} \Pr\left\{\mathrm{hit}(\mathsf{RANDOM}_k, r_{t_{s,i}})\right\} \geq \sum_{i=1}^{u_s} (1 - 1/k)^{m_{s,i}}, \tag{6}$$

where $\sum_{i=1}^{u_s} m_{s,i} \leq U + V - u_s - v_s$ and $m_{s,i} \geq 0$ for $i = 1, \ldots, u_s$. For each slot $s \in \text{slots}(S)$, we can apply Lemma 4, with $\gamma = (1 - 1/k)$, on the right-hand side of Inequality (6) to obtain

$$\sum_{i=1}^{u_s} (1 - 1/k)^{m_{s,i}} \geq u_s (1 - 1/k)^{(U+V-u_s-v_s)/u_s}. \tag{7}$$

Combining Inequalities (6) and (7), we have

$$\sum_{i=1}^{u_s} \Pr\left\{\text{hit}(\mathsf{RANDOM}_k, r_{t_{s,i}})\right\} \geq u_s (1 - 1/k)^{(U+V-u_s-v_s)/u_s}. \tag{8}$$

Finally, we sum Inequality (8) over all slots in slots(S) to obtain Inequality (5):

$$\begin{aligned} \mathrm{E}\left[\text{hit}(\mathsf{RANDOM}_k, R)\right] &= \sum_{s=1}^{h} \sum_{i=1}^{u_s} \Pr\left\{\text{hit}(\mathsf{RANDOM}_k, r_{t_{s,i}})\right\} \\ &\geq \sum_{s=1}^{h} u_s (1 - 1/k)^{(U+V-u_s-v_s)/u_s}. \end{aligned}$$

□

We now derive a lower bound for the right-hand side of Inequality (5) under the constraints $\sum_{s=1}^{h} u_s = U$ and $\sum_{s=1}^{h} v_s = V$.

Lemma 6. *Let $h \geq 1$ be an integer and $\gamma \in (0, 1)$ be a real number, For any real numbers $u_1, u_2, \ldots, u_h$ satisfying $\sum_{s=1}^{h} u_s = U$, and real numbers $v_1, v_2, \ldots, v_h$ satisfying $\sum_{s=1}^{h} v_s = V$, we have*

$$\sum_{s=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s} \geq U \gamma^{(h-1)(U+V)/U}. \tag{9}$$

Proof. We first consider the case where $h = 1$. The left-hand side of Inequality (9) is $U\gamma^{(U+V-U-V)/U} = U$; and the right-hand side is $U\gamma^{(h-1)(U+V)/U} = U\gamma^0 = U$. Inequality (9) is satisfied. Therefore, we can assume $h > 1$ for the rest of the proof.

We assume that $u_1, \ldots, u_h$ are chosen with the condition that $\sum_{s=1}^{h} u_s = U$. For any choices of $u_1, \ldots, u_h$, we show that Inequality (9) is true for any real numbers $v_1, \ldots, v_h$ satisfying $\sum_{s=1}^{h} v_s = V$.

Let $\boldsymbol{v} = (v_1, v_2, \ldots, v_h)$ be a vector in $\mathbb{R}^h$. Let

$$f(\boldsymbol{v}) = \sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s}$$

be a scalar function on $\mathbb{R}^h$, and let

$$g(\boldsymbol{v}) = \sum_{s=1}^{h} v_s$$

be an auxiliary function on $\mathbb{R}^h$, which is used to constrain the domain of f. Our goal is to find the global minimum value of $f(\boldsymbol{v})$ on on the constrained domain $D = \{\boldsymbol{v} \mid g(\boldsymbol{v}) = V\}$.

To find the minimum points, we use the Lagrange Method [25, page 276]. By the constrained critical point theorem [25, page 277], if $\boldsymbol{v}$ is an extreme point in D, then $\boldsymbol{v}$ must be a constrained critical point. A point $\boldsymbol{v}$ is a constrained critical point if there is a scalar value λ such that

$$\nabla f|_{\boldsymbol{v}} = \lambda \nabla g|_{\boldsymbol{v}}, \tag{10}$$

or in scalar form as

$$\frac{\partial f}{\partial v_s} = \lambda \frac{\partial g}{\partial v_s} \tag{11}$$

for $s = 1, \ldots, h$. To solve Equation (11), we differentiate f and g with respect to each v_s to obtain

$$\frac{\partial f}{\partial v_s} = (\ln \gamma)\gamma^{(U+V-u_s-v_s)/u_s} \tag{12}$$

and

$$\frac{\partial g}{\partial v_s} = 1. \tag{13}$$

Substituting Equations (12) and (13) into Equation (11) yields

$$(\ln \gamma)\gamma^{(U+V-u_s-v_s)/u_s} = \lambda \tag{14}$$

for $s = 1, \ldots, h$.

We first consider the degenerative case where $\lambda = 0$. In this case, $\gamma^{1/u_s} = 0$ for $s = 1, \ldots, h$, which means that $u_s = 0$ for $s = 1, \ldots, h$, and thus $\sum_{s=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s} = 0$. Moreover, since $U = \sum_{s=1}^{h} u_s = 0$, we have $U\gamma^{(h-1)(U+V)/U} = 0$ as well. Therefore, Inequality (9) is satisfied. In the rest of the proof, we assume that $\lambda \neq 0$ and that $u_s \neq 0$ for $s = 1, \ldots, h$.

Let $\boldsymbol{v'} = (v'_1, \ldots, v'_h)$ be the unique solution when we solve for the v_s in Equation (14). We must demonstrate that critical point $\boldsymbol{v'}$ is a global minimum point. The following argument is inspired by an example due to Rogers [25, page 285]. When v_1 approaches $-\infty$, there exists some v_s, $s \neq 1$, that becomes large. Therefore, f becomes large, because $0 < \gamma < 1$ implies $\lim_{v_s \to \infty} \gamma^{(U+V-u_s-v_s)/u_s} = \infty$. Similarly, f becomes large when v_1 approaches ∞. Applying the minimum-maximum existence

theorem [25, page 260] to an appropriate finite closed interval for v_1, we conclude that f must have a minimum point on domain D. As $\boldsymbol{v}'$ is the only critical point, $\boldsymbol{v}'$ must be the unique global minimum point.

Since $f(\boldsymbol{v}')$ is the global minimum value, we can put a lower bound on the left-hand side of Inequality (9) as follows:

$$\begin{aligned}\sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s} &= f(\boldsymbol{v}) \\ &\geq f(\boldsymbol{v}') \\ &= \sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v'_s)/u_s}. \end{aligned} \tag{15}$$

We would like to find a lower bound for the right-hand side of Inequality (15). Since Equation (14) can be written as

$$(U+V-u_s-v'_s)/u_s = (\ln(\lambda/\ln\gamma))/\ln\gamma, \tag{16}$$

by letting $(\ln(\lambda/\ln\gamma))/\ln\gamma$, we obtain

$$(U+V-u_s-v'_s)/u_s = C. \tag{17}$$

Substituting C into Inequality (15), we obtain

$$\begin{aligned}\sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s} &\geq \sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v'_s)/u_s} \\ &= \sum_{i=1}^{h} u_s \gamma^{C}. \end{aligned} \tag{18}$$

To solve for C, we multiply both sides of Equation (17) by u_s to get

$$U+V-u_s-v'_s = u_s C. \tag{19}$$

Summing both sides of Equation (19) for $s = 1, \ldots, h$ yields

$$\sum_{s=1}^{h}(U+V-u_s-v'_s) = \sum_{s=1}^{h} u_s C.$$

Since $\sum_{s=1}^{h} u_s = U$ and $\sum_{s=1}^{h} v'_s = V$, we have

$$h(U+V) - U - V = UC.$$

We can now express C in terms of U, V, and h:

$$\begin{aligned} C &= (h(U+V) - U - V)/U \\ &= (h-1)(U+V)/U. \end{aligned} \tag{20}$$

Substituting C in Inequality (18) establishes Inequality (9) for any given $u_1, \ldots, u_h$:

$$\begin{aligned} \sum_{i=1}^{h} u_s \gamma^{(U+V-u_s-v_s)/u_s} &\geq \sum_{i=1}^{h} u_s \gamma^{C} \\ &= \sum_{s=1}^{h} u_s \gamma^{(h-1)(U+V)/U} \\ &= U\gamma^{(h-1)(U+V)/U}. \end{aligned}$$

□

Before we proceed to state and prove our theorem on the hit-rate lower bound of RANDOM_k, we require one more technical lemma, a simple inequality which shall be useful in the proof of the theorem. This lemma is inspired by an exercise problem in Leung [20, page 45].

Lemma 7. *For any $k > 1$, we have*

$$1 - 1/k > e^{-1/(k-1)}. \tag{21}$$

Proof. We first show that if $x \in (0,1)$, then

$$\ln x > (x-1)/x. \tag{22}$$

The Mean Value Theorem [26, page 108] states that if f is a real continuous function on $[a,b]$ which is differentiable in (a,b) then there is a point $y \in (a,b)$ at which

$$f(b) - f(a) = (b-a)f'(x).$$

The natural logarithm function "ln" is continuous on $[x,1]$ and is differentiable in $(x,1)$ for any $x \in (0,1)$:

$$\frac{d\ln t}{dt} = 1/t.$$

By the Mean Value Theorem, there is a point $y \in (x,1)$ at which

$$\ln 1 - \ln x = (1-x)(1/y).$$

Since $\ln 1 = 0$,

$$\ln x = (x-1)/y.$$

Because $y > x$ and $x - 1 < 0$, we have

$$(x-1)/y > (x-1)/x,$$

and thus

$$\begin{aligned} \ln x &= (x-1)/y \\ &> (x-1)/x. \end{aligned}$$

Let $x = 1 - 1/k$. If $k > 1$, it follows that $0 < x < 1$. Substituting $1 - 1/k$ in Inequality (22), we obtain

$$\begin{aligned} \ln(1-1/k) &> (1-(1-1/k))/(1-1/k) \\ &= (1/k)/(1-1/k) \\ &= -1/(k-1). \end{aligned} \tag{23}$$

By exponentiating both sides of Inequality (23), we obtain

$$\begin{aligned} 1 - 1/k &= e^{\ln(1-1/k)} \\ &> e^{-1/(k-1)}. \end{aligned}$$

□

Finally, armed with Lemmas 5, 6, and 7, we are ready to prove our main theorem, which gives a lower bound for the expected hit rate of RANDOM_k on the entire request sequence.

Theorem 8. *Let $h \geq 1$ and $k \geq 2$ be integers. Let $\alpha \in [0,1]$ be a real number. For any request sequence R of length n such that hit$(\mathsf{OPT}_h, R)/n \geq \alpha$, we have*

$$\mathrm{E}\left[\mathit{hit}(\mathsf{RANDOM}_k, R)\right]/n \geq \alpha e^{-(h-1)/(k-1)\alpha}. \tag{24}$$

Proof. Let (S, Z) be the optimal off-line algorithm OPT_h's strategy on request sequence R. Let $u_s = \left|\{t \in \{1,\ldots,n\} \mid s_t = s, z_t = 1\}\right|$ be the number of inherent hits and $v_s = \left|\{ta \in \{1,\ldots,n\} \mid s_t = s, z_t = 0\}\right|$ be the number of inherent misses for each slot $s \in \mathrm{slots}(S)$. Let $U = \sum_{s=1}^{h} u_s$ be the total number of inherent hits and $V = \sum_{s=1}^{h} v_s$ be the total number of inherent misses. Then by Lemma 5, we have the following lower bound:

$$\mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right] \geq \sum_{s=1}^{h} u_s (1-1/k)^{(U+V-u_s-v_s)/u_s}. \tag{25}$$

By Lemma 6 with $\gamma = 1 - 1/k$, we have

$$\sum_{s=1}^{h} u_s (1-1/k)^{(U+V-u_s-v_s)/u_s} \geq U(1-1/k)^{(h-1)(U+V)/U}. \tag{26}$$

Combining Inequalities (25) and (26), we obtain

$$\mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right] \geq U(1-1/k)^{(h-1)(U+V)/U}.$$

The sum of inherent misses and hits is the length of R, and thus $U+V=n$. Moreover, the inherent hit rate is $U/n = \mathrm{hit}(\mathsf{OPT}_h, R)/n \geq \alpha$. Consequently, we have

$$\begin{aligned}\mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right]/n &\geq U(1-1/k)^{(h-1)(U+V)/U}/n \\ &= (U/n)\cdot(1-1/k)^{(h-1)n/U} \\ &\geq \alpha(1-1/k)^{(h-1)/\alpha}.\end{aligned}$$

Since $k>1$, we can apply Lemma 7 to obtain

$$\begin{aligned}\mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right]/n &\geq \alpha(1-1/k)^{(h-1)/\alpha} \\ &> \alpha\left(e^{-1/(k-1)}\right)^{(h-1)/\alpha} \\ &= \alpha e^{-(h-1)/(k-1)\alpha},\end{aligned}$$

which is Inequality (24). □

Theorem 8 works for all cache sizes except when the on-line cache has one slot only, because Lemma 7, which is used in Theorem 8, does not apply when $k=1$. To extend our result to this case, one possible approach is to treat k as a real number and take the right-hand limit of k approaching 1 on the right-hand side of Inequality (24):

$$\lim_{k\to 1^+} \alpha e^{-(h-1)/(k-1)\alpha} = \begin{cases} 0 & \text{if } h>1, \\ 1 & \text{if } h=1. \end{cases} \tag{27}$$

We can also, however, study this special case independently and obtain a result which is consistent with Equation (27) and is valid for any replacement algorithm. In Theorem 9, we show that any replacement algorithm A_1 is competitive with OPT_1, but not with any OPT_h for $h \geq 2$.

Theorem 9. *Let* A *be a replacement algorithm, and let* $\alpha \in [0,1]$. *For any request sequence* R *of length* n *such that* $\mathit{hit}(\mathsf{OPT}_1, R)/n \geq \alpha$, *we have*

$$\mathrm{E}\left[\mathit{hit}(\mathsf{A}_1, R)\right]/n \geq \alpha. \tag{28}$$

But, there exists length n' *where* $\mathit{hit}(\mathsf{OPT}_h, R')/n' \geq \alpha$ *for some integer* $h \geq 2$, *such that*

$$\mathrm{E}\left[\mathit{hit}(\mathsf{A}_1, R')\right] = 0. \tag{29}$$

Proof. We shall first prove Inequality (28). In this case, the off-line cache contains only one slot. Consider any request sequence $R = \langle r_1, \ldots, r_n\rangle$. Consider any inherent hit r_i, $2 \leq i \leq n$, in R by OPT_1. Request r_{i-1} must be same as r_i because the off-line cache has only one slot. But A_1 will certainly hit on r_i too, as there is no request to

evict r_{i-1}. Therefore, A_1 hits on every inherent hit of R, thus

$$\begin{aligned}\mathrm{E}\left[\mathrm{hit}(\mathsf{A}_1, R)\right]/n &= \mathrm{hit}(\mathsf{OPT}_1, R)/n \\ &\geq \alpha,\end{aligned}$$

which is Inequality (28)

We give a sequence R' satisfying Equation (29) if $h \geq 2$. Let a and b be two distinct locations. Consider the sequence $R' = \langle abab \cdots abab \rangle$ with length $n' \geq 2/(1-\alpha)$. For any $h \geq 2$, we have

$$\begin{aligned}\mathrm{hit}(\mathsf{OPT}_h, R')/n' &= (n'-2)/n' \\ &= 1 - 2/n' \\ &\geq \alpha,\end{aligned}$$

because a and b can be put into two different slots. Nevertheless, $\mathrm{E}\left[\mathrm{hit}(\mathsf{A}_1, R')\right] = 0$ because no single-slot cache can hit on any request in R'. □

The following corollary, on which the "RANDOM (new)" plot in Figure 3 is based, rephrases Theorem 8 in terms of miss rates.

Corollary 10. *Let $h \geq 1$ and $k \geq 2$ be integers. Let $\beta \in [0,1]$ be a real number. For any request sequence R of length n such that $miss(\mathsf{OPT}_h, R)/n \leq \beta$, we have*

$$\mathrm{E}\left[miss(\mathsf{RANDOM}_k, R)\right]/n \leq 1 - (1-\beta)e^{-(h-1)/(k-1)(1-\beta)}.$$

Proof. Letting $\alpha = 1 - \beta$, we have

$$\begin{aligned}\mathrm{hit}(\mathsf{OPT}_h, R)/n &= 1 - \mathrm{miss}(\mathsf{OPT}_h, R)/n \\ &\geq 1 - \beta \\ &= \alpha.\end{aligned}$$

From Theorem 8, we have

$$\mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right]/n \geq \alpha e^{-(h-1)/(k-1)\alpha}.$$

Thus, it follows that the expected miss rate of RANDOM_k can be bounded above:

$$\begin{aligned}\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_k, R)\right]/n &= (n - \mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right])/n \\ &= 1 - \mathrm{E}\left[\mathrm{hit}(\mathsf{RANDOM}_k, R)\right]/n \\ &\leq 1 - \alpha e^{-(h-1)/(k-1)\alpha}. \qquad (30)\end{aligned}$$

Substituting $1 - \beta$ in Inequality (30) yields

$$\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_k, R)\right]/n \leq 1 - (1-\beta)e^{-(h-1)/(k-1)(1-\beta)}.$$

□

We note that Corollary 10, in contrast to previously known competitive analysis results, always provides meaningful miss-rate upper bounds.

To facilitate comparisons with previous results in competitive analysis, we can express Theorem 8 in terms of competitive ratio with respect to misses for the special case where the on-line and off-line caches have the same size.

Corollary 11. *Let $k \geq 1$, $h \geq 1$ be integers, and $\alpha \in [0,1)$ be a real number. On any request sequence R of length n such that hit$(\mathsf{OPT}_k, R)/n = \alpha$, RANDOM_k is $(1 - \alpha e^{-1/\alpha})/(1-\alpha)$-competitive.*

Proof. To show that RANDOM_k is $(1 - \alpha e^{-1/\alpha})/(1-\alpha)$-competitive means showing that

$$\frac{\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_k, R)\right]}{\mathrm{miss}(\mathsf{OPT}_k, R)} \leq \frac{1 - \alpha e^{-1/\alpha}}{1-\alpha}. \tag{31}$$

We first consider the case of $k = 1$. Since $e^{-1/\alpha} < 1$, we have $1 - \alpha e^{-1/\alpha} > 1 - \alpha$, and thus

$$\frac{1 - \alpha e^{-1/\alpha}}{1-\alpha} \geq 1. \tag{32}$$

Theorem 9 shows that

$$\frac{\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_1, R)\right]}{\mathrm{miss}(\mathsf{OPT}_1, R)} = 1. \tag{33}$$

By combining Inequality (32) and Equation (33), Inequality (31) is satisfied for $k = 1$.

In the case that $k > 1$, we can apply Inequality (30) of Corollary 10 with $h = k$ to obtain Inequality (31) as follows:

$$\begin{aligned}
\frac{\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_k, R)\right]}{\mathrm{miss}(\mathsf{OPT}_k, R)} &= \frac{\mathrm{E}\left[\mathrm{miss}(\mathsf{RANDOM}_k, R)\right]/n}{\mathrm{miss}(\mathsf{OPT}_k, R)/n} \\
&\leq \frac{1 - \alpha e^{-1(k-1)/(k-1)\alpha}}{1-\alpha} \\
&\leq \frac{1 - \alpha e^{-1/\alpha}}{1-\alpha}.
\end{aligned}$$

□

6 Conclusions

The contributions of this thesis are, first, a new framework for describing the oblivious adversary strategy (Section 4) and, secondly, a competitive analysis conditioned on the inherent hit rate of a request sequence (Section 5). The analysis we have developed gives better numerical miss-rate bounds than all previously known results

for sequences with inherent miss rates larger than 10%. Our ratio is the first competitive bound of an on-line algorithm that does not deteriorate with increasing cache size. This result answers the question posed by Young [29] asking whether "we can effectively show constant competitiveness (independent of cache size) for any on-line paging strategy." In particular, we note that this is the first result to remain valid when the on-line cache is smaller than the oblivious adversary's off-line cache.

We predict that the competitive ratio derived in Section 5 can be further improved. In the proof of Theorem 8, we assumed cautiously that RANDOM always misses on the inherent hits. When the inherent miss rate is low, however, RANDOM has a reasonably low probability of missing the inherent hits. A proper utilization of this information might lead to a stronger bound of the expected total number of hits. We are currently pursuing this line of research.

Finally, we are hopeful that the techniques presented in this thesis can be applied to analyses of other on-line algorithms and be generalized to other on-line computational models, such as k-servers [21, 18, 19, 22, 13, 9] and metrical task systems [6].

References

[1] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. In *Annual European Symposium on Algorithms*, 1996.

[2] L. A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5(2):78–101, 1966.

[3] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11:73–91, 1994.

[4] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[5] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *Journal of Computer and System Sciences*, 50(2):244–258, April 1995.

[6] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, October 1992.

[7] Edward G. Coffman, Jr and Peter J. Denning. *Operating Systems Theory.* Prentice Hall, 1973.

[8] E. Dichterman. Randomized paging algorithms and measures for their performance. Technical Report 692, Haifa, Israel, October 1991.

[9] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *31st Annual Symposium on Foundations of Computer Science*, pages 454–463, 1990.

[10] Amos Fiat and Anna R. Karlin. Randomized and multipointer paging with locality of reference. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 626–634, Las Vegas, Nevada, 29 May–1 June 1995.

[11] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel D. Sleator, and Neal E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, December 1991.

[12] Michel X. Goemans. Advanced algorithms lecture notes, September 1994.

[13] E. F. Grove. The harmonic online K-server algorithm is competitive. In Lyle A. McGeoch and Daniel D. Sleator, editors, *On-line Algorithms*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 65–77. AMS/ACM, February 1991.

[14] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufman, second edition, 1996.

[15] Sandy Irani, Anna R. Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, June 1996.

[16] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel Dominic Sleator. Competitive snoopy caching. *Algorithmica*, 3:77–119, 1988.

[17] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. In *35th Annual Symposium on Foundations of Computer Science*, pages 394–400, Santa Fe, New Mexico, 20–22 November 1994. IEEE.

[18] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, September 1995.

[19] Elias Koutsoupias and Christos H. Papadimitriou. The 2-evader problem. *Information Processing Letters*, 57(5):249–252, 11 March 1996.

[20] Yuk-Lit Leung. *Methods of A-Level Pure Mathematics*, volume 2. Goodman Publisher, 1990.

[21] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 322–333, Chicago, Illinois, 2–4 May 1988.

[22] Lyle A. McGeoch. *Algorithms for Two Graph Problems: Computing Maximum-Genus Imbeddings and the Two-Server Problem.* PhD thesis, Carnegie Mellon University, Pittsburgh, 1987.

[23] Lyle A. McGeoch and Daniel D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[24] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. *IBM Journal of Research and Development*, 38(6):683–707, November 1994.

[25] Hartley Rogers. *Multivariable Calculus with Vectors.* Prentice Hall, May 1998.

[26] Walter Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, third edition, 1976.

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

[28] Eric Torng. A unified analysis of paging and caching. In *36th Annual Symposium on Foundations of Computer Science*, pages 194–203, Milwaukee, Wisconsin, 23–25 October 1995. IEEE.

[29] Neal Young. On-line caching as cache size varies. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–250, San Francisco, California, 28–30 January 1991.