Empirical Investigation of the Markov Reference Model

Vincenzo Liberatore[1]

March 24, 2001

[1]Department of Computer Science, Hill Center, Rutgers University, New Brunswick, NJ 08903 (liberato@cs.rutgers.edu). URL:http://www.cs.rutgers.edu/~liberato/. Supported by an NSF Career Development Award CCR-9501942, NATO Grant CRG 960215, NSF/NIH Grant BIR 94-12594-03-CONF and an Alfred P. Sloan Research Fellowship.

Abstract

We consider the Markov Reference Model (MRM) of program behavior. MRM generates sequences of page requests according to a discrete-time homogeneous Markov chain. We invalidate MRM for instruction traces by means of a cost independent test that is a specialization of the χ^2 test used in non-parametric statistics. We also find that observed traces have less entropy than MRM traces, which suggests that MRM does not capture the locality and structure of observed traces. Finally, we verify our results in terms of LRU page fault cost.

1 Introduction

Computer system design and performance critically depend on the *behavior* of running programs. Program behavior affects the design of efficient hardware architectures [HP96] and operating systems [Den80, Fer76]. For example, the workload generated by real programs determines the design parameters of efficient caching systems [HP96, Prz90]. The behavior of programs would be clearly identified by a *model* of program behavior. Consequently, program behavior models are crucial tools for the design and analysis of efficient computer systems [Spi77].

Markov Reference Model In general, several activities of a program affect performance: memory references, I/O, interrupts, resource reservation, and so on. However, the referencing behavior is so important in virtual memory systems that it is called "program behavior" *tout court* in the literature [Fer76, Spi77]. In this paper, a model of program behavior will be subjected to extensive empirical investigation. The *Markov Reference Model* (MRM) generates a sequence of page requests according to a Markov chain where each state corresponds to a program page. MRM is a very natural model of program behavior, and so it is not surprising that it has a long history. To the best of our knowledge, MRM was first mentioned by Aho *et al.* [ADU71] and studied by various authors [FG74, ST72]. MRM subsumes simpler models as special cases. For example, it includes the *Independent Reference Model* (IRM). IRM generates pages according to a common stationary probability distribution [ADU71, CD73, Den80, FW74, Smi76]. An important feature of MRMs, but not of IRMs, is that they can exhibit the phase transition behavior that has been observed in real traces [Cur75, CV76, Den80]. MRM has paging algorithms that achieve optimal and approximate page fault rates [KPR92].

Our Results In this paper, we will show that MRM is not a valid model of program behavior for the instruction traces generated by most of our benchmarks. MRM invalidation poses several methodological and technical questions, and we will discuss them below. We will argue that behavior models should be validated according to their intrinsic statistical properties. We use the χ^2 test from non-parametric statistics and we modify it into a novel algorithm for MRM validation. We are not aware of any previous experimental work in computer system analysis that uses the χ^2 test or a modification of the χ^2 test. We believe that our algorithm could also be exploited to analyze traces arising from Internet traffic and database logs. We will also provide an information-theoretical rejection of MRM by showing that MRMs possess more entropy than observed traces. To the best of our knowledge, no information-theoretical analysis of paging traces had been previously attempted. We will also review the implications of MRMs in the context of prefetching and path profiling, and detail those points below.

Empirical Invalidation Experimental invalidation of MRMs gives rise to a number of serious methodological considerations. Models are often validated only with respect to a small number of features that are important in an application. A model that is adequate only for some statistical feature is called a *functional model* because it serves a particular function in a certain computational framework [Spi77]. Examples include validation against page fault rates, working set behavior [Spi77], bandwidth requirements [HS95, HS96], and overall workload characteristics [MDO94]. The obvious problem with functional validations is that it is very hard to generalize them to a new and different setting. The problem is becoming more and more acute due to the proliferation of paging systems with different characteristics [AACS87, Fra96, FZ97, HP96, You94] and of page replacement strategies [AACS87, CN98, FR97, IKP96, KL98, KMRS88, KPR92, Tan92]. On the

other hand, program behavior is identical across several memory systems and operating systems. The stability of program behavior across platforms is a direct consequence of the concept of virtual memory system, which decouples program construction from the explicit management of memory [Den80]. Consequently, programs will exhibit the same behavior independent of architectural and operating system details. Since programs can be executed on a variety of platforms, functional models are becoming too limited in scope. Instead, models should be validated according to the statistical adequacy of generated traces as compared with observed traces. For example, MRM should not be validated against the paging characteristics of a particular system. MRM should be verified only according to its intrinsic characteristics. The crucial property of MRMs is expressed by the *Markov property*: the probability that page i is requested at time t is independent of all previous page requests except for the page requested at step $t-1$ [Fel67, Fre83]. Therefore, MRM should be validated by checking a condition of statistical independence or lack of association. We will invalidate MRM by resorting to a purely statistical test to check statistical independence. The χ^2 test is perhaps the most commonly used way to test for evidence of association. The test was proposed by Karl Pearson [Pea00], and is recommended in several standard textbooks on statistics [Cra58, DM83, Edw73, Fis70, Hal65, McN69, McP90, MGB74, Sie56, WL53]. We will modify the χ^2 test in order to obtain statistically significant results in our experiments, and we will show that MRMs should be ruled out for most instruction traces. Our modification of the χ^2 test provides purely statistical results and does not depend on any caching performance parameter. We remark that our modification of the χ^2 test can also be applied to the analysis of MRMs for Internet traffic and database logs.

Functional Validation We will ascertain that MRMs are not statistically valid models. Furthermore, we will show that MRMs grossly mispredict the LRU page fault rate. We will also find that MRM cost curves do not exhibit the non-linear behavior of real traces. The conclusion is surprising because theoretically MRMs could exhibit phase transition behavior [Cur75, CV76, Den80]. We conclude that MRMs are not statistically valid in general and also that they are not functionally valid with respect to LRU page fault rates.

Models and Prefetching We now discuss the relations between models of program behavior and prefetching. *Prefetching* is the operation of loading a page before it is requested. Prefetching has been extensively studied as a method to improve performance [CKP91, CB92, PZ91, RL92, Smi78, TE93]. Prefetching can be executed only if future requests are known. Much research has been devoted to the problem of predicting future requests from past accesses. Vitter *et al.* use compression techniques to predict future requests [VK96]. In the context of file systems, Madhyastha *et al.* classify access patterns by training a hidden Markov model [MR97]. Palmer *et al.* train an associative memory to recognize access patterns [PZ91]. We notice that all those prefetching systems predict future behavior from the past, and that prediction involves implicitly or explicitly the assumption of a model of program behavior. For example, Madhyastha *et al.* assume that references are generated by a hidden Markov chain [MR97] and Vitter *et al.* that references are generated by a probabilistic finite automaton [VK96]. Models of program behavior lie at the heart of prefetching systems. A strongly validated model would effectively predict future program behavior and help prefetching. We will show that our statistical rejection of MRM can be turned into a prediction method. Our predictions are based on the last two requested pages, and achieve better prediction rates than a method based on MRMs.

Behavior Complexity The principle of locality states that program behavior is loosely predictable. If a page is requested now, it will probably be requested in the near future [HP96]. It follows that program traces are not chaotic sequences of page requests, but follow some regular pattern. A model of program behavior should capture the lack of randomness of real traces. We propose to measure the lack of randomness of program traces and of models by measuring their entropy. High entropy corresponds to random behavior and low entropy to regular behavior [GPS94]. The entropy of models can often be computed with analytical formulas. The entropy of traces can be measured by their compressibility under universal compression algorithms, as, for example, Lempel-Ziv (LZ) [ZL77].

We will show that both IRM and MRM exhibit significantly more entropy than the corresponding observed traces in almost all cases. Consequently, we doubt that IRM and MRM could model the structure of observed traces. The information source test is much simpler and faster than the χ^2 test, However, the information source test provides conclusions that match those of the more rigorous χ^2 test. Furthermore, we observe that the entropy gap is a predictor of relative LRU page fault cost. For these reasons, we propose the information source test as a simple, fast, and general method to validate models of program behavior.

MRMs and Path Profiling The Markov property of MRMs states that the probability of requesting a page at step t depends only on the page requested at step $t-1$. Consider a representation of the Markov chain in terms of a directed graph $G = (V, E)$, where the set V of vertices is the set of states and each edge $(i, j) \in E$ is labeled with the probability of transition from state i to state j. *Edge profiling* determines the number of times an edge is crossed during a program execution [Fis81]. Edge profiling would be justified if the Markov property held. Although edge profiling is inexpensive, it does not always establish correctly the frequencies of execution of paths, and recent research has been devoted to *path profiling*, which counts the number of times each path is executed [BL96]. The present paper rejects MRM and provides a statistical justification to the interest for profiling approaches other than edge profiling.

Organization of the Paper In §2, we will define terms and notation. In §3, we will describe the results of the invalidation experiments for MRMs. In §4, we will instantiate the χ^2 test to obtain meaningful statistics for our experiments. We will also show a prefetching strategy that achieves better prediction rates than MRM. In §5, we will show that MRMs and IRMs almost always generate traces that are substantially more complex in an information-theoretical sense than observed traces. In §6, we compare the cost of MRM and IRM traces under the page fault cost model and the LRU page replacement strategy.

2 Preliminaries

A running program generates a sequence σ of references to virtual memory pages. We will denote the page requested at step t as σ_t. The length of σ will be denoted by m and the number of distinct pages requested in σ by n. The subject of this papers are models of program behavior. We will consider two models of program behavior. The first is the Independent Reference Model (IRM), that generates σ_t according to a stationary reference distribution $P\{\sigma_t = i\} = p_i$ for all t's. The second model is the Markov Reference Model (MRM), that generates sequences of page requests from a discrete-time homogeneous Markov chain, whose states corresponds to program pages. Notice that MRM include IRM as a special case. The *Markov property* of Markov chains

states that for any page j, for any non-negative integers $t_1 < t_2 < \ldots < t_l < t$ and for any pages $p_1, p_2, \ldots, p_l$, it holds that [Fel67, Fre83].

$$P\{\sigma_t = j | \sigma_{t_1} = p_1, \sigma_{t_2} = p_2, \ldots, \sigma_{t_l} = p_l\} = P\{\sigma_t = j | \sigma_{t_l} = p_l\} \ .$$

Notice that both IRM and MRM are *generative* models because they immediately provide a (randomized) algorithm for generating sequences of page requests [Spi77]. Some model do not have such property and are called descriptive. We do not consider descriptive models in this paper, and we refer the interested reader to [Spi77]. Finally, we will describe the experimental set-up in appendix A.

3 Markov Reference Model

The invalidation of MRMs is based on the rejection of the following special form of the Markov property: for all $t \geq 3$,

$$P\{\sigma_t = j | \sigma_{t-1} = i, \sigma_{t-2} = h\} = P\{\sigma_t = j | \sigma_{t-1} = i\} \ , \tag{1}$$

which states that the probability that page j is requested at time t depends on the page requested at time $t-1$, but not on the page requested at time $t-2$. Since (1) is a special case of the Markov property, if (1) is rejected for a significant fraction of pages, then σ is not generated by a Markov chain. Hypothesis (1) is rejected if there is evidence of association between σ_{t-2} and σ_t. The existence of an association between σ_t and σ_{t-2} will be exploited also by our prefetching strategy, which predicts σ_t on the basis of σ_{t-1} and σ_{t-2}.

In light of the previous intentions, it is clear that a fundamental role in our statistical analysis will be played by the quantity a_{hij}, which we define as the number of times when the substring (h, i, j) occurs in σ. We also define $s_i = \sum_{h,j} a_{hij}$. We classify pages in four categories:

Sporadic pages These are pages that are referenced for less than m/n^2 times during a trace. If pages were extracted uniformly at random, then a page should be referenced m/n times on average. Sporadic pages are not very frequent and significant statistics are hard to obtain. Sporadic pages account for about .2% of all trace references.

Transition pages We will say that a non-sporadic page i is a transition page if there are h and j with $a_{hij} \geq s_i - \sqrt{s_i}$. Intuitively, a transition page i is a page that is almost always accessed from the same page h and almost always results in an access to j. Therefore, page i is in the transition between h and j. The rationale behind the choice of the exact value $s_i - \sqrt{s_i}$ will become clear in §4. Transition pages account for roughly 23% of all trace references across all traces.

Conditioned pages We will say that a non-sporadic, non-transition page i is conditioned if there is a strong association between the page h that precedes i and the page j that follows i. The exact measure of association will be defined is §4. For the time being, we just mention that the measure of association is based on the χ^2 test, the most widely used statistical test for association. Conditioned pages account for roughly 64% of all trace references across all traces.

Undetermined pages All other pages will be called undetermined pages. Undetermined pages account for less than 14% of all trace references across all traces.

Conditional pages cannot fit into the Markov chain model because future references are strongly associated with past ones. On the other hand, references to conditional pages are 64% of all trace requests, and so it is quite doubtful that the Markov chain model could fit observed traces. Moreover, the bulk of non-conditioned pages are transition pages, which can be fitted into a Markov chain model, but only in a very trivial way. In some sense, the Markov chain model is too powerful to fit well the class of transition pages. On the whole, 87% of all trace references are to pages that do not fit well into the Markov chain model. In conclusion, we strongly doubt that the Markov chain model, or any of its simplifications, could explain observed traces of instruction requests.

4 The χ^2 Test

In this section, we will use the χ^2 test to establish that 64% of all trace references are to conditioned pages. We originally found an association between σ_{t-2} and σ_t by inspection of frequency tables corresponding to some traces. However, the χ^2 test is a method that can be applied to several tables and that is more automatic than inspecting each single frequency table. We will show the main ideas of the χ^2 test in appendix B. The general χ^2 test has to be modified to provide meaningful statistics from our experiments, and we will describe our instantiation of the test. In the rest of the section, we assume that the reader is familiar either with the χ^2 test or with the contents of appendix B.

First, we define some notation. Let $\beta_{ij} = \sum_h a_{hij}/s_i$ as the maximum likelihood probability estimate of the event that there is a transition from i to j. Analogously, let $\alpha_{hi} = \sum_j a_{hij}/s_i$ be the maximum likelihood probability estimate of the event that there has been a transition from h to i in the previous step. Suppose without loss of generality and for simplicity of notation that $a_{hij} = 0$ for all $h > H$ and $j > J$. Consider the following matrix

$$A_i = \begin{pmatrix} a_{1i1} & a_{2i1} & \dots & a_{Hi1} \\ a_{1i2} & a_{2i2} & \dots & a_{Hi2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1iJ} & a_{2iJ} & \dots & a_{HiJ} \end{pmatrix} .$$

The entry of A_i in row j and column h reports the frequency of the event when there is a transition from i to j combined with a preceding transition from h to i. Let B_i be the matrix of fitted values

$$B_i = \begin{pmatrix} \alpha_{1i}\beta_{i1} & \alpha_{2i}\beta_{i1} & \dots & \alpha_{Hi}\beta_{i1} \\ \alpha_{1i}\beta_{i2} & \alpha_{2i}\beta_{i2} & \dots & \alpha_{Hi}\beta_{i2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1i}\beta_{iJ} & \alpha_{2i}\beta_{iJ} & \dots & \alpha_{Hi}\beta_{iJ} \end{pmatrix} s_i .$$

The χ^2 test should be run only when no entry in B_i is less than 1 [McP90, Sie56]. Some authors also require that at least 80% of the entries of B_i are greater than 5 [Coc54, Sie56]. If the condition $\alpha_{hi}\beta_{ij}s_i \geq 1$ is not met, then the usual suggested procedure is to merge some of the rows and columns of A_i in some intuitively meaningful way in order to achieve $\alpha_{hi}\beta_{ij}s_i \geq 1$ [McP90]. We now detail our heuristic for merging rows and columns. If α_{hi} is much smaller than 1, then the transition from h to i is a sporadic event, and it is very hard to obtain meaningful information from such a transition. Therefore, a reasonable way to join columns is to sum up together all the columns whose α_{hi} is much smaller than 1. We can view the column combination as the operation of lumping all infrequent transitions into one category of sporadic transitions. Analogously, we

combine all rows where β_{ij} is much smaller than 1. The question is how small the α_{hi}'s and β_{ij}'s should be when they are combined. We chose to combine rows and columns until the smallest value of α_{hi} and of β_{ij} is at least $1/\sqrt{s_i}$. As a consequence the smallest entry of B_i will now be at least $\alpha_{hi}\beta_{ij}s_i \geq s_i/(\sqrt{s_i})^2 = 1$, and no entry of B_i is less than 1. Notice that in a transition page, the row and column combination yields $H = J = 1$, and so the degrees of freedom are $\nu_i = (H-1)(J-1) = 0$. It is now clear that the choice of the paramenter $s_i - \sqrt{s_i}$ in the definition of transition pages is motivated by the fact that those pages could not be addressed by our choice of the χ^2 test. After row and column combination, we could obtain a matrix B_i with more than 20% of the entries less than 1. In these cases, B_i has many small values and meaningful statistics are hard to obtain, so that we classify those pages as undetermined. We summarize our algorithm as follows. Add all columns with $\alpha_{hi} < 1/\sqrt{s_i}$. If the resulting combined column has still a value of α_{hi} with $\alpha_{hi} < 1/\sqrt{s_i}$, sum up this column with the remaining column that has the smallest value of α_{hi}. Then, repeat the same steps for the rows. If B_i has at least 20% of the entries less than 5, mark page i as a special page with "small entries", and consider it undetermined.

An example of analysis for one particular trace is reported in appendix E. We report the result of the experiments on all traces in table 1. We give the percentage of trace references to sporadic, transition, conditioned, and undetermined pages for each trace. We notice a significant variation in the percentages, but the overall results are that references to conditioned pages account for a large fraction of the traces under considerations. The exception are traces 21 to 24, that corresponds to the program fgrep. In the fgrep traces, most pages are undetermined or transition pages. Therefore, we cannot rule out the MRM for fgrep. A more accurate analysis of the fgrep traces shows that those traces are dominated by transition pages and by two undetermined pages. The two undetermined pages have $H = 2$, $J = 1$ and $H = 1$, $J = 2$ respectively, and so only the latter introduces randomness in the model. We conclude that MRM could hold for some programs, but, on the whole, MRM did not apply to most of our traces.

The experimental results show an association between σ_{t-2} and σ_t. On the negative side, such result implies the invalidation of MRMs. On the positive side, it could be transformed into a criterion for predicting the next reference on the basis of the last two page requests. The prediction of next references is fundamental in the context of prefetching. In appendix C, we will show that two references significantly improve the prediction success rate on most traces.

5 The Information Source Test

In this section, we introduce a new global parameter to validate the accuracy of generating models of program behavior. The gist is to view the generating model as a "black box" that acts as a source of random values. A generating model is considered as a source of random values in $\{0, 1, \ldots, n-1\}$, that form a sequence of requested pages. The entropy of the source is a measure of compressibility and, more interestingly for our purposes, of complexity, predictivity, and randomness. The entropy is maximized when all values are extracted uniformly at random. Hence, the maximum entropy of a source is $\log n$, and it corresponds to the case of an IRM that generates all pages uniformly at random. In this case, the complexity and randomness of a trace are maximized and its predictivity is minimum. However, we would not expect traces to be generated in a perfectly random way. In fact, the principle of *locality* prevents traces from being a sequence of uniformly random values. In other words, we would expect the entropy to decrease in proportion to the regularity of the model that emits it. The main idea of the information source test is to check the entropy of a model against the entropy of the observed trace to verify if the model characterizes the locality and the lack of randomness of the trace. The main advantage of the information source test is that it gives

Trace	n	m	Conditioned	Transition	Undetermined	Sporadic
1	84	982843	84.22%	8.38%	7.26%	.14%
2	88	1270694	91.92%	2.44%	5.5%	.14%
4	84	974130	84.34%	8.35%	7.17%	.14%
5	70	1261116	91.48%	4.94%	3.41%	.17%
6	84	977748	84.23%	8.39%	7.26%	.12%
7	78	1732163	98.85%	.36%	.54%	.16%
8	79	972917	98.55%	.38%	1.01%	.06%
9	76	688059	98.95%	.92%	.03%	.1%
10	78	2132159	99.26%	.59%	.02%	.13%
11	79	1278003	98.33%	.57%	1.03%	.07%
12	76	888055	98.4%	1.43%	.05%	.12%
13	31	309518	40.37%	40.32%	18.87%	.44%
14	31	409514	40.38%	40.33%	18.74%	.55%
15	32	25738	53.96%	44.75%	.83%	.46%
16	32	48489	53.89%	45.29%	.47%	.35%
17	27	363797	65.01%	29.28%	5.5%	.21%
18	21	3340855	52.01%	47.28%	.62%	.09%
19	27	463809	65.34%	28.77%	5.63%	.26%
20	21	3000458	52.2%	47.02%	.69%	.09%
21	22	1932966	.69%	32.8%	66.34%	.17%
22	22	2076354	0%	66.22%	33.01%	.77 %
23	22	1930525	.68%	32.8%	66.34%	.18%
24	22	2069211	0%	33.01%	66.21%	.78%
25	34	1991970	76.15%	15.67%	8.08%	.1%
Avg	51	1296712	63.72%	22.51%	13.54%	.23%

Table 1: Classification of page references as references to one of the four categories of classes. For each trace, n is the number of distinct pages accessed in the trace, m is the trace length. The last four columns report the percentage of requests for conditioned, transition, undetermined, and sporadic pages.

a measure of agreement between model and traces even under the assumption of non-stationary behavior. Moreover, it gives a more comprehensive measure than that expressed by (1), because we consider the trace as a whole rather than a collection of subsequences of length 3. However, the information source test also has some disadvantages. A first drawback of the information source test is that trace entropy could be hard to estimate, as we will discuss later on. A second drawback is that even if we had the exact trace entropy and if we ascertain that the trace entropy and the model entropy are roughly equal, then we could only conclude that the model and the trace exhibit a comparable "amount of structure", but not that the structure of the model is similar to that of the trace. A first problem is to measure the entropy of models. A conceptual hurdle is how to measure the "entropy" of a finite trace.

Consider first the problem of evaluating the entropy of a model. The entropy of an analytical model can often be estimated directly. For example, the entropy of an IRM that extracts page i with probability p_i is $H^I = \sum_{i=1}^{n} p_i \log(1/p_i)$ bits. Consider now MRM, and assume that the Markov chain is ergodic. Let p_i be the steady-state probability of state i and p_{ij} the probability of transition from i to j. The entropy of an ergodic Markov chain is [GPS94]

$$H^M = \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_{ij} \log \frac{1}{p_{ij}} \text{ (bits)} . \tag{2}$$

Equation (2) has a simple intuitive justification. If we are in state i, then we need $\sum_{j=1}^{n} p_{ij} \log(1/p_{ij})$ bits to communicate the next state. Therefore, the average amount of information transmitted with each page request in the steady state is given by (2).

Thus, we have estimated the entropy of two conceptual models of program behavior. A more philosophically challenging question is to estimate the "entropy" of a finite trace. First of all, it is not appropriate to talk about the entropy of a finite sequence. The more accurate notion is that of entropy of the random process that generates traces [GPS94]. However, an upper bound to trace entropy is often measured in terms of a compression algorithm [GPS94]. A justification for such a method is the following. Suppose that σ is generated by a stochastic process with entropy γ_0. Let G any universal compression algorithm. Then, G uses $\gamma_G \geq \gamma_0$ bits per trace request. In other words, a universal compression algorithm G provides an upper bound to the entropy of the stochastic process. Then, if $H^I, H^M > \gamma_G$, it follows that $H^I, H^M > \gamma_0$, and we have evidence that IRM and MRM do not completely characterize all the locality and lack of randomness of the trace. However, if $H^I, H^M \leq \gamma_G$, no meaningful conclusion can be reached. On the other hand, in our experiments, it will almost always be the case that $H^I, H^M > \gamma_G$. Obviously, we could select any universal compression algorithm G, but we also need γ_G to be as close as possible to γ_0 in order to obtain meaningful results. We chose the Lempel-Ziv algorithm (LZ) in `gzip`, but we remark that any universal compression algorithm would yield an upper bound to γ_0. An alternative to LZ is recency-rank coding [BSTW86, Eli87], which we will discuss in appendix D.

We now describe our experiments. Let f_i the number of occurrences of i in σ and f_{ij} the number of occurrences of the substring (i, j) in σ. For each observed trace, we constructed an IRM where page i is requested with probability $p_i = f_i/m$. We do not claim that this is the best IRM in any sense, but only that is a reasonable IRM for the observed trace. More details are found in appendix F. For each observed trace, we also constructed an MRM whose probability of transition from page i to page j is the maximum likelihood estimate $f_{ij} / \sum_l f_{il}$. We have checked that any such Markov chain is ergodic for our traces. Our algorithm to check ergodicity is as follows. We need only verify that for every pair of pages i and j there is some positive probability that a sequence of state transitions leads from i to j [Fel67, Fre83]. We check that property by constructing an auxiliary network $\mathcal{N} = (N, A)$ with a node for each page and an arc (i, j) if and

only if $P\{\sigma_t = j|\sigma_{t-1} = i\} > 0$. Then, the Markov chain is ergodic if $\mathcal{N}$ has only one strongly connected component. On the other hand, pages are numbered in the order they occur in the sequence, as described in appendix A. Therefore, page j is reachable from any page $i < j$. It follows that we need only to check that page 0 is reachable from $n-1$, which can be done with a search of $\mathcal{N}$ starting from node $n-1$. The chains corresponding to all our trace were indeed ergodic. We should compute H^M with formula (2), and so we need to estimate the steady-state probabilities p_i. Actually, we approximated the values of $p_i \simeq f_i/m$ and used (2). We computed the entropy of MRM and IRM and we report the results in table 2 under the columns H^M and H^I respectively. We also give the value of the maximum entropy $\log n$. Finally, column γ gives the number of bits per trace request in the gzip compressed file. Further details on gzip will be given in appendix A. In most traces, $\gamma \ll H^M < H^I \ll \log n$. In words, the MRM still generates instances that do not have enough regularity to match that arising in the actual traces. Thus, the information source test confirms the results of the χ^2 test. We notice that $\gamma \ll \log n$ even though consecutive references to the same page have been eliminated, as described in appendix A. The only exceptions are the fgrep traces where $H^M \simeq \gamma$. MRMs always had less entropy than IRMs.

6 The LRU Cost Test

In this section, we will compare the LRU cost on real traces with that of synthetic traces obtained from IRMs and MRMs. The LRU cost is expressed in number of page faults. The LRU cost test is by far the weakest of the tests in this paper. It assumes a specific cost model where only the number of page faults are of interest, a specific set of cache sizes, and a specific on-line algorithm. The only motivation for the LRU cost test is to measure the performance of a common on-line replacement strategy in a common cost model.

We did experiments only for trace 10 and 24. Trace 10 and 24 are at the opposite extremes according to the results of previous experiments. Trace 10 has the largest percentage of references to conditioned pages and one of the largest gaps among γ, H^M, and H^I. On the contrary, trace 24 does not have any conditioned page, and $H^M \simeq \gamma$.

We measured the cost as the number of page faults incurred by LRU. Actually, faults can be divided into *compulsory* and *capacity* faults. Compulsory faults are those that occur during the first access to a page. Capacity faults are those that occur because the cache is too small to hold all pages [HP96]. In this paper, we will count only the number c of capacity faults. We found the IRM and the MRM described in §5. Then, we generated 30 traces for each model with the inversive congruential generator (icg) described in appendix A. We then took the average cost and standard deviation among the 30 traces. We repeated the same experiments with icg replaced by `random`, as described in appendix A. We observed that the average `random` cost is within the interval (average $\pm$ standard deviation) of the icg simulation. Henceforth, we report only the results obtained with icg. We did experiments for the cache size values prescribed in [FR97]. We compare the synthetic cost with the actual cost in the following pictures and tables. Notice that the y-axis is in logarithmic scale, and reports the natural logarithm of LRU's capacity cost c. Since the y-axis is in logarithmic scale, a cost doubling corresponds to a constant increment, whose length is given by the arrow in the figure. We also observe that, in general, MRM exhibits larger values of standard deviation than IRM.

Trace 10 The results for trace 10 are reported in figure 1 and table 3. It is clear that the behavior of IRM and MRM are very different from that of the real trace. The difference is almost always much more than 100%. We also notice that MRM is slightly closer to the observed behavior than

Trace	γ	H^M	H^I	$\log n$
1	0.0504	1.3072	4.3506	6.3923
2	0.0493	1.5470	4.4246	6.4594
4	0.0511	1.3095	4.3545	6.3923
5	0.0470	1.5387	4.4021	6.1293
6	0.0514	1.3137	4.3743	6.3923
7	0.3757	1.9212	4.6452	6.2854
8	0.3548	1.9093	4.8576	6.3037
9	0.3824	1.8904	4.7634	6.2479
10	0.3769	1.9229	4.6595	6.2854
11	0.3458	1.9161	4.9115	6.3038
12	0.3641	1.8440	4.7505	6.2479
13	0.1770	0.4183	1.9622	4.9542
14	0.1807	0.4204	1.9719	4.9542
15	0.5433	0.6662	2.0571	5.0000
16	0.3270	0.6521	1.9703	5.0000
17	0.0471	0.7249	2.9488	4.7548
18	0.0358	0.1023	1.4807	4.3923
19	0.0495	0.7413	2.9661	4.7548
20	0.0384	0.1109	1.5173	4.3923
21	0.0249	0.0250	1.7016	4.4594
22	0.0191	0.0166	1.6610	4.4593
23	0.0250	0.0250	1.7017	4.4593
24	0.0192	0.0167	1.6613	4.4593
25	0.0402	1.2111	3.5129	5.0875

Table 2: Number of bits of information per request. γ is the number of bits per trace request in the gzip compressed file, H^M is the entropy of the maximum likelihood Markov chain corresponding to the trace, H_c is the entropy of the simple IRM, n is the number of distinct pages in the trace.

IRM, but MRM's cost curve is far away from the observed one. It is not clear why the added complexity of MRM should be favored to the simplicity of IRM given that both behaviors are glaringly inconsistent with that of the observed trace. Finally, we notice that the cost on IRM and MRM are not only different, but also greater than that of the observed trace. An explanation could be that $H^I, H^M \gg \gamma$ and the IRM and MRM traces are harder than the observed one.

Trace 24 The results for trace 10 are reported in figure 2 and table 4. The predicted and observed costs are much closer in this case. However, we still observe percentage cost differences ranging from 7% to 1500% for MRM. We notice that IRM is almost always worse than both the observed and MRM's cost. Again, $H^I \gg \gamma, H^M$ and the IRM trace was expected to be harder than the others. Notice that there are abrupt drops in observed costs for $k = 8$ and $k = 12$. Analogous drops have been taken to express the existence of a succession of working sets that are progressively included in fast memory as k grows [DKL+76, Den80]. The MRM curve does not give rise to such drops and interpolates smoothly before and after such drops. The conclusion is surprising because theoretically MRMs could exhibit phase transition behavior [Cur75, CV76, Den80]. We suggest that MRM is a first-order approximation of a much more complex structure. Such approximations could be helpful for some analytical purposes. On the other hand, they fail to capture the non-linear behavior of observed traces. In conclusion, it appears that MRM is a first-order approximations that do provide models with comparable entropy, but that do not have enough power to model the fundamental and highly non-linear behavior of the observed trace.

7 Conclusion

In this paper, we have performed an empirical investigation of the Markov Reference Model of program behavior. The experiments involved a new heuristic based on the χ^2 test from non-parametric statistics and an information-theoretical test to measure entropy of models and traces. Although MRMs have been extensively studied in the literature, we found that, for almost all benchmarks, MRMs do not match observed behavior. Moreover, MRMs are seldom good predictor of LRU page fault cost.

Acknowledgements

We are indebted with Brian Davison for many helpful suggestions and for proofreading this paper. We thank Ayman Khalfalah, Dan Lloyd, Naftaly Minsky, and Cenk Sahinalp for many helpful conversations.

References

[AACS87] A. Aggarwal, B. Alpern, A. K. Chandra, and M. Snir. A model for hierarchical memory. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 305–313, 1987.

[ADU71] Alfred V. Aho, Peter J. Denning, and Jeffrey D. Ullman. Principles of optimal page replacement. *Journal of the Association for Computing Machinery*, 18(1):80–93, January 1971.

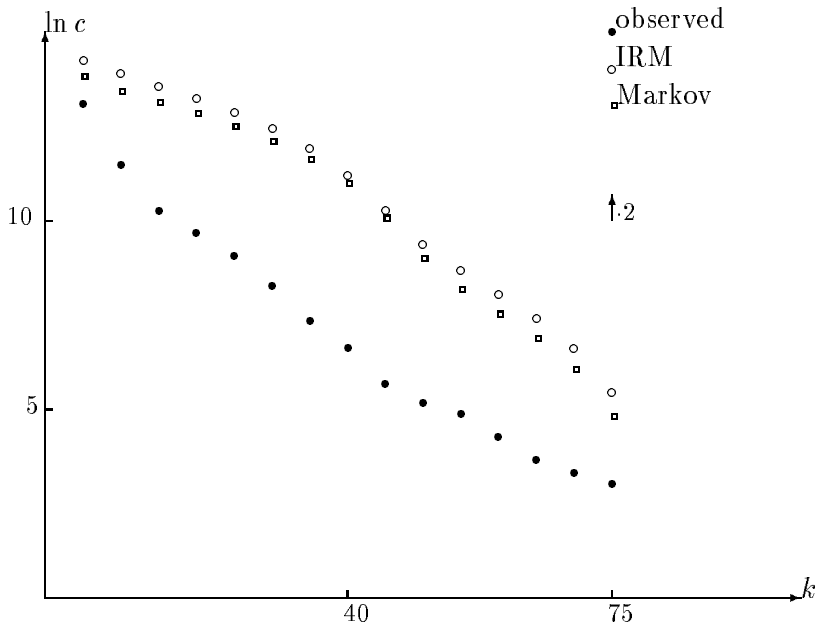

Figure 1: Cost of trace 10 and of the synthetic traces generated with IRM and LRU. The value k is the cache size. The cost c is given in a logarithmic scale. Cost doubling corresponds to the length of the arrow.

k	**observed**	**MRM**	**IRM**
5	488096	948916 ± 1414	1519360 ± 723
10	92072	638225 ± 1612	1077540 ± 917
15	28899	477386 ± 1670	772289 ± 848
20	16018	353566 ± 1465	554432 ± 777
25	8621	252296 ± 1252	387142 ± 711
30	3903	170580 ± 826	253026 ± 685
35	1547	105687 ± 621	147265 ± 507
40	769	55889 ± 419	71637 ± 324
45	297	22551 ± 298	28425 ± 228
50	180	7735 ± 155	11576 ± 111
55	133	3353 ± 106	5890 ± 72
60	72	1768 ± 67	3147 ± 58
65	39	930 ± 47	1643 ± 46
70	28	401 ± 29	756 ± 28
75	21	117 ± 10	232 ± 18

Table 3: Cost of trace 10, the Markov chain synthetic traces, and the IRM synthetic traces for values k of cache size and LRU replacement strategy. The cost for MRM and IRM are reported in the form average ± standard deviation.

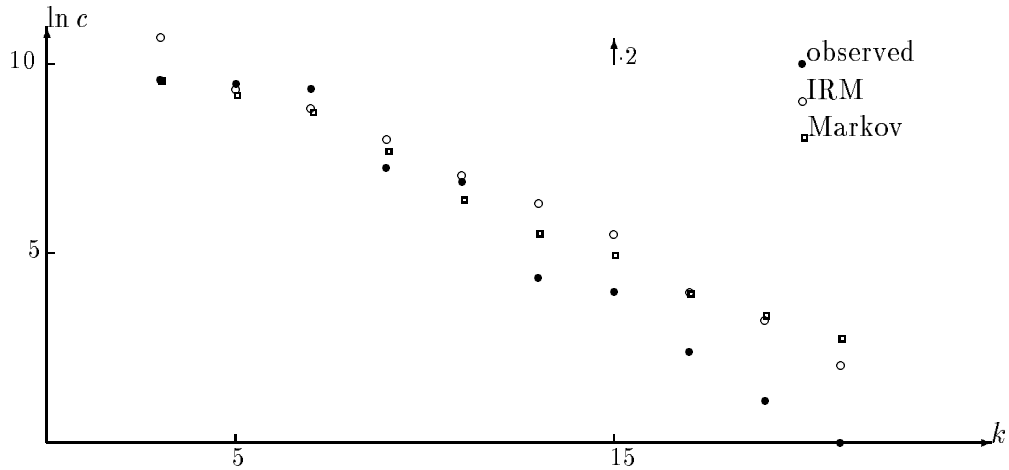

Figure 2: Cost of trace 24 and of the synthetic traces generated with IRM and LRU. The value k is the cache size. The cost c is given in a logarithmic scale. Cost doubling corresponds to the length of the arrow.

k	**observed**	**MRM**	**IRM**
3	14530	13496 ± 478	44673 ± 338
5	13013	9135 ± 373	11341 ± 118
7	11312	5887 ± 227	6858 ± 97
9	1408	2099 ± 95	3029 ± 66
11	991	584 ± 48	1159 ± 39
13	78	235 ± 43	564 ± 23
15	54	133 ± 25	244 ± 17
17	11	48 ± 14	53 ± 9
19	3	27 ± 9	25 ± 3
21	1	15 ± 11	7 ± 11

Table 4: Cost of trace 24, the Markov chain synthetic traces, and the IRM synthetic traces for values k of cache size and LRU replacement strategy. The cost for MRM and IRM are reported in the form average ± standard deviation.

[BEY97] Ran Bachrach and Ran El-Yaniv. Online list accessing algorithms and their applications: Recent empirical evidence. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 53–62, 1997.

[BL96] Thomas Ball and James R. Laurus. Efficient path profiling. In *Proceedings of MICRO-29*, 1996.

[BSTW86] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29(4):320–330, April 1986.

[CB92] Tien-Fu Chen and Jean-Loup Baer. Reducing memory latency via non-blocking and prefetching caches. In *Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 51–61, 1992.

[CD73] Edward G. Coffman, Jr. and Peter J. Denning. *Operating Systems Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[CKP91] David Callahan, Ken Kennedy, and Allan Portfield. Software prefetching. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 40–52, 1991.

[CN98] Marek Chrobak and John Noga. Competitive algorithms for multilevel caching and relaxed list update. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 87–96, 1998.

[Coc54] W. G. Cochran. Some methods for strengthening the χ^2 tests. *Biometrics*, 10:417–451, 1954.

[Cra58] Harald Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1958.

[Cur75] P. J. Curtois. Decomposability, instabilities, and saturation in multiprogramming systems. *Communications of the Association for Computing Machinery*, 18:371–377, July 1975.

[CV76] P. J. Curtois and H. Vantilborgh. A decomposable model of program paging behavior. *Acta Informatica*, 6(3):251–276, 1976.

[Den80] Peter J. Denning. Working sets: Past and present. *IEEE Transactions on Software Engineering*, SE-6(1):64–84, January 1980.

[DKL+76] P. J. Denning, K. C. Khan, J. Leroudier, D. Poitier, and R. Suri. Optimal multiprogramming. *Acta Informatica*, 7(2):197–216, 1976.

[DM83] Wilfrid J. Dixon and Frank J. Massey, Jr. *Introduction to Statistical Analysis*. McGraw-Hill, New York, 4th edition, 1983.

[Edw73] Allen Louis Edwards. *Statistical Methods*. Holt, Rinehart and Winston, New York, 3rd edition, 1973.

[EH92] Juergen Eichenauer-Herrman. Inversive congruential pseudorandom numbers: a tutorial. *International Statistical Review*, 60(2):167–176, August 1992.

[EH95] Juergen Eichenauer-Herrman. Pseudorandom number generation by nonlinear methods. *International Statistical Review*, 63(2):247–255, August 1995.

[EL86] J. Eichenauer and J. Lehn. A non-linear congruential pseudo random number generator. *Statist. Papers*, 27:315–326, 1986.

[Eli87] Peter Elias. Interval and recency ranking source coding: Two on-line adaptive variable-length schemes. *IEEE Transactions on Information Theory*, IT-33(1):3–10, January 1987.

[Fel67] William Feller. *An Introduction to Probability Theory and Its Applications*. Wiley, 1967.

[Fer76] Domenico Ferrari. Program behavior. *Computer*, 9(11):7–8, November 1976.

[FG74] M. A. Franklin and R. K. Gupta. Computation of page fault probability from program transition diagram. *Communications of the Association for Computing Machinery*, 17(4):186–191, April 1974.

[Fis70] Ronald Aylmer Fisher. *Statistical Methods for Research Workers*. Macmillan, 14th edition, 1970.

[Fis81] Joseph A. Fisher. Trace scheduling: A technique for global microcode compaction. *IEEE Transactions on Computers*, C-30(7):478–490, July 1981.

[FR97] Amos Fiat and Ziv Rosen. Experimental studies of access graph based heuristics: Beating the LRU standard? In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 63–72, 1997.

[Fra96] Michael J. Franklin. *Client data caching : a foundation for high performance object database systems*. Kluwer Academic Publishers, Boston, 1996.

[Fre83] David Freedman. *Markov chains*. Springer-Verlag, 1983.

[FW74] A. Franaszek and T. J. Wagner. Some distribution-free aspects of paging algorithm performance. *Journal of the Association for Computing Machinery*, 21(1):31–39, January 1974.

[FZ97] Michael Franklin and Stanley Zdonik. A framework for scalable dissemination-based systems. In *Proceedings of the International Conference Object Oriented Programming Languages Systems*, 1997.

[GPS94] Solomon W. Golomb, Robert E. Peile, and Robert A. Scoltz. *Basic Concepts in Information Theory and Coding*. Plenum Press, New York, 1994.

[Hal65] Anders Hald. *Statistical Theory with Engineering Applications*. Wiley, 1965.

[HP96] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Mateo, CA, 2nd edition, 1996.

[HS95] Andrew S. Huang and John P. Shen. A limit study of local memory requirements using value reuse profile. In *Proceedings of the 28th Annual International Symposium on Microarchitecture*, pages 71–81, 1995.

[HS96] Andrew S. Huang and John Paul Shen. The intrinsic bandwidth requirements of ordinary programs. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 105–114, 1996.

[IKP96] Sandy Irani, Anna Karlin, and Steven Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, June 1996.

[KL98] Sanjeev Khanna and Vincenzo Liberatore. On broadcast disk paging. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 634–643, 1998.

[KMRS88] Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):79–119, 1988.

[KPR92] Anna R. Karlin, Steven J. Phillips, and Prabhakar Raghavan. Markov paging. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 208–217, 1992.

[LW97] Hannes Leeb and Stefan Wegenkittl. Inversive and linear congruential pseudorandom number generators in empirical tests. *ACM Transaction on Modeling and Computer Simulation*, 7(2):272–286, April 1997.

[McN69] Quinn McNemar. *Psychological Statistics.* Wiley, New York, 4th edition, 1969.

[McP90] Glen McPherson. *Statistics in Scientific Investigation.* Springer-Verlag, New York, 1990.

[MDO94] Ann Marie Grizzaffi Maynard, Colette M. Donnelly, and Bret R. Olszewski. Contrasting characteristics and cache performance of technical and multi-user commercial workloads. In *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 145–156, 1994.

[MGB74] A. M. Mood, F. A. Graybill, and D. C. Boes. *Introduction to the Theory of Statistics.* McGraw-Hill, third edition, 1974.

[MR97] Tara M. Madhyastha and Daniel A. Reed. Input/output access pattern classification using hidden markov chains. In *IOPADS*, 1997.

[Pea00] K. Pearson. On a criterion that a system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen in random sampling. *Phil. Mag.*, 50:157, 1900.

[Prz90] Steven A. Przybylski. *Cache and Memory Hierarchy Design: A Performance-Directed Approach.* Morgan Kaufmann, San Mateo, 1990.

[PZ91] Mark Palmer and Stanley B. Zdonik. Fido: A cache that learns to fetch. In *Proceedings of the 17th International Conference on Very Large DataBases*, pages 225–264, 1991.

[RL92] Anne Rogers and Kai Li. Software support for speculative loads. In *Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 38–50, 1992.

[Ros96] Ziv Rosen. Access graph based heuristics for on-line paging algorithms. Master's thesis, Tel-Aviv University, December 1996.

[Sie56] Sidney Siegel. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York, 1956.

[Smi76] Alan Jay Smith. Analysis of the optimal, lookahead, demand paging algorithms. *SIAM Journal on Computing*, 5(4):743–757, December 1976.

[Smi78] Alan Jay Smith. Sequential program prefetching in memory hierarchies. *Computer*, 11(12):7–21, December 1978.

[Spi77] Jeffrey R. Spirn. *Program Behavior: Models and Measurements*. Elsevier, New York, 1977.

[ST72] G. S. Shedler and C. Tung. Locality in page reference strings. *SIAM Journal on Computing*, 1(3):218–241, September 1972.

[Tan92] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[TE93] Dean M. Tullsen and Susan J. Eggers. Limitations of cache prefetching on a bus-based multiprocessor. In *Proceedings of the 1993 International Symposium on Computer Architecture*, pages 278–288, 1993.

[VK96] Jeffrey Scott Vitter and P. Krishnan. Optimal prefetching via data compression. *Journal of the ACM*, 43(5):771–793, September 1996.

[WL53] Helen Mary Walker and Joseph Lev. *Statistical Inference*. Holts, New York, 1953.

[You94] Neal Young. The K-server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

Program	Traces
gnuplot	1–6
latex	7–12
gunzip	13–14
gzip	15–16
paging	17–20
fgrep	21–24
find	25

Table 5: Programs generating the Fiat-Rosen traces [FR97]. The Fiat-Rosen traces are numbered from 1 to 25 and are generated by the programs in the left column. The program input and further information on those traces are found in [FR97, Ros96].

A Experimental Set-up

In this section, we describe the set-up of our experiments. We used the Fiat-Rosen traces [FR97], which are available via anonymous ftp from `ftp.math.tau.ac.il` in the directory `/pub/rosen/results`. The Fiat-Rosen traces are sequences of instruction page requests, obtained from sample executions of LaTeX, gnuplot, grep, find, gunzip, gzip, and the paging utility on a Sparc-10 with Sun4 CPU architecture and running SunOS. Table 5 reports the traces in terms of the programs that generated them. The Fiat-Rosen traces do not contain requests for data pages. Moreover, in the Fiat-Rosen traces, consecutive requests to the same page i have been replaced by one access to i [Ros96]. Consecutive traces have been eliminated because any demand paging algorithm would not fault on any of the consecutive requests except possibly on the first one. Further information on those traces can be found in [FR97]. Notice that trace 3 is currently unavailable: contact the authors of [FR97] for more information. We normalized the traces to remap page addresses to the range $0, 1, \ldots, n-1$. Thus, we obtain a sequence of *nominal* pages, whose value have no significance for the original program. The renaming has been done only as a matter of convenience and to simplify notation and coding. Indeed, program behavior does not depend on the actual page address [Spi77]. Pages are renumbered in order of their first appearance in the trace, and we will use specific properties of such numbering later on.

In our experiments, we used a Sun UltraSparc1. In our simulations, we used two random number generators. The first one is `random (3B)`, the non-linear additive feedback random number generator in the SunOS/BSD compatibility library. We used a state of 256 bytes initialized to `state[i] = i` and seed 0. The second one is an inversive congruential random number generator [EL86]. Inversive congruential generators have good theoretical [EH92, EH95] and experimental properties [LW97]. We used an implementation of the inversive congruential random generator which is part of the pLab package, version 2.2, written at the University of Salzburg (`http://random.mat.sbg.ac.at/`). We used a period $p = 2147483647$, parameters $a = 14288$, $b = 758634$, and seed 0.

We used the `gzip -9` program, version 1.2.4 (18 Aug 93), which has been compiled with the options `DIRENT UTIME STDC_HEADERS HAVE_UNISTD_H`. The program `gzip` introduces a few extra unneeded bytes, specifically a file header and 5 bytes every 32K. We made no attempt to locate and delete any extra byte. Notice that the compressed traces are originated from the sequences of nominal page requests described above, where each page reference is represented by a printable

character.

B The χ^2 Test for MRMs

The main line of the χ^2 test for paging traces is as follows. Henceforth assume that i is not a sporadic page. If the Markov property holds, we should have that the transition from i to j is independent of the particular h from which we have reached i, and so it should be that $a_{hij} \simeq \alpha_{hi}\beta_{ij}s_i$. The independence assumption is checked by verifying that A_i and B_i are close under some measure. In particular, the χ^2 test uses Pearson's Chi-squared statistic

$$X^2 = \sum_{h=1}^{H}\sum_{j=1}^{J} \frac{(a_{hij} - \alpha_{hi}\beta_{ij}s_i)^2}{\alpha_{hi}\beta_{ij}s_i} .$$

The value X^2 is not the final outcome of the χ^2 test for two reasons. First, we would like to have the result of the test expressed by a single number $p \in [0,1]$, where a value of 1 denotes perfect independence and a value of 0 denotes perfect association. The parameter p is a measure of confidence in the lack of association between h and j. The parameter p is sometimes thought of as the probability that A_i is observed if the underlying probabilistic model is given by B_i. Even though this is not a mathematically correct explanation of p, it is sometimes a good insight on the meaning of p. Second, we would desire a parameter p that is normalized in the size of A_i. In other word, for a fixed value of p, the value X^2 should get larger as the size of A_i increases. Define the *degrees of freedom* of page i to be $\nu_i = (J-1)(H-1)$. The χ^2 test is completed by finding the probability

$$p = P\{\chi^2(\nu_i) \geq X^2\} ,$$

where $\chi^2(\nu_i)$ is a Chi-squared random variable with $\nu_i > 0$ degrees of freedom. Notice that $p \in [0,1]$ and that if p is fixed and the degrees of freedom increase, then also X^2 increases, which is exactly what we desired. We recall that a value of p less than .01 is considered strong evidence of association [McP90], and we will say that a page is conditioned if $p \leq .01$. The parameter p is defined when $\nu_i > 0$. Suppose now that $\nu_i = 0$. If a page has no degree of freedom, then either $J = 1$ or $H = 1$. If $J = 1$, then there is only one possible transition from i, and so (1) trivially holds. If $H = 1$, then there is only one possible transition into i, and again (1) trivially holds. Therefore, no information is needed from the χ^2 test if $\nu_i = 0$. Pages with no degree of freedom are either transition or undetermined pages (we do not consider sporadic pages in this section). We can interpret the χ^2 test as follows. We compare the matrix A_i of observed frequencies with the matrix B_i of maximum likelihood frequencies under the independence assumption. In other words, for each page i, we compare observed transition frequencies with maximum likelihood transition probabilities. Notice that different traces originating from the same program could have different maximum likelihood transition probabilities. If A_i is far from B_i, then we can assume that there is an association between h and j.

Some modifications are needed in order for the χ^2 test to provide meaningful answers. First of all, the χ^2 test is not applicable when $H = J = 2$ [McP90]. In this case, we resort to Fisher exact test. The exact test works as follows [Sie56]. Rewrite

$$A_i = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

In this case, p is the exact probability that A_i is observed if no association exists, and is equal to

$$p = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{s_i!a!b!c!d!} .$$

C Double Predecessor Prediction

In this section, we will compare prediction strategy that examine the last and the last two page requests. Define f_{ij} to be the number of times the substring (i, j) occurs in σ. If the prediction is based only on $i = \sigma_{t-1}$ (for $t > 1$), the maximum likelihood j that will be requested at step t is $j = \arg\max_j\{f_{ij}\}$. We will call such guesses j the *single predecessor prediction* from i. If the prediction is based on $h = \sigma_{t-2}$ and i, then the maximum likelihood j is the page that maximizes a_{hij}. We will call such guess j the *double predecessor prediction* from h and i. Notice that both single and double predecessor prediction are possible only if the traces have been profiled and off-line information is available. Although profiling is not always possible, prediction strategies based on off-line profiles have been used in other prediction strategies [PZ91]. Table 6 reports the percentage of trace references that are correctly predicted by the single predecessor and double predecessor strategies. In most cases, two reference prediction significantly improves the success rate. The exception is the fgrep traces, which was expected because few pages are conditioned in that benchmark. Another exception is trace 18. The reason is that in trace 18 about 82% of the trace is a loop between page 0 and page 19. Under the circumstances, wrong predictions might occur only on the remaining 18% of the trace. Notice also that for half of our traces there is a significant percentage of pages that are mispredicted even by the double strategy. Recall that a prefetching strategy corresponds to a model of program behavior. The double strategy corresponds to a second order Markov chain model. Since the double strategy still mispredicts a significant number of references, we doubt that a second order Markov chain model is an adequate model of program behavior. Moreover, it is not intuitively clear to us why program behavior could be modeled by a Markov chain of any fixed order.

D Recency-rank Coding

In this section, we will examine the recency-rank coding algorithm and its relation with the distance string, a well-know paging quantity. In *recency-rank coding*, the algorithm is given a *dictionary* of all the *words* that appear in the file to compress. The compression algorithm will maintain the dictionary as a linked list that is reorganized under the Move-To-Front [BSTW86, Eli87] strategy, or any other on-line algorithm [BEY97]. The output of the algorithm is the sequence of word position in the list. We compare recency-rank coding with the following notion of LRU behavior. The LRU algorithm maintains all n pages in a stack, and whenever a page is requested it moves it to the top of the stack. LRU always keep in fast memory the topmost k pages. The *distance string* is the sequence of page distances from the top of the LRU stack [Spi77]. We notice that when the dictionary is $\{0, 1, \ldots, n-1\}$, the recency-rank compression algorithm returns exactly the distance string of the trace. Unfortunately, this is not a very good choice of a compression algorithm, or, at least, not a very good choice for a dictionary. Indeed, recency-rank coding would return at least one bit per request, and we observed that LZ uses much less than that.

Trace	Single	Double
1	59.6%	81.1%
2	54.9%	80.4%
4	59.5%	81.1%
5	54.9%	80.4%
6	59.5%	81.1%
7	53.1%	71.3%
8	54%	73.7%
9	53.1%	72.8%
10	52.7%	71.5%
11	53.3%	73.6%
12	53.8%	73.7%
13	80.9%	96.9%
14	81%	96.6%
15	76.1%	82.8%
16	75.8%	83.7%
17	75.1%	86.2%
18	97.1%	98.4%
19	74.5%	85.8%
20	96.8%	98.3%
21	99.4%	99.7%
22	99.6%	99.8%
23	99.4%	99.7%
24	99.6%	99.8%
25	59.8%	76%

Table 6: Success rate of prediction based on the last and the last two references. Single prediction determines the next page according to the last request. Double prediction determines the next page according to the last two requests. The percentage is the fraction of correctly predicted references.

page	s_i	p	D	notes
0	76398	$9 \cdot 10^{-376}$		
14	70024	0		
1	66410		66382	$\nu = 0, (H = J = 1)$
15	50012	0		
13	40053	0		
10	20020			$\nu = 0, (H = 1, J = 2)$
2	10045		10000	$\nu = 0, (H = J = 1)$
8	10045		10001	$\nu = 0, (H = J = 1)$
18	10010		10000	$\nu = 0, (H = J = 1)$
12	10003		9999	$\nu = 0, (H = J = 1)$
4	198	10^{-43}		sporadic
5	198	5^{-17}		sporadic
3	132	$4 \cdot 10^{-54}$		sporadic
6	99			$\nu = 0, (H = 1, J = 2)$, sporadic
9	38	10^{-7}		sporadic
7	33			$\nu = 0, (H = J = 1)$, sporadic
16	14	0.1243		sporadic
11	13			small entries, sporadic
23	10			$\nu = 0, (H = 2, J = 1)$, sporadic
20	8			small entries, sporadic
17	6			small entries, sporadic
19	6			$\nu = 0, (H = J = 1)$, sporadic
21	6			small entries, sporadic
25	6			$\nu = 0, (H = J = 1)$, sporadic
26	4			small entries, sporadic
22	3			small entries, sporadic
24	3			small entries, sporadic

Table 7: Results of the χ^2 test for trace 17. The column s_i reports the number of requests for page i, p is the result of the χ^2 test, D is the maximum entry in the matrix A_i.

E The χ^2 Test for Trace 17

We now report some detailed results on trace 17 for the purpose of giving an example and clarify the experimental results. There is no particular reason why we chose trace 17 as opposed to any other trace, and the purpose of table 7 is only that of giving an example. Our results on trace 17 are as follows. We have $n = 27$ pages on the whole, and their characteristics are reported in table 7. The column D shows the size of the largest entry in the matrix A_i for transition pages. Notice that the 10 most frequent pages concentrate 99.79% of all the trace references. Among the other pages, we notice that page 6 removes randomness from the process because $H = 2$ and $J = 1$, page 23 introduces some randomness for the converse argument, and page 16 has incoming and outgoing edges that are not associated.

F An IRM model

In this section, we make some observations on the IRM underlying the observed traces. The main difficulty in finding an IRM for the Fiat-Rosen traces is that consecutive references to the same page have been deleted. We give the following example to illustrate the difficulty. Suppose that $n = 3$ pages are generated by an IRM that issues page i with probability π_i and that subsequently we delete consecutive references to the same page. Then, the probability p_i that i appears in the resulting trace is

$$p_i = \sum_{j \neq i} \frac{\pi_i \pi_j}{1 - \pi_j} = \pi_i \sum_{j \neq i} \frac{\pi_j}{1 - \pi_j} \ .$$

Suppose that $\pi_i = 1/3$ $(i = 1, 2, 3)$. Then, $p_i = 1/3$ $(i = 1, 2, 3)$. However, also for $\pi_1 = 1/2$, $\pi_2 = \pi_3 = 1/4$, we obtain $p_i = 1/3$ $(i = 1, 2, 3)$. We conclude that in general the knowledge of the p_i's does not allow us to reconstruct the underlying π_i's. However, if all π_j's are very small, then $1 - \pi_j \simeq 1$ and $p_i \simeq \pi_i$. Intuitively, the result is correct because if $\pi_j \simeq 0$, then consecutive repetitions of the same page are unlikely, and the deletion step leaves the request sequence virtually unchanged.