Probabilistic Methods for Web Caching*

David Starobinski †
Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215
staro@bu.edu

David Tse
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720
dtse@eecs.berkeley.edu

Abstract

We introduce and analyze new randomized policies for the management of web caches. The proposed policies are fully scalable, that is, handling a hit or an eviction requires only $O(1)$ time. Our analysis is probabilistic, in nature, and based on an extended version of the IR model. The extension is needed in order to deal with the varying-cost and varying-size features of web documents. Under this assumption, we derive closed-form expressions for the stationary probabilities of finding each possible arrangement of the documents within the cache. Our analysis shows that the performance of the proposed algorithms are close to that of the optimal off-line algorithm. Using simulations and real traces, we also show that the new algorithms perform at least as well as existing algorithms of higher complexity. Variations on the algorithms, aimed at increasing their responsiveness to non-stationary trends, are also investigated.

*This research was partially supported by NSF grant ANI-9872764.

†Corresponding author. The research of this author was performed at the University of California Berkeley, under a fellowship for prospective researchers from the Swiss National Science Foundation.

1 Introduction

Web caches play a prominent part in improving the performance of web services [CaI97, AWY99]. By storing frequently requested documents in high-speed memories, they allow significant reduction in bandwidth consumption and response latencies.

Caching can be implemented at various points in the network, as shown in Fig. 1. At the lowest level, caches are implemented in Web browsers where they store the most recently documents accessed by a (single) client. At a higher level, caches are often implemented in *proxies*. Proxies, usually located near gateways, act as transparent servers on behalf of a large community of clients such as a corporation or an ISP. Finally, at the highest level, caches may be implemented in front of the web servers themselves. Such caches are referred to as reverse proxies or httpd accelerators.

Web caches distinguish themselves from CPU caches in at least two key aspects. First, web documents vary significantly in size. If objects are requested with equal frequency, then the hit ratio, that is, the proportion of requests served by the cache, is maximized when the caching algorithms are biased towards the smaller objects. This is because a cache is able to store a larger number of small objects. Second, web documents have different access costs. For example, the transfer time cost for larger documents is higher. Similarly, the cost of an object requested from an overloaded or distant server is typically more significant that of one requested from an underloaded or nearby server.

Replacement policies for web caches have been been subject of extensive research in the literature. In particular, a competitive algorithm [BoE98], referred to as GreedyDual-Size (GDS), was introduced in [CaI97] and shown to outperform previously proposed algorithms. Refinements of the GDS policy have been proposed in [DiA99, JiB00].

Beside hit ratio, caching systems are typically evaluated according to the following three met-

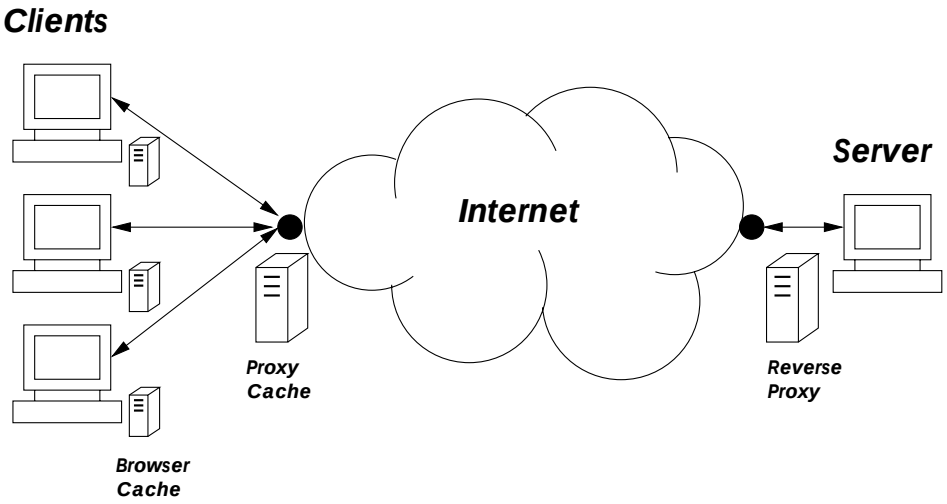

Figure 1: Cache levels

rics: speed, scalability, and reliability [BaO00]. Speed and scalability are of significant importance, since proxy servers frequently have to deal with very high load. In fact, the performance bottleneck of a proxy cache more often lies in its access speed rather than memory size. The policies developed in [CaI97] are not fully scalable since a hit or a replacement requires $O(\log K)$ time, where K corresponds to the cache size. Moreover, these policies need a fixed amount of metadata for each object, in order to store the object's value (or priority). Resorting to metadata is generally undesirable, from both storage and reliability points of view.

In this work, we develop and analyze new management techniques for web caches. The proposed policies are fully scalable, that is, a hit or a replacement requires only $O(1)$ time. This salient property is achieved by judicious use of randomization. In addition, our algorithms do not suffer from metadata overhead since the value of a document is represented by its position within the cache. Our work naturally complements other recent research [PsP01] that advocates use of randomization techniques in order to cope with scalability problems.

This paper is devoted to a presentation of the new algorithms and their performance evaluation. Our analysis is probabilistic in nature and based on the Independent Reference (IR) model. In

Section 2, we introduce the IR model, justify its use in the context of web proxy caches, and briefly review some known results related to the analysis of caching replacement algorithms under the IR model. Next, in Section 3, we augment the IR model in order to include the varying-size and varying-cost features of web documents. We present the new caching algorithms, and derive *closed-form* expressions for the probability of finding each possible arrangement of pages within the cache. Our analysis shows that the proposed algorithms achieved performances close to those of the optimal off-line algorithm. In Section 4, we investigate variations on the algorithms aimed at speeding-up their rate of convergence to steady-state. Then, in Section 5, we show through numerical simulations and real traces experiments that the new algorithms perform at least as well as existing algorithms of higher complexity. The last section is devoted to concluding remarks.

2 The Classical IR Model and Caching Algorithms

2.1 The Independent Reference Model

Let N denote the total number of web pages. Under the Independent Reference model, one assumes that, independently of previous requests, page i, $i = 1, 2, \ldots, N$, is requested with probability p_i. For convenience, and without any loss of generality, we rank the pages in the decreasing order of their popularity, i.e., if $i < j$ then $p_i \geq p_j$. Note that several experimental studies have shown that the popularity of web pages follows a Zipf distribution [Bre99], i.e., p_i is proportional to $1/i^{\alpha}$ where α is some positive constant. Nevertheless, in the sequel, we do not assume any specific distributions for the the popularity of web pages.

Although the pattern of page requests from a particular client typically exhibits a strong temporal locality (or temporal correlation), the IR model is certainly a reasonable *qualitative* model for representing accesses to proxy and reverse proxy caches. This is because proxy caches serve a large community of clients and therefore mitigate the temporal locality of requests from each par-

ticular client. Moreover, a significant amount of the temporal locality is absorbed by the browser caches of the clients. Experimental studies show indeed that proxy caches are mostly useful for objects requested by one client and later retrieved by another client [Dus97]. Other studies show that predictions based on the IR model are in excellent agreement with the qualitative behavior of proxy caches [Bre99].

2.2 Performance of Classical Caching Algorithms

In this section, we shortly review known results related to the performance of caching algorithms under the IR model. The seminal work in this area appeared in [Kin71, Gel73]. A comprehensive survey of literature results can be found in [ACK87].

We consider a cache memory with a capacity storage of K pages, where $K < N$. All the pages are assumed to have the same size and access cost. Under the IR model, if the request probabilities are known a priori, then the optimal algorithm OPT is to place the K most popular pages $i = 1, 2, \ldots, K$ into the cache. In practice, the popularity ranking among the pages is unknown *a priori* and practical caching algorithms need to learn this ranking. Such caching algorithms are referred to as *self-organizing* algorithms.

A straightforward approach for learning the popularity ranking is simply to count the number of requests for each of the N pages. This approach, known as the COUNT algorithm (or perfect-LFU [Bre99]), will converge to the true ranking as the size of the sample grows. One of the obvious disadvantages of this method is the major overhead required for implementing a separate counter for each of the N pages. Another problem of this algorithm in practice is that it reacts very slowly to any drift in the popularity of pages and may therefore lead to "cache pollution."

Several self-organizing algorithms are available to keep the cache in near-optimal order without resorting to counters. The most well-known is LRU, which, upon a request for a page not stored in

the cache, replaces the least recently used document. Another algorithm is CLIMB. This algorithm views the cache memory as a stack. Each time a page is requested, it is exchanged with the immediately preceding page; if the requested page is at the head of the stack nothing is done. Upon a request for a page not stored in the cache, CLIMB replaces the document positioned at the bottom of the stack. The time complexity, per access, of LRU and CLIMB is $O(1)$.

Under the IR model, the dynamics of the LRU and CLIMB algorithms can both described by an ergodic Markov chain. As a consequence, there exist unique stationary probabilities of finding the cache in each possible state $\vec{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$. Each state corresponds to a particular arrangement of K distinct pages within the cache. The pages σ_i should be viewed as stacked, with σ_1 at the top and σ_K at the bottom. We denote by $\pi(\vec{\sigma})$ the steady-state probability of finding the cache in some state $\vec{\sigma}$. Simple closed-form expressions for this quantity are available (see [ACK87], Section 4.6):

$$\pi^{\mathrm{LRU}}(\vec{\sigma}) = \prod_{i=1}^{K} \left(\frac{p_{\sigma_i}}{1 - \sum_{j=1}^{i-1} p_{\sigma_j}} \right) , \tag{1}$$

$$\pi^{\mathrm{CLIMB}}(\vec{\sigma}) = C_1 \cdot \prod_{i=1}^{K} (p_{\sigma_i})^{K-i+1} , \tag{2}$$

where C_1 is a normalization constant. The self-organizing property of the algorithms is illustrated from eqs. (1) and (2) by the fact that the state $\vec{\sigma} = \{1, 2, \ldots, K\}$ is the most likely among all the possible states.

The hit ratio is obtained by summing over all the indices i, the probability that page i is requested and found in the cache, i.e.,

$$H.R. = \sum_{i=1}^{N} p_i \sum_{i \in \vec{\sigma}} \pi(\vec{\sigma}). \tag{3}$$

The CLIMB algorithm is known to achieve higher hit ratio than LRU, although this result does

not seem to have been formally proven. The key reason for the success of LRU, in practice, is that it is much more responsive to non-stationary trends. This important property can also been observed with the IR model, where the convergence rate to steady-state of LRU is much faster than CLIMB [HeH85].

Another important property of LRU is that it can never perform significantly worse than OPT. Namely, for arbitrary $p_1, p_2, \ldots, p_N$, the ratio between the optimal hit ratio and the hit ratio of LRU is bounded [FrW74].

3 New Web Caching Algorithms

3.1 Extending the Independent Reference Model

The IR model has to be extended in order to address the non-uniform access cost and non-uniform size features of web documents. For simplicity, we present separately algorithms for each of these two issues. However, the combination of the algorithms is straightforward, as pointed out at the end of Section 3.3.

Let first assume that each page i has an access cost c_i. Under the IR model, the expected cost is minimized by placing in the cache the K documents with the largest $p_i c_i$ values. This optimal off-line policy will be referred to as OPT-C, where C stands for cost. In the next section, we devise two new on-line randomized algorithms, termed LRU-C and CLIMB-C, which tend to keep the cache in an ordering similar to OPT-C. We show that the performances of LRU-C and CLIMB-C with respect to OPT-C are identical to those of LRU and CLIMB with respect to OPT. From this point of view, LRU-C and CLIMB-C are dual to LRU and CLIMB.

Next, let s_i be the size (in bytes) of page i. The capacity storage of the cache is K bytes. The number of documents that the cache can accommodate depends, of course, on the respective sizes of

the documents. The off-line problem of finding which documents should be placed in the cache, in order to maximize the hit ratio, is known as the Knapsack problem which is NP-Complete [MoS91]. A simple greedy approximation is to fill the cache with the documents having the highest "density" values p_i/s_i. This policy is known to perform at most twice worse than the optimal solution, except for pathological case (see [MoS91], p.265). We refer to this sub-optimal policy, as OPT-S (S standing for size). In Subsection 3.3, we devise self-organizing algorithms, LRU-S and CLIMB-S, which learn to order the cache similarly to OPT-S. Once again, a duality property prevails between the new algorithms and the classical LRU and CLIMB algorithms.

3.2 Caching Algorithms for Documents with Different Access Costs

We define $c_{max} = \max(c_1, c_2, \ldots, c_N)$ as being the maximal access cost among all the N pages. Further, we define the normalized access costs as $\tilde{c}_i = c_i/c_{max}$. The algorithm LRU-C performs as follows. When a page i is requested, it is moved to the head of the cache with probability $\tilde{c}_i$; otherwise, nothing is done. Similarly, upon a request for page i, the CLIMB-C algorithm acts as CLIMB with probability $\tilde{c}_i$, and the state of the cache is left unmodified with a complementary probability $1 - \tilde{c}_i$.

The rationale behind these new algorithms is very simple. Let's consider two pages with respective access costs $c_1 = 10$ and $c_2 = 1$. It is valuable to place page 2 in the cache, instead of page 1, only if it is requested at least 10 times more frequently. In other words, a hit for page 2 is worth one tenth of a hit for page 1. By introducing randomization, each hit is given its appropriate weight.

The newly proposed caching algorithms lend themselves to a performance analysis. Using the same notation as in Section 2.2, the stationary probabilities of the Markov chains describing the LRU-C and CLIMB-C algorithms possess the following closed-form expressions:

Theorem 1

$$\pi^{LRU\text{-}C}(\vec{\sigma}) = \prod_{i=1}^{K} \left(\frac{p_{\sigma_i} c_{\sigma_i}}{\sum_{j=1}^{N} p_j c_j - \sum_{j=1}^{i-1} p_{\sigma_j} c_{\sigma_j}} \right) , \tag{4}$$

$$\pi^{CLIMB\text{-}C}(\vec{\sigma}) = C_2 \cdot \prod_{i=1}^{K} (p_{\sigma_i} c_{\sigma_i})^{K-i+1} , \tag{5}$$

where C_2 correspond to normalization constant.

Proof: (i) LRU-C: we prove eq. (4) by using a probabilistic argument. The lines of our reasoning follow [Hen76]. We look at the present state $\vec{\sigma}$ and attempt to reconstruct the past history by looking backwards in time. In the following, we refer to a request to a page as successful, if the page is subsequently moved to the head of the cache (which, for page i, happens with probability $\tilde{c}_i$); otherwise the request is referred to as unsuccessful.

In order for the cache to be in state $\vec{\sigma}$, the past history of requests, listed in reverse order from most remote to most recent, must be as follows:

2K. A last successful request for page σ_K is made.

2K-1. Requests (successful or unsuccessful) for pages $\{\sigma_1, \sigma_2, \ldots, \sigma_{K-1}\}$ or unsuccessful requests for any other pages may be made.

2K-2. A last successful request for page σ_{K-1} is made.

2K-3. Requests (successful or unsuccessful) for pages $\{\sigma_1, \sigma_2, \ldots, \sigma_{K-2}\}$ or unsuccessful requests for any other pages may be made.

...

4. A last successful request for page σ_2 is made.

3. Requests (successful or unsuccessful) for page σ_1 or unsuccessful requests for any other pages may be made.

2. A last successful request for page σ_1 is made.

1. Unsuccessful requests for all the pages may be made.

The probability of step $2i$ $(1 \leq i \leq K)$ is given by $p_{\sigma_i}\tilde{c}_{\sigma_i}$. The probability of steps $2i$ and $2i-1$, together, is given by

$$p_{\sigma_i}\tilde{c}_{\sigma_i} \cdot \sum_{n=0}^{\infty} \left(\sum_{j=1}^{i-1} p_{\sigma_j} + \sum_{j=1,\, j\neq\{\sigma_1,\sigma_2,\ldots,\sigma_{i-1}\}}^{N} (1-\tilde{c}_j)p_j \right)^n = \frac{p_{\sigma_i}\tilde{c}_{\sigma_i}}{\sum_{j=1}^{N} p_j\tilde{c}_j - \sum_{j=1}^{i-1} p_{\sigma_j}\tilde{c}_{\sigma_j}}.$$

The joint probability of step 1 up to step $2K$, is thus

$$\prod_{i=1}^{K} \frac{p_{\sigma_i}\tilde{c}_{\sigma_i}}{\sum_{j=1}^{N} p_j\tilde{c}_j - \sum_{j=1}^{i-1} p_{\sigma_j}\tilde{c}_{\sigma_j}} = \prod_{i=1}^{K} \left(\frac{p_{\sigma_i}c_{\sigma_i}}{\sum_{j=1}^{N} p_j c_j - \sum_{j=1}^{i-1} p_{\sigma_j}c_{\sigma_j}} \right),$$

which establishes the result.

(ii) CLIMB-C: Our approach is to show that the expressions given by eq. (5) satisfies the balance equations. The proof follows from the uniqueness of the stationary probabilities.

We denote by $q(\vec{\sigma}, \vec{\sigma}')$, the transition probability from state $\vec{\sigma}$ to state $\vec{\sigma}'$. We show next that, for this Markov chain, the following equations hold

$$q(\vec{\sigma}, \vec{\sigma}')\pi(\vec{\sigma}) = q(\vec{\sigma}', \vec{\sigma})\pi(\vec{\sigma}') \qquad \forall \vec{\sigma}, \vec{\sigma}'. \tag{6}$$

These equations are known as *detailed balanced equations*, and a Markov chain satisfying them is termed *reversible* [Kel79]. Note that if a Markov chain satisfies the detailed balanced equations, then the global balance equations are satisfied as well.

We show now that eq. (5) satisfies eq. (6). We distinguish between two types of transition. The first is when pages, already stored in the cache, have their positions exchanged. The second is when

the page currently positioned at the bottom of the cache is replaced by a new page. In the first case, assume $\vec{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_j, \sigma_{j+1}, \ldots, \sigma_K\}$ and $\vec{\sigma}' = \{\sigma_1, \sigma_2, \ldots, \sigma_{j+1}, \sigma_j, \ldots, \sigma_K\}$. Then

$$\begin{aligned} q(\vec{\sigma}, \vec{\sigma}')\pi(\vec{\sigma}) &= \left(p_{\sigma_{j+1}} \cdot \frac{c_{\sigma_{j+1}}}{c_{max}}\right) \cdot \left(C_2 \cdot \prod_{i=1}^{K} (p_{\sigma_i} c_{\sigma_i})^{K-i+1}\right) \\ &= \left(p_{\sigma_j} \cdot \frac{c_{\sigma_j}}{c_{max}}\right) \cdot \left(C_2 \cdot \prod_{i=1}^{K} (p_{\sigma_i} c_{\sigma_i})^{K-i+1} \cdot \frac{p_{\sigma_{j+1}} c_{\sigma_{j+1}}}{p_{\sigma_j} c_{\sigma_j}}\right) \\ &= q(\vec{\sigma}', \vec{\sigma})\pi(\vec{\sigma}'). \end{aligned}$$

The second case can be handled in a similar way (see proof of Theorem 3). □

Looking at eqs. (1) and (2) we remark that if we replace the request probabilities p_i by the "weighted" request probabilities

$$\frac{p_i c_i}{\sum_{j=1}^{N} p_j c_j}, \tag{7}$$

then we exactly obtain eqs. (4) and (5) (note that the denominator in eq. (7) is just a normalization constant). This shows duality between the performances of the newly proposed algorithms and those of the classical caching algorithms.

The hit ratio is computed, in this case, with respect to the weighted request probabilities defined by eq. (7). The duality property guarantees that the ratio between the hit ratio of OPT-C and the hit ratio of LRU-C is bounded.

3.3 Caching Algorithms for Documents with Different Sizes

The randomization technique is very similar to that introduced in the previous section. We let $s_{min} = \min(c_1, c_2, \ldots, c_N)$ be the size of the smallest documents among the N documents, and $\tilde{d}_i = s_{min}/s_i$ be the normalized "density" of page i. Then, following a request for page i, the algorithm

LRU-S (respectively, CLIMB-S) acts as LRU (resp., CLIMB) with probability $\tilde{d}_i$; otherwise the cache state is left unmodified.

The analysis of these algorithms is somewhat more difficult, since the number of documents stored in the cache changes over time. For LRU-S, a simple analysis can still be carried out by slightly modifying the definition of the state-space of the Markov chain. A state is now described by a N-length vector $\vec{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ which corresponds to the order at which each document has last been put at the head of the cache. The stationary probabilities of this Markov chain are given by the following theorem:

Theorem 2

$$\pi^{LRU\text{-}S}(\vec{\sigma}) = \prod_{i=1}^{N} \left(\frac{p_{\sigma_i}/s_{\sigma_i}}{\sum_{j=1}^{N} p_j/s_j - \sum_{j=1}^{i-1} p_{\sigma_j}/s_{\sigma_j}} \right) . \tag{8}$$

The proof of this theorem is essentially the same as of eq. (4) in Theorem 1.

Given the stationary probabilities, the hit ratio of LRU-S can be computed as follows. First we define a set $\mathcal{A}_i$ which includes all the states with page i in the cache

$$\mathcal{A}_i = \{\vec{\sigma} | \sum_{j=1}^{i} s_{\sigma_j} \leq K\}. \tag{9}$$

The hit ratio is then given by

$$H.R. = \sum_{i=1}^{N} p_i \sum_{\mathcal{A}_i} \pi^{\text{LRU-S}}(\vec{\sigma}), \tag{10}$$

where we sum over all the indices i, the probability that page i is requested and found in the cache.

Note that a similar analysis can be performed for CLIMB-S. However, for the sake of analysis, an auxiliary memory maintaining the identity and size of documents not stored in the cache should be implemented.

Finally, we remark that it is straightforward to design an algorithm which deals with both varying-size and varying-cost documents. For this purpose, we define the following quantities

$$\beta_i = c_i/s_i; \qquad \beta_{max} = \max_i \beta_i; \qquad \tilde{\beta}_i = \beta_i/\beta_{max}.$$

Upon a request for page i, an algorithm generalizing the LRU (CLIMB) algorithm will perform the same operation as LRU (CLIMB) with probability $\tilde{\beta}_i$, and with a complementary probability will leave the cache state unmodified. The analysis of this algorithm is similar to the previous ones.

4 Speeding-up the Convergence Rate

As explained in Section 2.2, it is desirable that caching algorithms converge quickly. However, there may be instances where the convergence rate of the new caching algorithms is slow. Consider, for instance, the case of a single document having a high access cost, equal to 100, and all the $N-1$ other documents having a same low access cost, equal to 1. Then, upon a request for a low cost document, the probability that the state of the cache will be updated does not exceed 1/100. The ability to speed-up the convergence rates of the caching algorithms would be of particular significance, from both theoretical and practical perspectives.

The convergence rate of CLIMB-C (and CLIMB-S) can be significantly improved, without affecting the steady-state probabilities. This is mainly due to the reversibility property that the underlying Markov chain of this algorithm satisfies.

The quick converging version of CLIMB-C, which we refer to as CLIMB-CF, works as follows. Assume that the current cache state is $\vec{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_i, \sigma_{i+1}, \ldots, \sigma_K\}$. Then, if page σ_{i+1} is requested, its position is exchanged with the preceding page σ_i; with probability $\min(1, c_{\sigma_{i+1}}/c_{\sigma_i})$. With complementary probability, the cache state is left unchanged. Similarly, if the requested page, say page j, is not currently stored in the cache, then it replaces page σ_K, with probability

$\min(1, c_j/c_{\sigma_K})$; otherwise, nothing is done.

Note that the transition probabilities of CLIMB-CF are always greater or equal of those of CLIMB-C. In particular, documents with the same access cost are exchanged with probability 1.

The following theorem shows that the stationary probabilities of CLIMB-C and CLIMB-CF are identical.

Theorem 3 *Under the IR model,*

$$\pi^{CLIMB\text{-}CF}(\vec{\sigma}) = \pi^{CLIMB\text{-}C}(\vec{\sigma})$$

Proof: We show that the detailed balanced equations, given by eq. (6), are satisfied. We distinguish between the same two types of transition, as in the proof of Theorem 1, part (ii). This time, we show the proof for the case where the page at the bottom of the cache is being replaced by a new page. Let $\vec{\sigma} = \{\sigma_1, \sigma_2, , \ldots, \sigma_{K-1}, \sigma_K\}$ and $\vec{\sigma}' = \{\sigma_1, \sigma_2, \ldots, \sigma_{K-1}, j\}$. Without any loss of the generality, assume $c_{\sigma_K} > c_j$. Then,

$$q(\vec{\sigma}, \vec{\sigma}') = p_j \cdot \frac{c_j}{c_{\sigma_K}}; \qquad q(\vec{\sigma}', \vec{\sigma}) = p_{\sigma_K}$$

and

$$\begin{aligned} q(\vec{\sigma}, \vec{\sigma}')\pi(\vec{\sigma}) &= \left(p_j \cdot \frac{c_j}{c_{\sigma_K}}\right) \cdot \left(C_2 \cdot \prod_{i=1}^{K} (p_{\sigma_i} c_{\sigma_i})^{K-i+1}\right) \\ &= p_{\sigma_K} \cdot \left(C_2 \cdot \prod_{i=1}^{K} (p_{\sigma_i} c_{\sigma_i})^{K-i+1} \cdot \frac{p_j c_j}{p_{\sigma_K} c_{\sigma_K}}\right) \\ &= q(\vec{\sigma}', \vec{\sigma})\pi(\vec{\sigma}'), \end{aligned}$$

which proves the statement. □

One can envision a similar mechanism aiming at speeding-up LRU-S (or LRU-C). For instance, with LRU-SF, when page i is requested, it is placed at the head of the cache with probability

$\min(1, s_{\sigma_1}/s_i)$, where s_{σ_1} is the size of the page currently at the head of the cache. Unfortunately, the steady-state performance of LRU-SF is not identical to LRU-S. Nevertheless, LRU-SF still performs comparably to caching algorithms of higher complexity, as shown in the next section.

Another simple way to accelerate the convergence rate, at the expense of possibly reduced steady-state performance, is to set a higher value for the minimum size s_{min} (or, respectively, a lower value for the maximal cost c_{max}). For instance, if there are only a few documents of size smaller than 1K, then s_{min} may be set to 1K. If a document of size smaller than s_{min} is requested, then it is handled as a document of size s_{min}.

5 Numerical Results

In this section, the performance of the new algorithms are illustrated through a couple of examples. We focus on the LRU algorithm and its variants, which are expected to be the most useful in practice. The following results show that the new randomized algorithms achieve high performance, at low implementation complexity.

We first use simulation to compare the performance of the LRU-S, LRU-SF, GDS and LRU algorithms. We consider $N = 500$ documents and assume that the popularity of the documents follows a Zipf distribution, $p_i = \omega/i^{\alpha}$, where $\alpha = 0.8$ and ω is a normalization constant. The file sizes (in kilobytes) are generated according to the following heavy-tailed distribution

$$\Pr(s > x) = \frac{1}{1+x}.$$

Figure 2 depicts the hit ratio of the various algorithms as a function of the cache size. The cache size is given as a fraction of the amount of memory needed to achieve 100% hit ratio. The figure shows that LRU-S exhibits the best performance, while LRU-SF and GDS are statistically equivalent, and LRU performs the poorest.

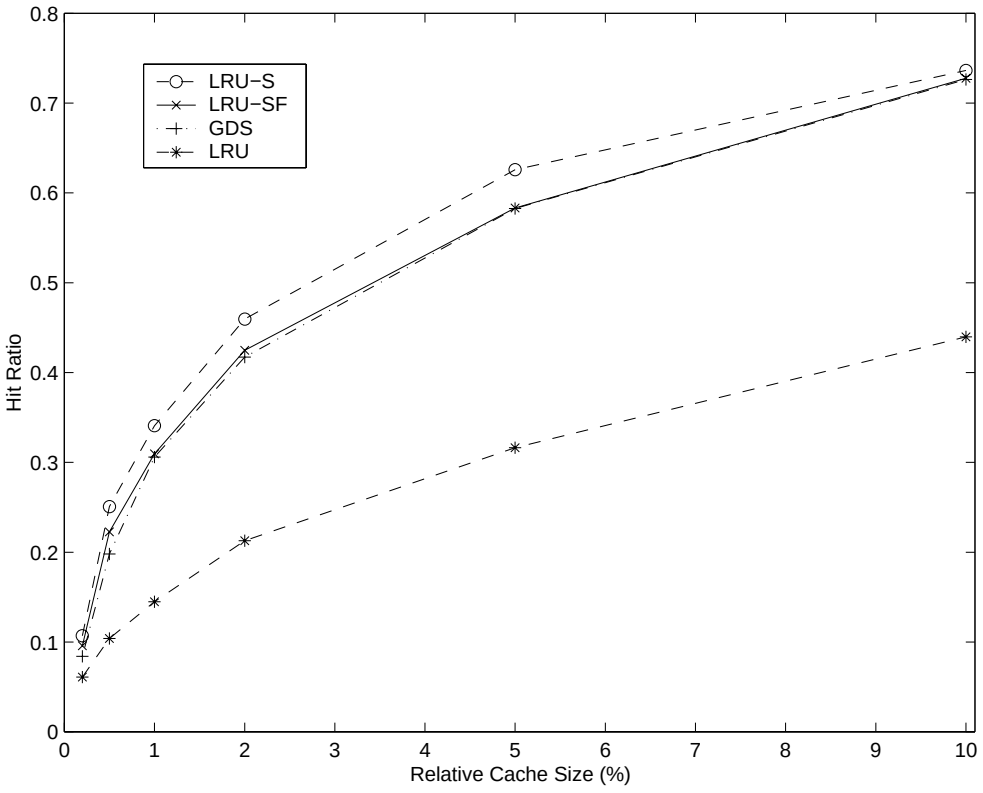

Figure 2: Simulation Results

In Fig. 3, we compare the performance of LRU-S, GDS and LRU, using real traces of requests to a Web proxy cache [DEC96]. In this graph, we present the results we obtained from the first two days of the traces, representing over 10G of data (results from other parts of the trace are similar). The data were post-processed similarly to the method described in [CaI97]. The hit ratio depicted in Fig. 3 is given as a fraction of the maximal achievable hit ratio. This latter quantity is smaller than 1 (typically 0.5) because many documents are requested only once over the trace. In order to speed-up LRU-S, we set s_{min} equal to 10K. From Fig. 3, we observe that LRU-S performs slightly better than GDS, and significantly better than LRU. We note that the results obtained with real traces are qualitatively similar to those obtained with simulation, although LRU seems to perform better with the real traces. This reinforces the case for the IR model, as a useful tool for predicting the relative performance of web caching algorithms.

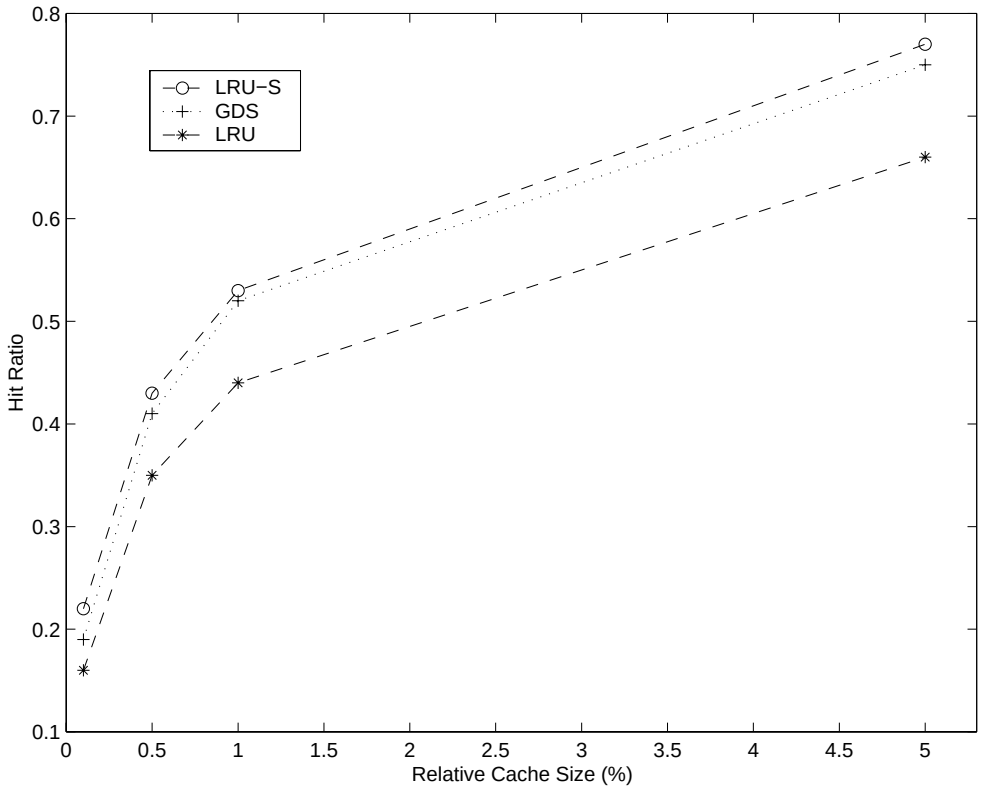

Figure 3: Trace Results

6 Concluding Remarks

In this paper, we have proposed new scalable algorithms for web caching. We have shown that randomization is the key feature in achieving full scalability. We have introduced new algorithms called LRU-C and CLIMB-C, and LRU-S and CLIMB-S, which, respectively, address the non-uniform access cost and and the non-uniform size properties of web documents. We showed that the performances of these algorithms under an augmented version of the IR model are fully dual to those of the classical LRU and CLIMB algorithms under the standard IR model. Note that this result is extendible to other Markovian algorithms [ACK87] such as FIFO. The analysis of the CLIMB algorithm provided interesting insight into ways of increasing the responsiveness to non-stationary trends. In particular, we showed that CLIMB-CF, a variant of the CLIMB-C algorithm,

may exhibit a convergence rate significantly faster than CLIMB-C while achieving the same steady-state performance. Through numerical simulations and real traces experiments, we showed that the new algorithms achieve performances comparable to existing algorithms of higher complexity.

Throughout the work, we have focused on optimizing the hit ratio metric. However, when bandwidth is scarce, other performance measures may be of interest such as the "byte hit ratio" and the "packet savings ratio" [CaI97] that is, the percentage of packet transmissions saved by the cache. Our algorithms can easily be tuned to optimize these performance metrics by appropriately setting the cost parameter c_i. For instance, according to [CaI97], a cache hit's packet saving is $(2 + \lceil file\_size/536 \rceil)$, as an estimate of the actual number of network packets needed to be transmitted if the request is a cache miss (1 packet for the request, 1 for the reply, and $\lceil file\_size/536 \rceil$ extra data packet(s), assuming a 536-byte TCP segment size). In such a case, the access cost of document i is $c_i = 2 + \lceil s_i/536 \rceil$.

An important feature, which we haven't considered here, is the finite life time of web documents. Web caches may potentially store stale documents which are useless to the users, such as pages reporting out-dated stock quotes. An important area for future research is in the development of caching algorithms that do not only take into consideration the popularity of documents, but also their degree of freshness.

Acknowledgement: The authors would like to thank Prof. A. Trachtenberg for useful comments on an earlier version of this paper.

References

[ACK87] O. Aven, E. Coffman Jr., and Y. Kogan, *Stochastic Analysis of Computer Storage,* Series: Mathematics and its Applications, D. Reidel Publishing Company, 1987.

[AWY99] C. Aggarwal, J. Wolf, P. Yu, "Caching on the World Wide Web," *IEEE Transactions on Knowledge and Data Engineering*, Vol.11, No.1, pp.94-107, January/February 1999.

[BaO00] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," *IEEE Communications Magazine*, Vol.38, No. 5, pp.178-185, May 2000.

[BoE98] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis,* Cambridge University Press, 1998.

[Bre99] L. Breslau et al., "Web Caching and Zipf-like Distributions: Evidence and Implications," in the proceedings of *Infocom '99*, March 1999, New York.

[CaI97] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms," in the proceedings of *USENIX '97.*

[DEC96] Digital Equipment Corporation, Digital's Web Proxy Traces, ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html.

[DiA99] J. Dilley and M. Arlitt, "Improving Proxy Cache Performance," *IEEE Internet Computing*, Vol. 3, No. 6, pp.44-50, November 1999.

[Dus97] B. Duska et al., "The Measured Access Characteristics of World-Wide-Web Client Proxy Caches," in the proceedings of *Usenix Symposium on Internet Technologies and Systems*, December 1997.

[FrW74] P. Franaszek and T. Wagner, "Some Distribution-Free Aspects of Paging Algorithm Performance", *Journal of the ACM*, Vol.21, No.1, pp.31-39, January 1974.

[Gel73] E. Gelenbe, "A Unified Approach to the Evaluation of a Class of Replacement Algorithms," *IEEE Transactions on Computers*, Vol.C-22, No.6, pp.611-618, June 1973.

[Hen76] W.J. Hendricks, "An Account of Self-Organizing Systems," *SIAM J. Comput.*, Vol. 5, No. 4, pp.715-723, December 1976.

[HeH85] J. Hester and D. Hirschberg, "Self-Organizing Linear Search," *Computing Surveys*, Vol. 17, No. 3, pp.295-310, September 1985.

[JiB00] S. Jin and A. Bestavros, "GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Streams," Technical Report 2000-011, CS Department, Boston University, April 2000.

[Kel79] F. P. Kelly, *Reversibility and Stochastic Networks*, Wiley, 1979.

[Kin71] W.F. King, III, "Analysis of Paging Algorithms," in proceedings of *IFIP Congress*, 1971, Ljubjana, Yugoslavia.

[MoS91] B. Moret and H. Shapiro, *Algorithms from P to NP: Volume 1 Design and Efficiency,* The Benjamin/Cummings Publishing Company, Redwood City: CA, 1991.

[PsP01] K. Psounis and B. Prabhakar, "A Randomized Web-Cache Replacement Scheme," in the proceedings of *Infocom 2001*, April 2001, Anchorage.