Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 45 (2004) 379–398
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

Modeling correlations in web traces and implications for designing replacement policies

Konstantinos Psounis [a,*], An Zhu [b], Balaji Prabhakar [c], Rajeev Motwani [b]

[a] Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA
[b] Department of Computer Science, Stanford University, Stanford, CA 94305, USA
[c] Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305, USA

Received 2 February 2003; received in revised form 15 August 2003; accepted 8 January 2004
Available online 20 February 2004

Responsible Editor: G. Pacifici

Abstract

A number of web cache-related algorithms, such as replacement and prefetching policies, rely on specific characteristics present in the sequence of requests for efficient performance. Further, there is an increasing need to synthetically generate long traces of web requests for studying the performance of algorithms and systems related to the web. These reasons motivate us to obtain a simple and accurate model of web request traces.

Our Markovian model precisely captures the degrees to which temporal correlations and document popularity influence web trace requests. We describe a mathematical procedure to extract the model parameters from real traces and generate synthetic traces using these parameters. This procedure is verified by standard statistical analysis. We also validate the model by comparing the hit ratios for real traces and their synthetic counterparts under various caching algorithms.

As an important by-product, the model provides guidelines for designing efficient replacement algorithms. We obtain optimal algorithms given the parameters of the model. We also introduce a spectrum of practicable, high-performance algorithms that adapt to the degree of temporal correlation present in the request sequence, and discuss related implementation concerns.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Request sequence model; Web traces; Web caching; Replacement schemes; Performance analysis

1. Introduction

The use of web caches for reducing network traffic and download latency has rendered them an important component of the Internet infrastructure. Algorithms devised for web caches need to take advantage of specific characteristics present in the request sequence for efficient performance. For example, cache replacement policies exploit the

* Corresponding author. Tel.: +1-213-7404453.
E-mail addresses: kpsounis@usc.edu (K. Psounis), anzhu@stanford.edu (A. Zhu), balaji@stanford.edu (B. Prabhakar), rajeev@cs.stanford.edu (R. Motwani).

1389-1286/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2004.01.004

temporal locality between successive requests during eviction times. [1] Prefetching policies take advantage of correlations present in inter-document requests (see [23], for example). Since algorithm design relies on the features inherent in HTTP request sequences, it is important to obtain a simple and accurate model of such sequences. This allows one to understand the impact of various characteristics like temporal correlations and document popularity on the performance of the cache, and guides the design of better algorithms. [2]

Broadly, there are three key features of web traffic that we wish to model. They are: (i) temporal correlations—this captures the likelihood of requesting a document in the near future, given that it has been currently requested, (ii) long-term document popularity—this captures the probability that a document is likely to be requested relative to other documents, and (iii) spatial (inter-document) correlations—this models the chance that document j will be requested next, given that document i has been requested now. In this work we focus on the relationship between temporal correlations and popularity as it relates to designing cache replacement schemes, and only briefly discuss spatial correlations. For the same reason, we also initially ignore other attributes of a document, such as its size.

Our approach simultaneously models both temporal correlations and document popularity. Temporal correlations are captured using a Markov model for the request sequence. We do not assume specific parameters for the Markov chain, instead we infer them from the real traces we wish to model. Similarly, we also infer the distribution of the long-term popularity from the traces. Further, our approach can easily model document popularities that change slowly over time, a phenomenon recently observed in traces [21].

Observations about the characteristics of web trace sequences have influenced the design of document replacement schemes. For example, the least recently used (LRU) algorithm is known to exploit temporal correlations present in the request sequence [31]. On the other hand, the least frequently used (LFU) algorithm exploits long-term document popularity to achieve high hit rates. Indeed, it is easy to see that when request sequences are independent and identically distributed, LFU achieves the highest hit rates [3] [12].

Real web request sequences, however, are influenced both by temporal correlations and document popularity. This observation and clever heuristics have led to the proposal of replacement algorithms which combine features of both LRU and LFU [2,8,10,17,18,20]. Our model adds to this body of work in three ways: (a) it allows us to quantify the degree to which real traces are influenced by temporal correlations and document popularity, (b) it allows us to explore extreme cases by considering traces with different mixtures of temporal correlation and document popularity, and (c) this, in turn, points to a way of designing algorithms that trade-off correlations and popularity.

Traditionally, the performance of algorithms is studied using competitive analysis. Another approach is to probabilistically analyze the behavior of algorithms under realistic input. In the case of replacement algorithms, due to the difficulty in taking into account temporal correlations in an analytically tractable manner, these studies are usually done under the assumption of an independence reference model (e.g. [6,32]). Our model allows to analytically study the performance of eviction schemes under request sequences that exhibit correlations.

Another contribution of our work is for designing good workload generation tools. An

[1] Temporal locality was first introduced in the context of program behavior. Despite extensive studies, it is usually loosely defined. For now, we adopt the following definition found in many studies, e.g. [24]: a sequence of requests exhibits temporal locality if an object just referenced has high probability of being referenced in the near future. In Section 2.2 we make this precise.

[2] Temporal correlations and variable document popularities are known to be the two causes of locality, see, for example [16]. A sequence of requests exhibits temporal correlation if the document requested at time n depends on the documents requested at previous times. In Section 2.2 we elaborate more on these notions and show how they are related.

[3] This result holds under the assumption that all documents are of the same size.

important challenge for these tools is to capture temporal correlations present in web request traces. This is commonly done using the LRU stack model (LRUSM) [30]. One issue with the LRUSM is that each document in the generated sequence of requests has an equal long-term probability of being requested [27], which is incompatible with the Zipfian long-term popularity observed in real traces. To solve this problem, researchers have to use various heuristics (see, for example [4,7,11]). Our model provides a rigorous and simple way to incorporate any long-term popularity distribution for the documents. Further, the LRUSM works with stack distances, whereas our model works directly with document requests, and hence it is a more natural tool for generating request sequences.

It has been observed that in addition to temporal correlation and long-term popularity, other attributes of a document, such as its size and the cost of fetching it from the origin-server, play an important role in the performance of web replacement schemes. Since the focus of this work is on the trade-off between temporal correlations and popularity, we initially ignore these other attributes. Later, we comment on how to incorporate these parameters in designing good eviction schemes. As a final comment, we limit our discussion to static documents and do not consider documents that change dynamically over time.

The paper is organized as follows. Section 2 introduces the model whose parameters are extracted from real traces. We then generate synthetic traces and verify that they exhibit the desired statistical properties. Further, we validate the model by comparing the hit rates for real traces and their synthetic counterparts under various caching algorithms. Based on the trace model, in Section 3 we formulate the problem of obtaining an optimal algorithm with respect to hit rate as a Markov decision process, and obtain an online replacement algorithm that is optimal conditioned on the state of the Markov process. This algorithm, however, relies on knowledge of trace characteristics and it is hard to use in practice. Thus, we introduce γ-LRU, a class of practicable replacement policies parameterized by γ with LRU and LFU obtained for extreme values of γ, and study its performance. Finally, we address implementation issues related to γ-LRU, like taking into account its data structure requirement and the variability in document sizes.

1.1. Related work

There have been many attempts to model a sequence of web requests. The simplest approach is to assume an independent reference model, i.e, to assume that the next request is independent of all previous requests (e.g. [6,32]). Web traces are known to exhibit strong temporal locality, that is, the probability of requesting a document in the near future is high if this document has been recently requested. This property is not incompatible with the independent reference model. For example, the authors in [6] showed that under the independent reference model, the Zipfian nature of the long-term popularity of documents creates temporal locality similar to that observed in real traces.

A more careful study of real traces revealed that long-term popularity alone does not suffice to fully capture the temporal locality exhibited by real traces, the request sequence needs to be correlated in time (see [16], or [18]). In other words, temporal locality emerges from two distinct phenomena: the Zipfian *popularity* of documents and the temporal *correlation* between requests.

Hence, there is a need to characterize and quantify the degree of correlation in a sequence of web requests, and to devise a correlated reference model that can be used to synthetically generate traces.

The need for a correlated reference model made researchers consider the well known LRU stack model (LRUSM) [30]. This model is obtained by maintaining the documents in a stack in increasing order of time since the last reference to the document. The position in which a document is found upon a request is termed the stack distance of that reference. The distribution of stack distances is an indicator of temporal locality because the stack distance measures the number of (unique) intervening references between references to the same document.

However, the LRUSM requires that each document in the generated sequence of requests has an equal long-term probability of being requested [27], which is incompatible with the Zipfian long-term popularity observed in web traces. In essence, the LRUSM fails to model varying document popularities. To deal with this problem, researchers have used various heuristics when using the LRUSM to synthetically generate sequences of web requests. Three such heuristics are described in [4,7,11]. The idea of the first heuristic is to move to the top of the stack a document that sits close to the position of the requested document with some appropriate probability, rather than always moving the requested document. The idea behind the other two heuristics is to factor out the effects of non-uniform popularity by normalizing the stack distance. The above ideas work well in practice when generating synthetic traces. However, there is no theoretical proof that the LRUSM together with the proposed heuristics yields request sequences with the desired statistical properties. Further, since these approaches are based on the LRUSM, they don't clearly distinguish how popularity and correlation affect locality.

A simpler task from obtaining a model to synthesize web request sequences is that of characterizing the degree of correlation in a given trace. Many studies have used the stack distance to capture and characterize temporal locality, and then correlation (see, for example [1,11,21]). These early studies succeed in characterizing locality but are less successful with correlation. The reason is that, as already mentioned, the LRUSM cannot distinguish the causes of locality, and hence correlation, directly. (This is also noted in a more recent work of the authors of one of the above papers [14].)

To this end, researchers attempted to characterize correlation using the distribution of the inter-reference distance [14,16]. [4] However, while inter-reference distance is a good measure of locality, it is not a good metric for correlation. Indeed, as also observed by the authors of the papers above, the distribution of inter-reference distances does not differ sizeably between an original trace and a randomly permuted version of it, because it is heavily affected by the popularity of documents. To sidestep this issue, the authors in [16] worked with the distribution of the inter-reference distance of equally popular documents, and the authors in [14] with the coefficient of variation obtained from the inter-reference distance of every unique reference of a trace. Both works devise a metric that expresses the degree of correlation in a trace with a single number. However, these metrics cannot answer questions of the type "how fast temporal correlations die?", and are based on a measure, the inter-reference distance, that is somewhat inappropriate.

In our work we take a new approach towards the problem of modeling correlations in web races. Our goal is to devise a mathematically rigorous model that can be used to synthetically generate traces of arbitrary correlation and popularity. At the same time, we want to use the model to characterize and quantify temporal correlations in a precise manner and at various levels of detail. Essential elements of our approach is a Markov model that captures correlations, and the estimation from real traces of the probability that the nth request in a trace is the same with the $(n-i)$th request. As it is shown in Sections 2.2 and 2.3, our model can generate traces of arbitrary correlation and popularity using a mathematically rigorous procedure, while its parameters characterize locality, correlation, and popularity at various levels of detail and in a precise manner.

2. The model

In this section we introduce our model for capturing long-term document popularity and short-term temporal correlation in web request sequences.

Index all documents in decreasing order of popularity. Let p_i be the probability of requesting the ith most popular document. We shall call the

[4] Note that the stack distance measures the number of *unique* intervening references between references to the same document, while the inter-reference distance measures the total number of intervening references between references to the same document.

probability distribution $\{p_i\}$ the distribution of the long-term popularity of the documents. In practice, $\{p_i\}$ is known to be Zipf-like [6]; i.e., $P[Y_n = i] = p_i \propto 1/i^{\theta}$, $0 < \theta < 1$.

Let X_n be the document requested at time $n \geqslant 1$ (time is slotted, one request per slot), and Y_n, $n \geqslant 1$, be a sequence of random variables that are independent and identically distributed according to $\{p_i\}$. We propose the following model for web requests for $n > h$:

$$X_n = \begin{cases} X_{n-1} & \text{with probability } \alpha_1, \\ X_{n-2} & \text{w.p. } \alpha_2, \\ \vdots & \\ X_{n-h} & \text{w.p. } \alpha_h, \\ Y_n & \text{w.p. } \beta, \end{cases}$$

where

$$\beta + \sum_{i=1}^{h} \alpha_i = 1. \tag{1}$$

In the model we keep a history of the h most recent requests to capture short-term (temporal) correlations. Y_n injects documents that may or may not have been recently requested and captures long-term (document) popularity. The probability that document i is going to be requested at time n, given that it was not requested in the last h requests is simply βp_i. The probability that document i is going to be requested at time n, given that it was requested only once in the last h requests at time $n - j$ ($1 \leqslant j \leqslant h$), equals $\alpha_j + \beta p_i$. In general,

$$P(X_n = i \,|\, X_{n-1}, \ldots, X_{n-h}) = \beta p_i + \sum_{j=1}^{h} \alpha_j 1_{(X_{n-j}=i)}. \tag{2}$$

For mathematical convenience, another way to interpret the model is the following. At each time slot, toss an $h+1$-sided, biased coin to decide the value of X_n. Let T_n be a random variable indicating the outcome of the toss at time $n > h$. Then, $P(T_n = j) = \alpha_j$ for $1 \leqslant j \leqslant h$ and $P(T_n = h+1) = \beta$. Using this notation, the model is described by the following equation:

$$X_n = \sum_{j=1}^{h} X_{n-j} 1_{(T_n=j)} + Y_n 1_{(T_n=h+1)}. \tag{3}$$

The parameters of the model are the history window h, the α_i's, the β, and the p_i's. Let $\overline{X}_n = (X_n, X_{n-1}, \ldots, X_{n-h+1})$ and $\bar{i} = (i_1, i_2, \ldots, i_h)$ be a vector of h requested documents. Since $P(\overline{X}_{n+1} = \bar{i} | \overline{X}_n, \overline{X}_{n-1}, \ldots) = P(\overline{X}_{n+1} = \bar{i} | \overline{X}_n)$, $\overline{X}_n$ is a Markov chain or equivalent X_n is an hth-order Markov chain. The $\overline{X}_n$, also X_n, has a unique stationary distribution since its state space is finite and it is irreducible and aperiodic (for $\beta > 0$).

In stationarity, X_n is distributed like Y_n. To see this, use Eq. (3) to get

$$P(X_n = i) = \sum_{j=1}^{h} \alpha_j \cdot P(X_{n-j} = i) + \beta \cdot P(Y_n = i). \tag{4}$$

Now, in stationarity, X_n equals in distribution some random variable X. Therefore, $P(X_n = i) = P(X_{n-j} = i) = P(X = i)$ for all $j = 1, \ldots, h$, and we get

$$P(X = i) \cdot \left(1 - \sum_{j=1}^{h} \alpha_j\right) = \beta \cdot P(Y_1 = i)$$

for all i. Since $1 - \sum_{j=1}^{h} \alpha_j = \beta$, X has the same distribution as Y_1 (recall that Y_n is i.i.d.).

2.1. Inferring model parameters

In this section we present techniques for inferring the parameters of the model, and use these techniques to obtain the parameters from real traces.

We infer $\{p_i\}$ from its empirical distribution; i.e. set p_i to equal the sample frequency of document i in the trace.

Inferring the α_i's and β is more involved. Let R be the number of requests in the trace. Suppose, for now, that a suitable value of the history parameter, h, has been chosen. (We present a procedure for determining h at the end of this section.) Let C_i, $1 \leqslant i \leqslant h$ be a counter that counts how many times the nth request, $h < n \leqslant R$, is the same with the $(n-i)$th request. Denote by c_i, $1 \leqslant i \leqslant h$, the ratio $C_i/(R-h)$. In other words

$$c_i = \frac{\sum_{n=h+1}^{R} 1_{(X_n = X_{n-i})}}{R-h} \quad \text{for } 1 \leqslant i \leqslant h.$$

If the piece of real trace was indeed generated by the proposed Markov chain model, then for $1 \leqslant i \leqslant h$,

$$\begin{aligned} c_i &\approx P(X_n = X_{n-i}) \\ &= \sum_{j=1}^{h} P(X_{n-i} = X_{n-j}; T_n = j) \\ &\quad + P(X_{n-i} = Y_n; T_n = h+1) \\ &= \sum_{j=1}^{h} P(X_{n-i} = X_{n-j}) P(T_n = j) \\ &\quad + P(X_{n-i} = Y_n) P(T_n = h+1) \\ &= \alpha_i + \sum_{j=1; j\neq i}^{h} c_{|i-j|}\alpha_j + \beta P(X_{n-i} = Y_n). \end{aligned}$$

Using the independence of Y_n and X_{n-i}, and the fact that X_{n-i} is distributed like Y_n in stationarity, $P(Y_n = X_{n-i}) = \sum_j P(Y_n = j \,|\, X_{n-i} = j) P(X_{n-i} = j) = \sum_j p_j^2$ for all $1 \leqslant i \leqslant h$. Hence,

$$c_i = \alpha_i + \sum_{j=1, j\neq i}^{h} c_{|i-j|}\alpha_j + \beta \sum_{j: p_j \neq 0} p_j^2. \tag{5}$$

Eq. (5), for $1 \leqslant i \leqslant h$, together with Eq. (1) form a linear system of $h+1$ equations with $h+1$ unknowns that can be solved to calculate α_i's and β. Use Eq. (1) to substitute β with $1 - \sum_{i=1}^{h} \alpha_i$ in Eq. (5). Then, the α_i's are computed by solving the linear system $A \cdot \alpha = B$ where $\alpha = (\alpha_i)$ is a $1 \times h$ vector, $B = (c_i - \sum_j p_j^2)$ is a $1 \times h$ vector, and A is an $h \times h$ matrix given by the following expression:

$$A = \begin{pmatrix} 1 - \sum p_j^2 & c_1 - \sum p_j^2 & c_2 - \sum p_j^2 & \cdots & c_{h-1} - \sum p_j^2 \\ c_1 - \sum p_j^2 & 1 - \sum p_j^2 & c_1 - \sum p_j^2 & \cdots & c_{h-2} - \sum p_j^2 \\ c_2 - \sum p_j^2 & c_1 - \sum p_j^2 & 1 - \sum p_j^2 & \cdots & c_{h-3} - \sum p_j^2 \\ \vdots & & & \ddots & \vdots \\ c_{h-1} - \sum p_j^2 & c_{h-2} - \sum p_j^2 & c_{h-3} - \sum p_j^2 & \cdots & 1 - \sum p_j^2 \end{pmatrix}.$$

Once the α_i's are determined, β is computed from Eq. (1).

What remains is a procedure for determining the proper history h. In theory, overestimating h, which results in a larger matrix, would still find the correct α_i's and history ($\alpha_i = 0$ for all $i > h$). However, in practice, larger history leads to more rounding and statistical errors, hence to small negative α_i's in the solution of the linear system. A solution to this is to start from an overestimated history, and lower its value until all the α_i's are positive.

This procedure terminates faster, the closer the overestimated history is to the actual history. To determine an overestimated history we execute the following steps: (i) we start with a large value of h, traverse the trace, and compute the c_i's, (ii) randomly permute the trace to cancel any short-term correlations, traverse the permuted trace, and compute the corresponding normalized counters, denoted by c_i''s, and (iii) output as an overestimated history the number of c_i''s that are larger than the average value of the c_i''s. [5]

2.2. Locality, correlation, and popularity

A word on the connection between temporal locality, temporal correlation, and document popularity is in order. We start with stating the usual definition for temporal locality: a sequence of requests is said to exhibit temporal locality if recently requested documents have high probability of being requested in the near future [24]. This definition is intuitive but can be interpreted in many ways. To avoid confusion, we will use as a metric of temporal locality the distribution of inter-reference distance. In particular, if X_n is the document requested at time n, define $d_i(k)$ as follows:

$$\begin{aligned} d_i(k) = P(&X_{n+k} = i, X_{n+j} \neq i \\ &\text{for } j = 1, \ldots, k-1 \,|\, X_n = i), \end{aligned}$$

that is, $d_i(k)$ is the probability that document i is going to be requested again after k requests, given that it has just been requested. Also, denote by $d(k)$ the average over all documents, i.e.,

$$d(k) = \sum_i p_i d_i(k).$$

Notice that previous studies have used $d(k)$ to measure the degree of locality in traces, e.g. [6,21], but only under the (unrealistic) assumption of an independent reference model.

[5] In theory, $c_j' = \sum p_j^2$ for all i. Since there is some positive correlation between consecutive requests of the original trace, $c_i \geqslant \sum p_j^2$ for some i. Hence, since the c_i's are decreasing, there is an index i_h, such that $c_{i_h} < \sum p_j^2$ while $c_i \geqslant \sum p_j^2$ for $i < i_h$. This index is the overestimated history that we use.

A sequence of requests with no locality is created by the independent reference model when all documents have the same probability of being requested. In this case, if N is the total number of documents, it is easy to see that $d_i(k) = d(k) = (1/N)(1-1/N)^{k-1}$. For large N, this is roughly equal to $1/N$ and does not depend on k.

The independent reference model together with a skewed distribution yields some degree of locality. This was shown in [6] for the case of a Zipfian distribution. In particular. it was shown that $d(k) \approx 1/(k \ln N)$. Since $d(k)$ decreases linearly with the distance k, documents have on average larger probability of being requested sooner than later. (Notice that under the independent reference model, $d_i(k)$ is simply equal to $P(X_{n+k} = i, X_{n+j} \neq i$ for $j = 1, \ldots, k-1)$.)

In most real traces including web request streams, temporal locality emerges from two distinct phenomena: the popularity of documents and the (temporal) correlation between requests. Before we proceed to show how correlations induce locality, let us rigorously define the concept of temporal correlation: a sequence of requests is said to exhibit temporal correlation if the probability of requesting a particular document at time n depends on the documents requested at previous times. Usually, traces exhibit short-term, positive temporal correlation, that is, the probability of requesting a document in the future given that the document is recently requested, is higher than it would be if the document was not recently requested. This is equivalent to stating that $P(X_n = k \,|\, X_{n-i} = k) > P(X_n = k \,|\, X_{n-i} \neq k)$ for "small" i. [6] Notice that if requests are independent, the above probabilities are equal and there is no temporal correlation in the trace.

Our correlated reference model clearly distinguishes the two causes of locality. To see this, consider Eq. (4). Document i is requested next, either because it was recently requested, i.e. due to correlation, or because it is popular. The effect of correlation in locality is expressed by the α_i's, while the effect of popularity is expressed by β. The sum of the α_i's, or equivalently, $1-\beta$, is a simple quantity that characterizes the degree of correlation in the trace. The values of the α_i's are a more detailed measure of correlation: they describe how correlation dies with distance (see Fig. 1).

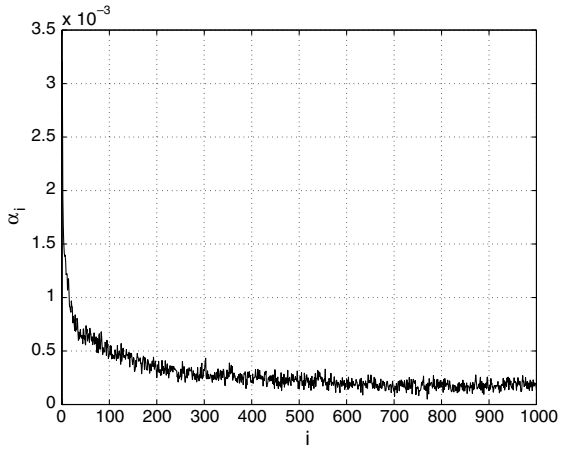

Fig. 1. α_i's for the original trace ($h = 1000$, $\beta = 0.708$).

To precisely show how correlation affects locality under our correlated reference model it is useful to compute $d(k)$. Without loss of generality, assume that correlations exist only for a distance of one and hence $h = 1$. Then, $d_i(1) = P(X_{n+1} = i \,|\, X_n = i) = \beta p_i + \alpha_1$, and $d(1) = \sum_i p_i(\beta p_i + \alpha_1) = \beta \sum_i p_i^2 + \alpha_1$. From this expression it is clear that locality is caused by popularity, the first term, and by correlation, the second term. Further, notice that $\sum_i p_i^2$ minimized for a uniform distribution, and maximized for the deterministic case, i.e. when $p_i = 1$ for a specific i and zero otherwise. Hence, the more skewed the popularity distribution, the larger its effect to locality.

It is interesting to compare $d(1)$ with $d(1)'$, the corresponding probability under the independent reference model. First, note that $d(k)' = \sum_i p_i(1-p_i)^{k-1}$. Now, since $d(1)' = \sum_i p_i^2$ and $\beta p_i + \alpha_1 > p_i$, it follows that $d(1) > d(1)'$ which means that, as expected, correlation increases locality.

One can compute $d_i(k)$ and $d(k)$ for larger values of k in a similar fashion. For example, $d_i(2) = P(X_{n+2} = i, X_{n+1} \neq i \,|\, X_n = i) = \beta p_i(1 - \beta p_i - \alpha_1) = \beta^2 p_i(1-p_i)$. Hence, $d(2) = \sum_i p_i \beta^2 p_i(1-p_i) = \beta^2 \times$

[6] In the context of our model, a "small" i is one that is less than or equal to the history h.

$d(2)'$, where $d(2)'$ is the corresponding probability under the independent reference model.

2.3. Generating synthetic traces

The model can be used to generate synthetic traces of arbitrary popularity and correlation. We show this by inferring the parameters h, α_i, β and p_i from a real trace, and using the model to generate a synthetic trace exhibiting the same statistical characteristics. We experiment with daily proxy cache traces from the National Laboratory for Applied Network Research (NLANR) [22] recorded from September 1999 to July 2003. The length of the traces varies from 300000 to 700000 requests. The traces are from three different sites, the PA, the SD and the RTF site. [7]

Using the procedure described above, the proper history for most of the traces is computed to be around 5000. We also experiment with shorter histories, namely 500, 1000, 2000 and 3000. In the rest of this section we present results from one trace only, since there are no differences in the results obtained from the various traces.

Fig. 1 presents the α_i's and β inferred from the trace for $h = 1000$. Notice that the α_i's are rapidly decreasing. Therefore, using a history that is smaller than the one obtained through the procedure above, suffices to capture the most essential short-term correlations. The rapid decrease of the values of the α_i's also implies that long-term correlations are very weak.

To determine whether the synthetic traces exhibit the same long-term popularities and short-term temporal correlations with their real counterparts, we perform the following procedure: (i) we verify that they possess the same history with the real traces, (ii) infer the rest of the model parameters from the synthetic traces, denoted by $\hat{\alpha}_i$. $\hat{\beta}$, and $\hat{p}_i$, and (iii) compare these parameters to the parameters inferred from the real traces.

In Table 1, we compare the values of β as obtained from the real and the corresponding synthetic trace. The values of β match very well for all histories shown. Fig. 2 plots the $\hat{\alpha}_i$'s for the synthetic trace that is generated using the parameters of the real trace.

Table 1
β values for various histories

h	β	$\hat{\beta}$
500	0.789	0.787
1000	0.708	0.707
2000	0.616	0.617

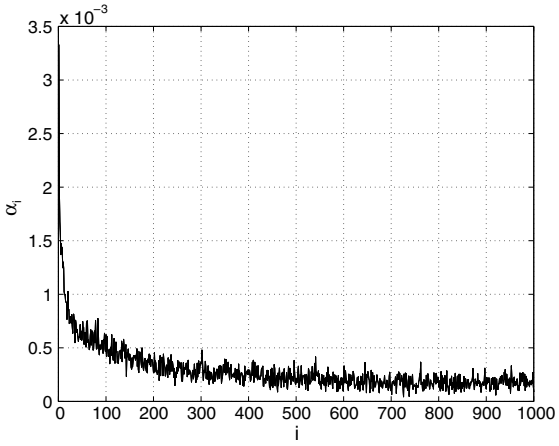

Fig. 2. $\hat{\alpha}_i$'s for the synthetic counterpart of the original trace ($h = 1000$, $\hat{\beta} = 0.707$).

By comparing Figs. 1 and 2, it is evident that the plots match very well. [8] Note that generating longer synthetic traces will only improve the matching between the model parameters. We also generated synthetic traces of length half of that of the original trace, and the matching was quite good again.

In Fig. 3 we plot the $\alpha_i's$ of the real trace in logarithmic scale. The values of α_i roughly follow a power law with $\theta = 0.46$. We will use this observation to generate α_i''s for simulation purposes.

As expected, for popular documents $p_i \approx \hat{p}_i$ as it is shown in Fig. 4. However, this is not the case for

[7] NLANR traces are a collection of traces from many sites. We use these sites because they have relatively long traces.

[8] If α_i's and $\hat{\alpha}_i$'s are plotted in the same plot, the plots are nearly indistinguishable with small differences only in the oscillations.

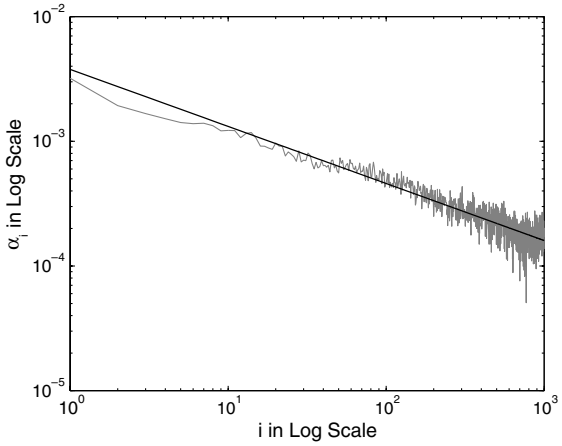

Fig. 3. α_i's in logarithmic scale.

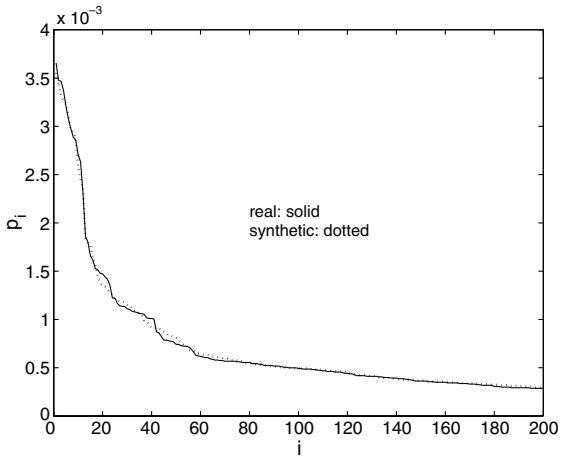

Fig. 4. Long-term popularity comparison between the original trace and its synthetic counterpart for popular documents.

very unpopular documents since estimating the p_i's of such documents from the sample frequencies is not accurate. The insufficient statistics for unpopular documents, and in particular one-timers, lead us to treat them in a special manner. We will first elaborate on the problems associated with one-timers, and then, we will describe one solution to circumvent these problems.

The first issue with unpopular documents is a well-known problem of statistics, often referred to as the problem of estimating the number of classes [15] or distinct values [9] in a population. In the context of web traces, the problem is that when a trace is not long enough, there are many unpopular documents that are not present in the trace. If one uses the sample frequencies to generate a trace with the same length as the original, there will be fewer one-timers in the generated trace than in the original. This is because the pool of available one-timers, from where the i.i.d. sequence Y_n takes values, is a lot smaller than in reality. To fix this problem we need to estimate the actual number of unpopular documents available to the users from which the trace is recorded.

The second issue with unpopular documents is the following. Recall that in stationarity X_n is distributed like Y_n. However, since the available real traces are short, the statistics for unpopular documents are far from their stationary values, and the correlations inherent in the model force unpopular documents to be more popular. The problem is well exemplified with one-timers: Suppose a one-timer, say document j, is requested during the generation of a synthetic trace. Then, based on the model, the probability of requesting document j again in the next h time slots equals $1-\Pi_{i=1}^{h}(\beta(1-p_j)+(1-\beta)(1-\alpha_i)) \approx 1-\Pi_{i=1}^{h}\times(1-\alpha_i) \approx \sum_{i=1}^{i}\alpha_i = 1-\beta$. Hence, since this probability is positive, documents that appear only once in the real trace may appear more than once in the synthetic trace.

We solve these problems as follows: whenever a one-timer is requested during the generation of a synthetic trace, we record a special identifier instead of the identifier corresponding to the document. After the trace is generated, we parse it and replace all the occurrences of the special identifier with one-timers. [9]

2.3.1. Model validation

A word on model validation is in order. The proposed model captures two important properties of web request streams: (i) the Zipfian nature of long-term popularity, and, (ii) the short-term

[9] It is easy to see that the number of occurrences of the special identifier will be very close to the total number of one-timers.

temporal correlations. There are other properties that are not captured, e.g. inter-document correlations.

To validate the model, we should investigate if it captures all the important properties of request streams for a given application. In this work we use web caching as our application. Hence, we should compare the hit rates achieved by real traces and their corresponding synthetic traces, under various cache replacement schemes. Table 2 compares the hit rates for six traces under LRU and LFU. (We use online LFU; i.e. the popularity/frequency of a document at time t is inferred from the requests that occurred at times $t' \leqslant t$.) In every experiment the cache size is set to 5% of the total number of distinct requests in each trace. For the first five traces this is around 13,000, and for the last, longer trace, it equals 19,400. As it is evident from the table, the hit rates are close for all traces. (Please refer to Table 5 of Section 3.3 for statistical characteristics of the used traces and exact cache sizes. Note that we do not take different document sizes into account here. Section 3.4 addresses this issue.) In Section 3.3, we compare the hit rates under more sophisticated eviction schemes, after a thorough discussion on cache replacement policies that takes place in Section 3.

2.4. Extensions of the basic model

In this section we show how one can extend the proposed model to capture more properties of a request sequence. First, we briefly comment on how to take into account inter-document correlations. Further, we show how to extend the model to capture different degrees of temporal correlations for each document, and work with document popularities that change slowly over time.

A simple way to model inter-document correlations is to associate with each document i a set of neighboring documents $N_i = \{j_1, j_2, \ldots\}$ that are strongly correlated with i. Let $|N_i|$ denote the cardinality of that set. Further, assume that all pairs of neighbors exhibit the same degree of inter-document correlation, and that these correlations die after one time slot. Then, it is straightforward to extend the basic model as follows:

$$X_n = \begin{cases} X_{n-1} & \text{with probability } \alpha_1, \\ X_{n-2} & \text{w.p. } \alpha_2, \\ \vdots & \\ X_{n-h} & \text{w.p. } \alpha_h, \\ Y_n & \text{w.p. } \beta_1, \\ Z_n & \text{w.p. } \beta_2, \end{cases}$$

where $\alpha_1 + \cdots + \alpha_h + \beta_1 + \beta_2 = 1$, $Z_n = j_k$ w.p. $1/|N_{X_{n-1}}|$ for all $k = 1, \ldots, |N_{X_{n-1}}|$, and $N_{X_{n-1}} = \{j_1, j_2, \ldots, j_{|N_{X_{n-1}}|}\}$ is the set of neighbors of the document requested at time $n-1$. Notice that even this simplistic way to account for inter-document correlations requires us to identify a set of neighbors for each document, which can be quite burdensome in practice.

Recall from Eq. (2) that the α_i's are the same for all documents. We generalize this by assuming that the probability of requesting document i, given the last h requests, equals

$$P(X_n = i \,|\, X_{n-1}, \ldots, X_{n-h}) = \beta_i p_i + \sum_{j=1}^{h} \alpha_{j,i} 1_{(X_{n-j}=i)},$$

where $\alpha_{1,i} + \cdots + \alpha_{h,i} + \beta_i = 1$ for all i. If document i_1 is more popular than document i_2, then

Table 2
Hit rates comparison for real and synthetic traces under LRU and LFU

Trace		LRU		LFU	
Date	Site	HR (real) (%)	HR (synthetic) (%)	HR (real) (%)	HR (synthetic) (%)
May 21st, 2001	PA	32.2	32.1	29.4	29.1
May 23rd, 2001	PA	31.7	31.4	28.5	28.1
January 19th, 2002	PA	30.7	31.1	28.9	28.7
January 20th, 2002	PA	32.7	32.4	31.1	30.8
July 10th, 2003	SD	29.3	29.2	25.2	25.0
July 10th, 2003	RTP	34.2	34.2	32.5	32.4

α_{j,i_1} may be larger than α_{j,i_2} for some j's. In particular, the $\alpha_{j,i}$'s corresponding to one-timers are equal to zero for all j. Thus, the issue with the basic model where one-timers appear more frequently in generated traces than in original ones due to correlations, is resolved. Notice that this extension requires to keep track of h parameters for each document, which is also burdensome in practice.

In the interest of simplicity and tractability, we do not study further these extensions to the basic model. The closeness of the hit rates for real and synthetic traces under various replacement schemes (Tables 2, 6, and 7) justify this decision.

Finally, note that our model can work with document popularities that change slowly over time, a phenomenon recently observed in traces [21]. All that is needed is to allow the distribution of Y_n to change slowly over time during the generation of a synthetic trace.

3. Caching algorithms

As mentioned previously, researchers have proposed various caching algorithms that exploit both short-term temporal correlations and long-term popularity. Despite the good performance of these algorithms, the precise trade-offs between temporal correlations and popularity are not well-understood. Using the request model introduced in Section 2 we can shed light on these trade-offs, formulate the problem of obtaining an optimal algorithm with respect to hit rate as a Markov decision process, and obtain an online replacement algorithm that is optimal conditioned on the state of the Markov process. We also introduce γ-LRU, a class of practicable replacement policies parameterized by γ, $0 < \gamma \leqslant 1$, with LRU and LFU obtained for extreme values of γ. When γ is properly chosen, the performance of γ-LRU is close to the performance of the optimal policy.

3.1. Optimal replacement policies

We start by introducing some notation and terminology. Let K be the size of the cache. Fix a replacement policy r. Let $\overline{Z}_n^r$ be a random vector with the K elements $Z_{n,i}^r$, $i = 1, \ldots, K$, each of which holds the document at the ith position of the cache at time n.

Let H_n^r be a random variable equal to 1 if there is a hit at time n, and zero otherwise. The long-term hit rate of a replacement policy r equals

$$HR^r \triangleq \lim_{N\to\infty} \frac{\sum_{n=1}^{N} H_n^r}{N}.$$

The H_n^r is a function of $\overline{Z}_n^r$ and X_n. Since $\overline{Z}_n^r$ depends only on the process X_n, and X_n is stationary and ergodic, the ergodic theorem [13] states that

$$\frac{\sum_{n=1}^{N} H_n^r}{N} \xrightarrow{\text{a.s.}} E(H_1^r) = P(H_1^r = 1). \tag{6}$$

Recall that $\overline{X}_n = (X_n, \ldots, X_{n-h+1})$. It is easy to see that $(\overline{Z}_n^r, \overline{X}_n)$ is a Markov chain with a finite state space S. Denote by π_s^r the stationary probability of being at state $s \in S$. [10] Then

$$HR^r \overset{(6)}{=} P(H_n^r = 1) = \sum_{s\in S} \pi_s^r P(H_n^r = 1 \,|\, s). \tag{7}$$

A replacement policy r^* is optimal with respect to hit rate if it maximizes the stationary probability $P(H_n^r = 1)$ over all policies r. Finding r^* is the object of the theory of Markov Decision Processes [26,29]. Using the vocabulary of this field, the Markov chain $(\overline{Z}_n^r, \overline{X}_n)$ is observed to be in a particular state s_n at time n. After observation of the state, an *action* must be chosen: which document to evict from the cache in case of a miss. [11] Based on the state s_n and the action chosen, a *reward* $R(s_n, r) = P(H_{n+1}^r = 1 \,|\, s_n)$ is earned and the probability distribution for the next state is determined.

The action for every state is dictated by a stationary policy r. The problem of interest is to determine the policy that maximizes the average

[10] The Markov chain $(\overline{Z}_n^r, \overline{X}_n)$ is aperiodic for $\beta > 0$ hence it has a stationary distribution.

[11] In case of a miss, $X_n \neq Z_{n,i}^r$ for all i, $1 \leqslant i \leqslant K$, and there are $K + 1$ documents out of which one should be evicted. In case of a hit. the action is null.

reward over an infinite time horizon, $\lim_{N\to\infty} E((1/N)\sum_{i=1}^{N} R(s_i, r))$. It is easy to see by inter-changing the limit and the expectation, [12] and using ergodicity, that the average reward is identical to the long-term hit rate. Hence,

$$r^* = \arg\max_r [HR^r] = \arg\max_r \left[\lim_{N\to\infty} E\left(\frac{1}{N} \sum_{i=1}^{N} R(s_i, r) \right) \right].$$

The theory of Markov Decision Processes provides a mathematical framework to identify optimal policies. However, usually only numerical solutions are possible. Furthermore, in many problems including the one we study, the computational requirements to obtain a numerical solution are overwhelming, because the number of states and/or policies is very large. In such situations, sub-optimal approximate solutions are derived [5]. Investigating optimality along these lines is out of the scope of this work. Instead, we set for a weaker optimality criterion, namely we wish to identify the policy r that maximizes the reward $R(s, r)$ for all states $s \in S$. This policy is, by definition, optimal with respect to the hit rate conditioned on the current state. In particular, it maximizes $P(H_n^r = 1 \mid s)$ in Eq. (7) for all s but does not necessarily maximize $P(H_n^r = 1)$.

An optimal algorithm. Suppose the cache is full. A new request arrives, and the cache must evict one out of the $K + 1$ documents. Order the documents in the cache according to their probability of being requested at the next request time, given the last h requests. Call by *LocalOpt*, the algorithm which evicts the document with the smallest probability of being requested at the next iteration. LocalOpt maximizes $P(H_n^r = 1 \mid s)$ for all s by definition.

The probability of requesting document i at the next request time, given the last h requests, is given by Eq. (2) for all i. We assume that LocalOpt either knows the p_i's, α_i's and β, or uses the ideas of Section 2.1 to estimate them. Thus, the algorithm can identify the document to evict at each eviction time. In the rest of the section we use LocalOpt as a benchmark for other algorithms.

[12] This is justified by the bounded convergence theorem [13].

How much better is LocalOpt than LRU and LFU? In particular, we are interested in comparing the hit rates of LocalOpt, LRU, and LFU under various request sequences with different temporal correlation characteristics. Thus, we use our model to generate such sequences.

We generate request sequences consisting of 5 million requests, drawn from a pool of 10000 documents whose popularity distribution is Zipf-like with parameter 0.5. We set h to 100, β to 0.5, 0.75 and 0.95, and assign to α_i's the rest of the probability according to a Zipf-like distribution with parameter 0.5. The cache size is set to 1000.

A relatively small value of β indicates a request sequence where short-term correlations are more significant as compared to a request sequence with a larger β. Thus, smaller values of β correspond to larger hit rates.

Table 3 shows the hit rate of LocalOpt, LRU, and LFU. When β is relatively small, LRU's performance is quite good (the hit rate of LRU is at least $1 - \beta$) while LFU's performance is bad. This is expected since the primary characteristic of the request sequence is the strong short-term correlations. On the other hand, when β is relatively large, LFU's performance is close to that of LocalOpt, while LRU's performance is bad. Therefore, both LRU and LFU fail to adapt to the degree of short-term correlation versus long-term popularity.

3.2. The γ-LRU class of replacement policies

Given a trace, which algorithm performs better depends on the degree of temporal correlation versus popularity. LocalOpt adapts to the degree of correlation, but requires the knowledge of α_i's and β which is hard to compute in practice. This motivates us to introduce γ-LRU, a class of

Table 3
Hit rate comparison for LocalOpt, LRU, and LFU under various values of β

	$\beta = 0.5$ (%)	$\beta = 0.75$ (%)	$\beta = 0.95$ (%)
LocalOpt	65.34	47.98	34.09
LRU	59.01	38.55	22.20
LFU	34.00	32.23	31.23

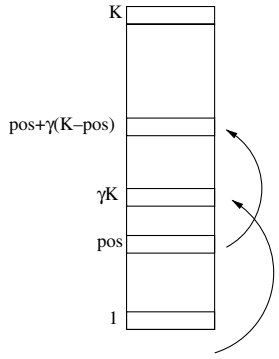

Fig. 5. How documents move in a cache that uses γ-LRU.

practicable replacement policies parameterized by γ with LRU and LFU obtained for extreme values of γ, that also adapts to the degree of correlation in a trace.

γ-LRU algorithm. Consider the following variation of LRU: whenever a page is requested, instead of moving the page to the top of the linked-list move it half way up. More generally, whenever a page is requested move it up by a fraction γ, $0 < \gamma \leqslant 1$. This is how γ-LRU performs. In particular, name the positions in the cache $1, 2, \ldots, K$ moving from bottom to top, as illustrated in Fig. 5. If the requested page is not in the cache, then insert it at position γK. Else, if it is currently at position *pos*, move it up to position $pos + \lceil \gamma(K - pos) \rceil$. [13]

How well does γ-LRU perform? Table 4 shows the hit rate of γ-LRU for the same parameters as in Table 3. The entries of Table 3 are repeated for comparison. It follows from this table that γ-LRU performs closely to LocalOpt for all values of β. In Table 4 we show the performance of γ-LRU for $\gamma = h/K$ and for the optimal γ, i.e., the γ value that maximizes hit rate, denoted by γ^*. We note that the performance of γ-LRU for $\gamma = h/K$ is close to the performance of γ^*-LRU. Later in this section we discuss this issue in more detail.

Table 4
Hit rate for γ-LRU and other schemes using the same parameters as in Table 3

	$\beta = 0.5$ (%)	$\beta = 0.75$ (%)	$\beta = 0.95$ (%)
LocalOpt	65.34	47.98	34.09
γ^*-LRU	62.81	45.61	32.25
γ-LRU ($\gamma = h/k$)	61.77	44.87	31.86
GD-F	61.77	44.00	29.05
LRU	59.01	38.55	22.20
LFU	34.00	32.23	31.23

The closeness in the performance of γ-LRU and LocalOpt, for a large range of β values, can be intuitively explained by the documents that they cache. LocalOpt retains two classes of documents: (i) recently accessed documents, because their probability of being requested is high due to the α_i's, and (ii) very popular documents, because their probability of being requested is large due to βp_i. γ-LRU also retains these two classes of documents; the former in the lower part of the cache, and the later in the upper part of the cache.

As mentioned in the introduction, researchers have proposed a number of algorithms that combine features of both LRU and LFU. LFU-DA [2] is one such algorithm that does not take document sizes into account. Hence, it is interesting to compare its performance with that of LocalOpt and γ-LRU, which also ignore document sizes. LFU-DA associates each document in the cache with a value, called eviction value, and evicts the document with the minimum such value. Whenever there is a request for a document i, the new eviction value for that document is set to $f_i = \min(f_j : j \text{ in cache}) + F_i$, where F_i is the number of times the document has been requested since it entered the cache. Because of the close connection of this algorithm to an earlier algorithm called GD-Size [8], an alternative name for it is GD-F, and this is the one we use here.

Table 4 also shows the performance of GD-F. As it is evident from the table, GD-F also manages

[13] In a different context, the authors in [3] introduce $POS(k)$, another parameterized algorithm that performs Transpose [28] in the k topmost positions of the cache, and LRU in the rest of the positions. $POS(k)$ has the same behavior with γ-LRU for the two extreme values of k and γ. However, it behaves very differently from γ-LRU for intermediate values of k and γ, and has worse performance. The problem with $POS(k)$ is that it requires an impractically large period of time to gather popular documents at the top of the cache.

to adapt to the degree of short-term (temporal) correlation versus long-term (document) popularity. Its performance is slightly worse than that of γ-LRU, and the larger the value of β, the worse GD-F does relatively to LocalOpt and γ-LRU.

3.2.1. Extreme values of γ

Let K be the size of the cache. Then, $1/K \leqslant \gamma \leqslant 1$. For $\gamma = 1$ γ-LRU coincides with LRU by definition. We will show that for $\gamma = 1/K$, γ-LRU closely resembles LFU in stationarity.

When $1/K$-LRU is used, [14] documents that get requested while in the cache swap positions with the ones that are above them, and documents that get requested while being outside the cache swap position with the document at the bottom of the cache.

Let the total population of documents in the universe be N. Index the documents in decreasing order of their popularity. Denote by S_n the cache occupancy just prior to the nth request. S_n is a Markov chain. Depending on the eviction scheme used, this chain may have up to $\binom{N}{K}K!$ states, corresponding to all possible orderings of all possible K-tuples of the N documents.

Call by *Offline-LFU* the algorithm which knows the overall popularity of the documents in advance and always keeps the most popular documents in the cache. For long request sequences, the empirical distribution that LFU uses is quite close to the actual distribution, and LFU and Offline-LFU have nearly the same behavior. When Offline-LFU is used, $S_n = (1, 2, \ldots, K)$ for all n. We will analyze $1/K$-LRU and show that this is the highest probability state of the corresponding Markov chain, and that most of the other states have significantly lower probability.

Let $1/K$-LRU be the eviction scheme used. In this case, S_n is time-reversible [15] [19] and it is easy to find the steady-state distribution in closed-form. Denote by $\pi_{(i_1,i_2,\ldots,i_k)}$ the steady state probability of state $i = (i_1, i_2, \ldots, i_K)$, i.e., document i_1 occupies position 1 in the cache, i_2 occupies position 2 in the cache and so on. Further, denote by P_{ij} the transition probability from state i to state j. One may verify that the condition

$$\pi_i P_{ij} = \pi_j P_{ji} \tag{8}$$

is satisfied by the stationary distribution

$$\pi_{(i_1,i_2,\ldots,i_K)} = C \cdot p_{i_1}^K p_{i_2}^{K-1} \cdots p_{i_K}, \tag{9}$$

where C is a normalizing constant. For example, if $\pi_i = \pi_{(1,2,\ldots,K-1,K)}$ and $\pi_j = \pi_{(1,2,\ldots,K-1,K+1)}$, then $P_{ij} = p_{K+1}$ and $P_{ji} = p_K$, and Eq. (8) holds.

Since the p_i's are decreasing as i increases, the highest probability state is $(1, 2, \ldots, K)$. Also, since p_i's are Zipf-like distributed, the probability of the states where unpopular documents reside in the cache is very low.

3.2.2. On the optimal value of γ

It is interesting to further investigate the behavior of γ-LRU as γ varies from $1/K$ to 1. Let γ^* denote the optimal value of γ with respect to hit rate given the history h, α_i's, β, p_i's, and the cache size K. We first argue that $\gamma^* \leqslant h/K$.

To see this, name the positions in the cache $1, 2, \ldots, K$ moving from bottom to top. Recall that $Z_{n,i}$ is a random variable indicating the document occupying position i at time n. Also, recall that in stationarity the hit rate of a replacement policy equals $P(H_n = 1) = \sum_{i=i}^{K} P(X_n = Z_{n,i})$, where H_n equals 1 if there is a hit at time n and zero otherwise, and X_n represents the document requested at time n.

If $\gamma K \geqslant h$ then all documents stay in the cache for at least h time slots, after their last request before they leave the cache. Since $\alpha_{h+i} = 0$ for all $i \geqslant 1$, these replacement policies take full advantage of short-term temporal correlation. Thus, $P(H_n = 1) = \sum_{j=1}^{h} a_j + \beta \sum_{i=1}^{K} P(Y_n = Z_{n,i})$ where Y_n, $n \geqslant 1$, is independent and identically distributed according to $\{p_i\}$.

The sum $\sum_{i=1}^{K} P(Y_n = Z_{n,i})$ is maximized by Offline LFU, since Offline LFU places in the cache the K most popular documents. Also, the more popular documents a replacement policy places in the cache, the larger this sum is. Now, the smaller the value of

[14] $1/K$-LRU is analogous to Transpose [28], which is an algorithm for list search.

[15] When γ is larger than $1/K$ the corresponding Markov chain is not time-reversible.

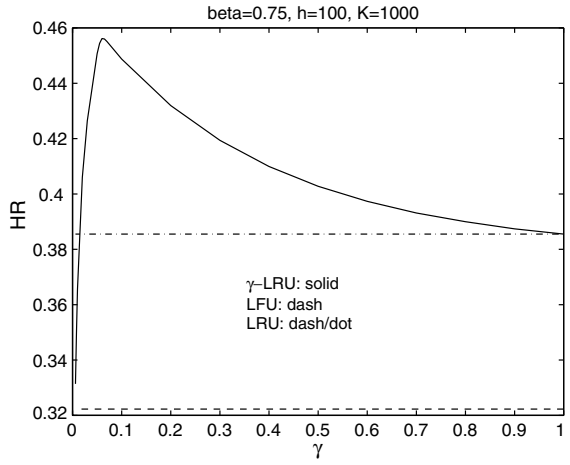

Fig. 6. Hit rate of γ-LRU as a function of γ under the second synthetic trace ($\beta = 0.75$) of Table 3. $\frac{h}{k} = 0.1$.

γ, the more popular are the documents in the cache. [16] Thus, this sum is larger for $\gamma K = h$ than for $\gamma K > h$, and the same is true for the hit rate.

Very small values of γ suffer from very large mixing times of the corresponding Markov chains. In practice, this means that it takes too much time for the cache to be populated with the most popular documents in the upper part of the cache. We choose to use γ values close to h/K since we find their performance to be close to optimal, see for example Table 4, and the corresponding mixing times to be small.

Fig. 6 plots the hit rate of γ-LRU under the second synthetic trace used in Table 3 ($\beta = 0.75$), as a function of γ. For $\gamma = 1$ the hit rate equals that of LRU, and for small γ it is close to LFU's hit rate. Also, the maximum hit rate is achieved for a value of γ less than h/K while for $\gamma = h/K$ the performance is very close to optimal.

3.3. Performance under real traces

In this section we study the performance of γ-LRU under real web traces from *NLANR* [22]. We also compare the hit rate obtained for real traces and their synthetic counterparts under various algorithms, to investigate whether the request model captures the essential properties of real traces as they relate to web caching.

We present results from six traces from three different sites, recorded in May 2001, January 2002 and July 2003. For the generation of the synthetic traces we use the procedure presented in Section 2. The history is set to 1000, and the parameters α_i's, β, and p_i's are inferred from the traces. The cache size is set to 5% of the total number of distinct requests in each trace, and the value of γ is set to 0.1 which is close to h/K in all the simulations. Finally, the cache is warmed up in all simulations. We don't take document size into account for now. Section 3.4 addresses this issue.

Table 5 reports the total requests and total distinct requests of each of the traces, as well as the cache size used in each case.

Table 6 presents the hit rates of LocalOpt, γ-LRU, GD-F, LRU and LFU under the two May traces and their synthetic counterparts. Tables 7 and 8 present the same quantities for the January and July traces. As expected, LocalOpt is superior to all other algorithms. Also, γ-LRU is competitive against GD-F, LRU and LFU, while GD-F is better than LRU and LFU. Results are similar for all the other daily NLANR traces that we tried.

The hit rates obtained for the real traces and their synthetic counterparts are close. LocalOpt performs better under synthetic traces since it is designed to exploit the statistical characteristics that the synthetic traces exactly possess. The small differences in the hit rates are likely to be due to (i) the insufficient statistics for the unpopular documents, (ii) the inter-document correlations present

Table 5
Trace characteristics and cache size

Trace	Total requests	Distinct requests	Cache size
May 21st, 2001, PA	558826	275136	13500
May 23rd, 2001, PA	508980	261480	13000
January 19th, 2002, PA	512458	256486	12800
January 20th, 2002, PA	507827	238785	11900
July 10th, 2003, SD	434604	265674	13200
July 10th, 2003, RTP	681383	387828	19400

[16] We do not have a formal proof for this statement. Intuitively, larger values of γ cause larger perturbations whenever unpopular documents are requested. This intuition is verified by simulations, see for example Fig. 6.

Table 6
Hit rates under May 2001 traces and their synthetic counterparts

Scheme	May 21st, 2001, PA		May 23rd, 2001, PA	
	HR (real) (%)	HR (synthetic) (%)	HR (real) (%)	HR (synthetic) (%)
LocalOpt	37.8	41.1	38.8	40.2
γ-LRU	35.3	36.6	35.1	36.1
GD-F	33.3	34.4	33.0	34.2
LRU	32.2	32.1	31.7	31.4
LFU	29.4	29.1	28.5	28.1

Table 7
Hit rates under January 2002 traces and their synthetic counterparts

Scheme	January 19th, 2002, PA		January 20th, 2002, PA	
	HR (real) (%)	HR (synthetic) (%)	HR (real) (%)	HR (synthetic) (%)
LocalOpt	37.8	41.2	39.6	42.5
γ-LRU	34.9	36.5	36.6	37.7
GD-F	32.3	32.4	34.4	34.5
LRU	30.7	31.1	32.7	32.4
LFU	28.9	28.7	31.1	30.8

Table 8
Hit rates under July 2003 traces and their synthetic counterparts

Scheme	July 10th, 2003, SD		July 10th, 2003, RTP	
	HR (real) (%)	HR (synthetic) (%)	HR (real) (%)	HR (synthetic) (%)
LocalOpt	33.4	35.5	39.3	41.2
γ-LRU	31.8	33.3	37.1	39.0
GD-F	30.3	30.2	34.6	34.6
LRU	29.3	29.2	34.2	34.2
LFU	25.2	25.0	32.5	32.4

in the real traces that are not modeled, and (iii) the fact that the values of the α_i's are the same for all the documents (see Section 2.4 for a related discussion). These modeling decisions do not introducing significant inaccuracies, yet they simplify the model substantially.

3.4. Implementation concerns

To be able to use γ-LRU in practice, we must address some implementation concerns. First, the size of the documents should be taken into consideration. It is straightforward to extend γ-LRU to take size into account, by making γ depend on the document size. Let $\gamma(S)$-LRU be such an extension where S denotes size and $\gamma(S)$ is a decreasing function of S. [17]

We use a specific function $\gamma(S)$ and calculate the performance of $\gamma(S)$-LRU using real traces. The goal is to show that $\gamma(S)$-LRU is competitive against other high performance replacement schemes that take into account the size and the frequency of use in their eviction decision, in addition to recency of use.

We avoid doing an elaborate design of $\gamma(S)$, or optimizing its design. For $S = 30$ KB, $\gamma(S)$ is set to 0.1. (We use 0.1 as the typical γ value. This value works well for all the conducted simulations, since the performance of γ-LRU under real traces is good for a wide range of γ values around h/K.) To penalize large documents, as S grows from 30KB to 250 KB, γ drops quadratically to 0.05. Documents larger than 250KB are never cached. As S decreases from 30KB to 1KB, γ increases linearly to 0.2 and retains this value for smaller document sizes.

We compare $\gamma(S)$-LRU to various variants/extensions of the Greedy-Dual-Size (GD-Size) algorithm [8]. In particular, we compare it to Greedy-Dual-Size-Frequency-Connection (GD-SFC), Greedy-Dual-Size-Frequency (GD-SF), and Greedy-Dual-Frequency (GD-F, or alternatively, LFU-DA [2]). These algorithms associate each document in the cache with a value, called eviction value, and evict the document with the minimum such value. Whenever there is a request for a document i, the new eviction value for that document is set to

$$f_i = \min(f_j\text{: } j \text{ in cache}) + G_i;$$

G_i equals $(C_i \cdot F_i)/S_i$ for GD-SFC, F_i/S_i for GD-SF, and F_i for GD-F, where C_i is an estimate of the latency for connecting to the corresponding server computed in the same manner as in [33], F_i is the number of times the document has been requested

[17] Taking into account the size of a document independently of its popularity is justified by the evidence that there is no correlation between them [7].

since it entered the cache, and S_i is the document size. Note that the quantity $\min(F_j: j \text{ in cache})$ is increasing in time and is used to take into account the recentness of a document. Indeed, whenever a document is accessed, its eviction value is increased by the currently minimum eviction value. Thus, the most recently used documents tend to have larger eviction values.

The traces we use are taken from (NLANR) [22]. We only present results from a daily trace of length 500,000, since there are no significant differences in the results obtained from the various traces. We only simulate requests with a known reply size.

The performance criteria used are three:

(i) the hit rate (HR), which is the fraction of client-requested URLs returned by the proxy cache,
(ii) the byte hit rate (BHR), which is the fraction of client requested bytes returned by the proxy cache, and
(iii) the latency reduction (LR). which is the reduction of the waiting time of the user from the time the request is made till the time the document is received (download latency), over the sum of all download latencies.

For each trace, the HR, the BHR and the LR are calculated for a cache of infinite size. Then, they are calculated for a cache of size 0.5%. 5%, 10%, and 20% of the maximum size required to avoid any evictions. This size is around 2GB.

Fig. 7 presents the ratio of the HR of $\gamma(S)$-LRU, GD-SFC, GD-SF, and GD-F over the HR achieved by an infinite size cache. The figure also presents the HR of LRU. The performance of $\gamma(S)$-LRU, GD-SFC, and GD-SF are pretty close, while GD-F and LRU do worse.

Fig. 8 presents the ratio of the BHR of $\gamma(S)$-LRU, GD-SFC, GD-SF, and GD-F over the BHR achieved by an infinite size cache. The figure also presents the BHR of LRU. The performance of $\gamma(S)$-LRU is superior to the performance of GD-SFC and GD-SF for small cache sizes. This is expected since these schemes have a strong bias against large documents even when these documents are popular. (The sizeably worse performance of GD-SFC for small cache sizes is due to its bias against some large, popular documents whose connection-latency is very small.) On the other had, GD-F is quite close to $\gamma(S)$-LRU since it does not penalize large documents.

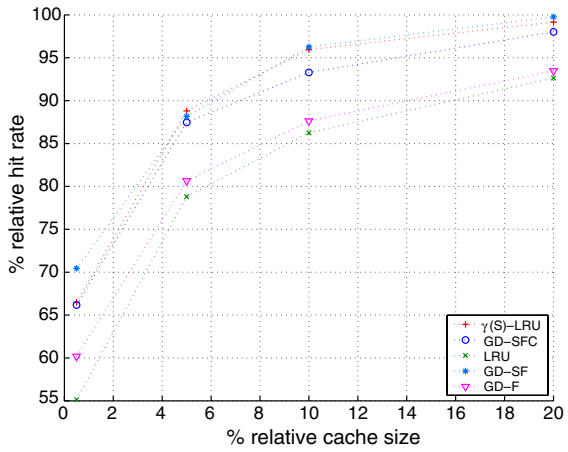

Fig. 7. Hit rate comparison between various schemes.

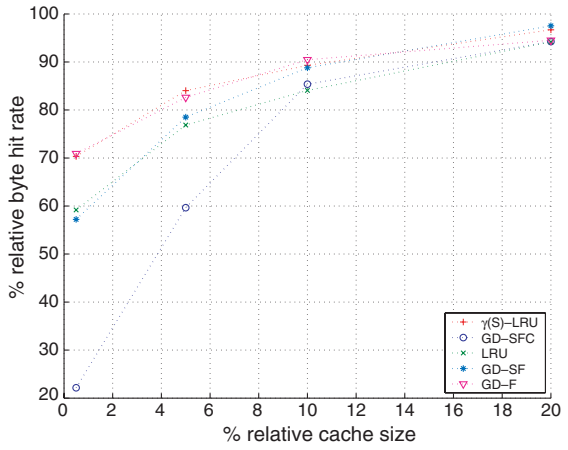

Fig. 8. Byte hit rate comparison between various schemes.

Fig. 9 presents the ratio of LR of $\gamma(S)$-LRU, GD-SFC, GD-SF, and GD-F over the LR achieved by an infinite cache. The figure also presents the LR of LRU. The performance of the schemes is close, with $\gamma(S)$-LRU, GD-SFC, and GD-SF doing a bit better than the rest.

GD-SF is known to perform quite well with respect to HR, while GD-F is known to perform quite well with respect to BHR. Figs. 7 and 8 show

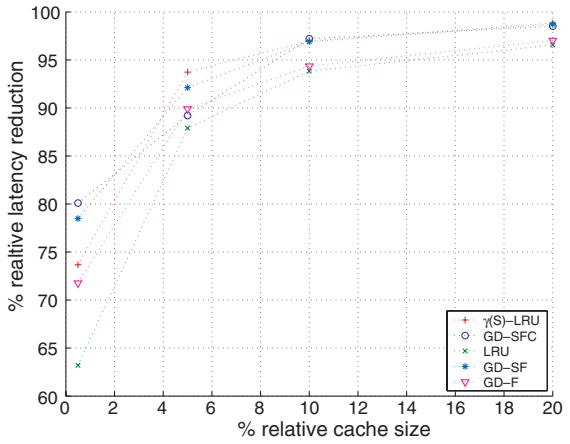

Fig. 9. Latency reduction comparison between various schemes.

that this is indeed the case, and more importantly, show that γ-LRU performs competitively with respect to both HR and BHR.

The difference in the performance of the five schemes decreases rapidly as the relative cache size increases. Indeed, it has been observed that for very large caches, the performance of different replacement schemes is similar [8,25]. The model introduced in Section 2 provides the following explanation for this phenomenon: (i) due to the Zipf-like distribution of the popularity of web documents, the fraction of documents that corresponds to most of the hits is small enough to fit in the cache, and (ii) due to the rapid decrease of temporal correlations (Fig. 3), large caches store documents long enough to fully exploit these correlations.

Another implementation concern is the data structure required to implement γ-LRU. LRU uses a linked list to maintain the order of the documents in the cache. However, using a simple linked list for γ-LRU is not efficient. Indeed, linked lists are efficient when insertion and deletion operations take place at the top and the tail of the list only. In γ-LRU it is required to perform insertions at arbitrary places, which has complexity $O(\log n)$. [18] To maintain constant-time complexity, we propose to only allow insertions to happen at predefined positions properly distributed along the length of the list. Each list entry will have a pointer pointing to the predefined position that the entry should jump to, if it is accessed. It is easy to see that whenever a jump takes place, only two of these pointers need to be updated.

The focus of this work is on the modeling of web trace correlations. The γ-LRU algorithm serves to illustrate the implications of these correlations on designing replacement policies. Since it is not our goal to fully investigate the performance of γ-LRU as a replacement scheme in practice, we refrain from further discussion on $\gamma(S)$-LRU with predefined insertion positions.

[18] Note that algorithms like GD-SFC, GD-SF, and GD-F also have a complexity of $O(\log n)$ [8].

4. Conclusions

In this paper we have proposed a model that precisely captures the degrees to which temporal correlations and document popularity influence web trace requests. We also described a mathematical procedure to extract the parameters of the model from real traces. The model was used to generate synthetic traces of arbitrary length exhibiting any degree of correlation, and provide guidelines for designing efficient replacement algorithms.

We formulated the problem of obtaining an optimal replacement algorithm with respect to hit rate as a Markov decision process, and obtained an online replacement algorithm that is optimal conditioning on the state of the Markov process. We also introduced γ-LRU, a class of practicable replacement policies parameterized by γ with LRU and LFU obtained for extreme values of γ. Finally, we showed that γ-LRU performs close to the optimal policy, under any degree of temporal correlation present in the request sequence, and extended the algorithm to take into account documents of varying sizes.

References

[1] V. Almeida, A. Bestavros, M. Crovella, A. de Oliveira, Characterizing reference locality in the WWW, in: Proceedings of IEEE PDIS, 1996.

[2] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, T. Jin, Evaluating content management techniques for Web proxy caches, in: Proceedings of the Second Workshop on Internet Server Performance, May 1999.
[3] A. Tenenbaum, R. Nemes, Two spectra of self-organizing sequential search algorithms, SIAM Journal of Computing 11 (3) (1982).
[4] P. Barford, M. Crovella, Generating representative Web workloads for network and server performance evaluation, in: Proceedings of the ACM SIGMETRICS Conference, June 1998.
[5] D. Bertsekas, J. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, MA, 1996.
[6] L. Breslau, P. Cao, L. Fan, G. Philips, S. Shenker, Web caching and Zipf-like distributions: evidence and implications, in: Proceedings of IEEE INFOCOM, New York, 1999.
[7] M. Busari, C. Williamson, Prowgen: a synthetic workload generation tool for simulation evaluation of Web proxy caches, Computer Neworks 38 (6) (2002) 779–794.
[8] P. Cao, S. Irani, Cost-aware www proxy caching algorithms, in: Proceedings of USENIX Symposium on Internet Technologies and Systems, Monterey, CA, December 1997.
[9] M. Charikar, S. Chaudhuri, R. Motwani, V. Narasayya, Towards estimation error guarantees for distinct values, in: Proceedings of the 19th ACM Symposium on Principles of Database Systems (PODS), 2000.
[10] L. Cherkasova, G. Ciardo, Role of aging, frequency, and size in Web cache replacement policies, in: Proceedings of HPCN Europe, 2001.
[11] L. Cherkasova, G. Ciardo, Characterizing temporal locality and its impact on Web server performance, in: Proceedings of ICCCN, October 2000.
[12] E. Coffman, P. Denning, Operating Systems Theory, Prentice-Hall, Englewood Cliffs, NJ, 1973.
[13] R. Durrett, Probability: Theory and Examples, second ed., Duxbury Press, 1996.
[14] R. Fonseca, V. Almeida, M. Crovella, B. Abrahao, On the intrinsic locality properties of Web reference streams, in: Proceedings of IEEE INFOCOM, 2003.
[15] P. Haas, L. Stokes, Estimating the number of classes in a finite population, Technical Report RJ 10025, IBM Almaden Research Center, San Jose, CA, 1996.
[16] S. Jin, A. Bestavros, Sources and characteristics of Web temporal locality, in: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2000.
[17] S. Jin, A. Bestavros, Popularity-aware greedydual-size algorithm for Web access, in: Proceedings of the IEEE ICDCS Conference, April 2000.
[18] S. Jin, A. Bestavros, Greedydual* Web caching algorithm: exploiting the two sources of temporal locality in Web request streams, in: Proceedings of the 5th International Web Caching and Content Delivery Workshop, Lisbon, Portugal, May 2000.
[19] F.P. Kelly, Reversibility and Stochastic Networks, Wiley, New York, 1979.
[20] D. Lee, J. Choi. S. Noh, S.L. Min, Y. Cho, C.S. Kim, On the existence of a spectrum of policies that subsumes the LRU and LFU policies, in: Proceedings of the ACM SIGMETRICS Conference, May 1999.
[21] A. Mahanti, D. Eager, C. Williamson, Temporal locality and its impact on Web proxy cache performance, Performance Evaluation 42 (2–3) (2000) 187–203.
[22] NLANR cache access logs, Available from <ftp://ircache.nlanr.net/Traces/> accessed January 2002.
[23] V. Padmanabhan, J. Mogul, Using predictive prefetching to improve world wide Web latency, ACM SIGCOMM Computer Communication Review 26 (3) (1996).
[24] V. Phalke, B. Gopinath, An interreference gap model for temporal locality in program behavior, in: Proceedings of ACM SIGMETRICS, 1995.
[25] K. Psounis, B. Prabhakar, Efficient randomized Web-cache replacement schemes using samples from past eviciton-times, IEEE/ACM Transactions on Networking 10 (4) (2002) 441–454.
[26] M. Puterman, Markov Decision Processes, Discrete Stochastic Dynamic Programming, Wiley Series in Probability and Mathematical Statistics, Wiley, New York, 1994.
[27] B. Ramakrishna Rau, Properties and applications of the least-recently-used stack model, Technical Report CSL-TR-77-139, Stanford University, 1977.
[28] R. Rivest, On self-organizing sequential search heuristics, Communications of the ACM 19 (2) (1976) 63–67.
[29] S. Ross, Introduction to Stochastic Dynamic Programming, Academic Press, New York, 1983.
[30] G. Shedler, C. Tung, Locality in page reference strings, SIAM Journal of Computing 1 (3) (1972) 218–241.
[31] A. Silberschatz, P. Galvin, Operating System Concepts, fifth ed., Addison-Wesley Longman, Reading, MA, 1997.
[32] D. Starobinski, D. Tse, Probabilistic methods for Web caching, Performance Evaluation 46 (2–3) (2001) 125–137.
[33] R. Wooster, M. Abrams, Proxy caching that estimates edge load delays, in: Proceedings of the 6th International World Wide Web Conference, Santa Clara, CA, April 1997.

Konstantinos Psounis is an assistant professor of Electrical Engineering and Computer Science at the University of Southern California. He received his first degree from the department of Electrical and Computer Engineering of National Technical University of Athens, Greece, in June 1997, the M.S. degree in Electrical Engineering from Stanford University, California, in January 1999, and the Ph.D. degree in Electrical Engineering from Stanford University in December 2002. He researches probabilistic, scalable algorithms and methods for Internet-related problems. He has worked mainly on web caching and performance, web traffic modeling, congestion and flow control, and performance prediction of IP networks and web farms. He has been a Stanford Graduate Fellow throughout his graduate studies. He has received the best-student National Technical University of Athens award for graduating first in his class.

An Zhu received her BS degree in Computer Science (high honor) and Mathematics from the University of Maryland at College Park in 1999. She is currently a Ph.D. student in the Computer Science department at Stanford University. Her research interests are the design and analysis of algorithms, with emphasis on approximations, online computations, and randomized algorithms, as well as related complexity theory. She was awarded the Bell-labs GRPW (Graduate Research Program for Women) fellowship and the Stanford Graduate Fellowship (SGF).

Balaji Prabhakar is an Associate Professor of Electrical Engineering and Computer Science at Stanford University. He is interested in network algorithms (especially for switching, routing and bandwidth partitioning), wireless networks, web caching, network pricing, information theory and stochastic network theory. He has been a Terman Fellow at Stanford University and a Fellow of the Alfred P. Sloan Foundation. He has received the CAREER award from the National Science Foundation, the Erlang Prize from the INFORMS Applied Probability Society, and the Rollo Davidson Prize for young scientists who have contributed to probability or its applications.

Rajeev Motwani is a Professor of Computer Science at Stanford University, where he also serves as the Director of Graduate Studies. He obtained his Ph.D. in Computer Science from Berkeley in 1988. His research has spanned a diverse set of areas in computer science, including databases and data mining, web search and information retrieval, robotics, computational drug design, and theoretical computer science. He has written two books—Randomized Algorithms published by Cambridge University Press in 1995, and an undergraduate textbook published by Addison-Wesley in 2001. He has received the Godel Prize, the Okawa Foundation Research Award, the Arthur Sloan Research Fellowship, the National Young Investigator Award from the National Science Foundation, the Bergmann Memorial Award from the US–Israel Binational Science Foundation, and an IBM Faculty Award. He is a Fellow of the Institute of Combinatorics and serves on the editorial boards of SIAM Journal on Computing, Journal of Computer and System Sciences, and IEEE Transactions on Knowledge and Data Engineering.