Can Entropy Characterize Performance of Online Algorithms?

Gopal Pandurangan* Eli Upfal*

Abstract

We focus in this work on an aspect of online computation that is not addressed by the standard competitive analysis. Namely, identifying request sequences for which non-trivial online algorithms are useful versus request sequences for which all algorithms perform equally bad. The motivation for this work are advanced system and architecture designs which allow the operating system to dynamically allocate resources to online protocols such as prefetching and caching. To utilize these features the operating system needs to identify data streams that can benefit from more resources.

Our approach in this work is based on the relation between entropy, compression and gambling, extensively studied in information theory. It has been shown that in some settings entropy can either fully or at least partially characterize the expected outcome of an iterative gambling game. Viewing online problem with stochastic input as an iterative gambling game, our goal is to study the extent to which the entropy of the input characterizes the expected performance of online algorithms for problems that arise in computer applications. We study bounds based on entropy for three online problems – list accessing, prefetching and caching. We show that entropy is a good performance characterizer for prefetching, but not so good characterizer for online caching. Our work raises several open questions in using entropy as a predictor in online computation.

*Computer Science Department, Brown University, Box 1910, Providence, RI 02912-1910, USA. E-mail: {gopal, eli}@cs.brown.edu. Supported in part by NSF grant CCR-9731477. A preliminary version of this paper appeared in the proceedings of the 12th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Washington D.C., 2001.

1 Introduction

Advanced system and architecture design allows dynamic allocations of resources to online tasks such as prefetching and caching. To fully utilize this feature the system needs an efficient mechanism for estimating the expected gain from using these resources. Prefetching, for example, is an expensive operation since it "burns instruction bandwidth"[17]. However, successful prefetching can significantly speedup computation. Thus, one needs to compare the gain from prefetching on a given data stream to the cost in instruction bandwidth. The tradeoff between resource allocation and gain is even more transparent in the case of malleable cache [21, 26, 8]. In this architecture the cache can be dynamically partitioned between different data streams. A data stream that can make better use of a larger cache is assigned more space, while a stream with very little structure or repeats is allocated a smaller cache space. Again, efficient utilization of this technology requires a mechanism for predicting caching gain for a given data stream.

Online algorithms have been studied in the theory community mainly in the context of *competitive analysis* (see [5] for a comprehensive survey). Competitive analysis compares the performance of different algorithms, but it gives no information about the actual gain from using them. In particular, even the best algorithm under the competitive analysis measure might fail on almost all requests of some sequence. Thus, an entirely new approach is needed in order to quantify the amount of resources the system should allocate to a given online process. In this work we explore the relation between the entropy of the stream of requests and the gain expected from online algorithm performing on this request sequence. Entropy measures the randomness or uncertainty of a random process. We expect online algorithms to perform well on highly predictive request sequences, generated by a source with low entropy, and to perform poorly on sequences with little pattern, generated by a high entropy source. Our work is motivated by the extensive work in information theory relating data compression, entropy and gambling. It has been shown that for some special cases of gambling games the entropy of the stochastic process fully characterizes the maximum expected profit for any strategy for that game (see section 1.1). Our goal is to explore similar relations between entropy and online problems in computer applications. We discuss here three online problems: list accessing, prefetching and caching. We show that in the case of prefetching entropy gives a good characterization of the best online performance that is possible, while in the case of caching entropy does not fully characterize the best online performance.

1.1 Related Work

The three online problems considered here were extensively studied in the competitive analysis model. It has been shown in [25] that the competitive ratio [1] of the *move to front (MTF) algorithm* for the list accessing problem is two [25]. In the case where the input sequence is drawn from a discrete memoryless source the MTF algorithm has been compared to the performance of a static offline algorithm SOPT that initially arranges the list in decreasing order of request probabilities and never reorders them thereafter. It was shown in [15] that $\mathrm{MTF}(D) \leq \frac{\pi}{2}\mathrm{SOPT}(D)$, where D is the distribution of the source. Albers et.al [1] analyze the performance of the TIMESTAMP algorithm on a discrete memoryless source with

[1] An online algorithm ALG has a competitive ratio of c if there is a constant α such that for all finite input sequences I, $\mathrm{ALG}(I) \leq c \times \mathrm{OPT}(I) + \alpha$, where OPT is the optimal offline algorithm.

distribution D and proved that for any distribution D, TIMESTAMP$(D) \leq 1.34 \times$ SOPT(D) and with high probability, TIMESTAMP$(D) \leq 1.5 \times$ OPT(D). The actual work done by the MTF algorithm was studied when the request sequence is generated by a discrete memoryless source with probability distribution D[1, 4, 15].

For online caching (or demand paging) the well known LRU (Least Recently Used) has a competitive ratio of k [25], where k is the cache size, while the randomized MARKER algorithm is $2 \log k$ competitive [10]. Franaszek and Wagner [13] studied a model in which every request is drawn from a discrete memoryless source. Karlin et.al [18] study Markov paging where the sequence of page requests is generated by a Markov chain. Their main result is an efficient algorithm which for any Markov chain will achieve a fault-rate at most a constant times optimal.

For the problem of prefetching, competitive analysis is meaningless as the optimal offline algorithm will always prefetch the correct item and hence incurs no cost. Vitter and Krishnan [27] consider a model where the sequence of page requests is assumed to be generated by a Markov source (defined in section 5), a model which is closest in spirit to (but less general than) our model of a stationary ergodic process. They show that the fault rate of a Ziv-Lempel [28] based prefetching algorithm approaches the fault rate of the best prefetcher (which has full knowledge of the Markov source) for the given Markov source as the page request sequence length $n \to \infty$.

Kelly [19] was the first to study the relation between data compression, entropy and gambling, showing that the outcome of a horse race gambling with fair odds is fully characterized by the entropy of the stochastic process. It was shown [19, 2] that the growth rate of investment in the horse race is equal to $\log m - H$, where m is the number of horses and H is the entropy of the source. Similar results have been shown for portfolio selection strategies in equity market investments [3, 6]. Our results on list accessing are based on the work of Bentley et.al [4] who showed that any list update algorithm can be used to develop a data compression scheme. They also showed that for a discrete memoryless source the expected number of bits needed to encode an alphabet using MTF is linear in the entropy of the source. Similar results have been shown by Albers et.al [1] for the TIMESTAMP algorithm. Our results on prefetching are motivated by the work of Feder and Merhav [11] relating entropy of a discrete random variable to the minimal attainable probability of error in guessing its value. In the context of prefetching their results can be viewed as giving a tight bound on the fault rate when the size of cache is 1. A tight lower bound on this error probability is given by Fano's inequality [6, theorem 2.11.1]. Their main result is a tight upper bound for the fault rate when $k = 1$. Feder and Merhav also showed that the same lower and upper bounds (for $k = 1$) hold for a stationary ergodic source. However, their upper bound does not seem to generalize to higher values of k. Note that there is more work in the information theory on predicting binary sequences (corresponding to prefetching in a universe of two pages with cache of size 1) [12], however these results cannot be generalized to our prefetching scenario. Our approach to deriving the upper bound on the fault rate for an arbitrary ergodic source and arbitrary cache size k is different and is based on the well-known Liv Zempel universal algorithm for data compression [28]. Our proof uses Rissanen's interpretation of the Liv Zempel Algorithm [24]. See Algoet [2] for further results on universal schemes for prediction, gambling and portfolio selection.

1.2 New Results

We focus on three online problems in this paper: list accessing, prefetching, and caching. Our goal is to study the relation between the entropy of the sequence of requests and the best performance of an online algorithm for these problems.

We assume that the sequence of requests is generated by a *discrete stationary ergodic process* [14, definition 3.5.13] which is the most general stochastic source considered in information theory. It includes powerful models such as memoryless and (stationary) Markov sources [14, 27, 7].

For the list accessing problem we show that any deterministic online algorithm requires an average work of $\Omega(2^H)$ steps per item access, where H is the entropy of the input source.

For the prefetching problem we give an upper and lower bound showing that the average number of faults of the best algorithm is linear in H, the entropy of the input source. Our lower bound on the fault rate can be seen as a generalization of Fano's inequality for $k > 1$. Our upper bound generalizes the well known upper bound of $\frac{1}{2}H$ on the minimal error probability for guessing the value of a discrete random variable (i.e. $k = 1$) shown by completely different techniques [14, pages 520-521], [11, 16].

In contrast, we consider the caching problem for two stationary ergodic sources with equal entropy. We show that the best caching fault rate for the two sources fall in two disjoint intervals as a function of H, the entropy of the source. Thus, in the case of caching, entropy alone is not a sufficient predictor for online performance.

2 Preliminaries

We model a request sequence for an online process as an indexed sequence of (discrete) random variables (also called as a *stochastic process*), denoted by $\{X_i\}$. The range of $\{X_i\}$, for all i is the finite alphabet set $\mathcal{H}$. Let $l = |\mathcal{H}|$. Throughout this paper we use the notation lg to denote logarithm to the base 2. To define the entropy rate of $\{X_i\}$ we recall some basic information theory terms. The entropy $H(X)$ of a discrete random variable X with alphabet $\mathcal{H}$ and probability mass function $p(x) = \Pr\{X = x\}, x \in \mathcal{H}$ is

$$H(X) = -\sum_{x \in \mathcal{H}} p(x) \lg p(x) \tag{1}$$

The *joint entropy* $H(X_1, X_2)$ of a pair of discrete random variables (X_1, X_2) with a joint distribution $p(x_1, x_2)$ is

$$H(X, Y) = -\sum_{x_1 \in \mathcal{H}} \sum_{x_2 \in \mathcal{H}} p(x_1, x_2) \lg p(x_1, x_2) \tag{2}$$

The *conditional entropy* $H(X_2|X_1)$ is

$$\begin{aligned} H(X_2|X_1) = \sum_{x_1 \in \mathcal{H}} p(x_1) H(X_2|X_1 = x) = \\ -\sum_{x_1 \in \mathcal{H}} \sum_{x_2 \in \mathcal{H}} p(x_1, x_2) \lg p(x_2|x_1) \end{aligned} \tag{3}$$

The entropy per letter $H_n(\mathcal{H})$ of a stochastic process $\{X_i\}$ in a sequence of n letters is defined as

$$H_n(\mathcal{H}) = \frac{1}{n} H(X_1, X_2, \ldots, X_n) \tag{4}$$

Definition 2.1 *The entropy rate of a stochastic process $\{X_i\}$ is defined by*

$$H(\mathcal{H}) = \lim_{n\to\infty} H_n(\mathcal{H})$$

when the limit exists.

Definition 2.2 *A stochastic process is* stationary *if the joint distribution of any subset of the sequence of random variables is invariant with respect to shifts in the time index, i.e.,*

$$\Pr\{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\} = \\ \Pr\{X_{1+t} = x_1, X_{2+t} = x_2, \ldots, X_{n+t} = x_n\}$$

for every shift t and for all $x_1, x_2, \ldots, x_n \in \mathcal{H}$.

It can be shown that [14, Theorem 3.5.1] for *stationary processes* (with finite $H_1(\mathcal{H})$)) the limit $H(\mathcal{H})$ exists and

$$\lim_{n\to\infty} H_n(\mathcal{H}) = \\ \lim_{n\to\infty} H(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) = H(\mathcal{H}) \tag{5}$$

$$H(X_n|X_{n-1}, \ldots, X_1) \text{ is} \\ \text{non-increasing with } n \tag{6}$$

$$H_n(\mathcal{H}) \geq H(X_n|X_{n-1}, \ldots, X_1) \tag{7}$$

$$H_n(\mathcal{H}) \text{ is non-increasing with } n \tag{8}$$

In particular when $X_1, X_2, \ldots$ are independent and identically distributed random variables (also called as a *discrete memoryless source*)

$$H(\mathcal{H}) = \lim \frac{1}{n} H(X_1, X_2, \ldots, X_n) = \lim \frac{nH(X_1)}{n} = H(X_1).$$

Henceforth in the paper when we say *entropy of a request sequence* we mean the entropy rate of the stochastic process (or the source) generating the sequence, denoted simply by H (or H_n for a sequence of length n).

3 List Accessing

We start with a simple example relating the cost of online list accessing to the entropy of the request sequence. As in Borodin & El-Yaniv [5] we consider the *static list accessing model* in which a fixed set of l items, is stored in linked list. The algorithm has to access sequentially a sequence of n requests for items in the list. The access cost $a(X_i)$ of an item is the number of links traversed by the algorithm to locate the item, starting at the head of the list. Before each access operation the algorithm can rearrange the order of items in the list by means of transposing an element with an adjacent one. The cost is 1 for a single exchange. Let $c(X_i)$ be the total cost associated with servicing element X_i. $c(X_i)$ includes both the access cost $a(X_i)$ and any transposition cost incurred before servicing X_i.

Following Bentley et al. [4] we explore the relation between list accessing and data compression by using the linked list as a data structure of a data compression algorithm. Assume that a sender and a receiver start with the same linked list, and use the same rules for rearranging the list throughout the execution. Instead of sending item X, the sender needs only to send the distance i of X from the head of the linked list, i.e. the work involved in retrieving item X. We encode the integer distance by using a variable length prefix code. The lower bound depends on the particular encoding used for the distance. Consider an encoding scheme that encodes an integer i using $f(i)$ bits. To get a lower bound on the work done, we need f to be a concave nondecreasing function (when defined on the non-negative real).

Theorem 3.1 *Let $\bar{c}$ be the average cost of accessing an item by any deterministic algorithm $\mathcal{A}$ on a stationary ergodic sequence of requests $< X >= X_1, X_2, \ldots, X_n$. Then $\bar{c} \geq f^{-1}(H)$, where H is the entropy of the sequence, and f is a concave nondecreasing invertible function such that there is an encoding scheme for the integers that encodes integer i with up to $f(i)$ bits.*

Proof: $\bar{c} \geq \frac{1}{n}\sum_{i=1}^{n} c(X_i)$, and $c(X_i) \geq a(X_i)$, where $a(X_i)$ is the distance of X_i from the head of the linked list at time i, which is the value sent by the sender at time i. If the sender encodes $a(X_i)$ using $f(a(X_i))$ bits, then by variable-length source coding theorem [14, theorem 3.5.2] and by equations 5 to 8,

$$\frac{1}{n}\sum_{i=1}^{n} f(a(X_i)) \geq H_n \geq H \tag{9}$$

Since f is concave, by Jensen's inequality and using 9,

$$f(\bar{c}) \geq f(\frac{1}{n}\sum_{i=1}^{n} a(X_i)) \geq \frac{1}{n}\sum_{i=1}^{n} f(a(X_i)) \geq H$$

Hence, $\bar{c} \geq f^{-1}(H)$. □

We can now get concrete lower bounds by plugging in appropriate coding functions. A simple prefix coding scheme encodes an integer i using $1 + 2\lfloor \lg i \rfloor$ bits [2] [9]. The encoding of i consists of $\lfloor \lg i \rfloor$ 0's followed by the binary representation of i which takes $1 + \lfloor \lg i \rfloor$ bits, the first of which is a 1. This encoding function gives the following corollary to theorem 3.1.

Corollary 3.1 *Any deterministic online algorithm for list accessing has to incur an average cost of $2^{(H-1)/2}$ per item, where H is the entropy rate of the sequence.*

We get a better lower bound by replacing the $\lfloor \lg i \rfloor$ 0's followed by a 1 in the above scheme by $\lg\lfloor 1 + \lg l \rfloor$ bits giving an encoding for i with $\lfloor \lg i \rfloor + \lg\lfloor 1 + \lg l \rfloor$ bits. Using this scheme we prove:

[2]This function can be made invertible in the obvious way to obtain the lower bound specified in corollary 3.1. Similar comment holds for the encoding function used to derive corollary 3.2.

Corollary 3.2 *The average cost of accessing an item for a deterministic online algorithm is at least* $\frac{2^H}{\lfloor \lg l+1 \rfloor}$*, where* l *is the size of the alphabet.*

We note that theorem 3.1 actually applies for any list accessing algorithm even if it is a randomized algorithm. That is, for a randomized list accessing algorithm the expected cost of accessing an item is lower bounded as specified by theorem 3.1. This follows from Yao's Minimax Principle [22]. Thus the lower bounds derived in the corollaries hold for randomized algorithms also.

We conclude our discussion on list accessing by showing a lower bound on the performance of the well-studied static offline algorithm (SOPT) on a discrete memoryless source with probability distribution $D = \{p_1, \ldots, p_l\}$ where p_i is the probability of accessing item x_i. SOPT initially arranges the list in decreasing order of request probabilities and never reorders them thereafter. Although SOPT is not an online algorithm, because of its simplicity has been used as a benchmark for comparison to other (online) algorithms [5]. The expected access cost per item is given by $\mathrm{SOPT}(D) = \sum_{i=1}^{l} ip_i$. The following theorem gives a lower bound on the expected cost of accessing an item by SOPT based on the entropy of the input distribution D.

Theorem 3.2 *The expected cost of accessing an item by SOPT on a discrete memoryless source with distribution* D *is at least* 2^{H-1} *where* H *is the entropy of* D*.*

Proof: Suppose let w (an integer for now) be the average cost of accessing an item by SOPT. Note that $w \leq (l+1)/2$. Then the probability distribution which maximizes the entropy is given by $(\underbrace{\frac{1}{2w-1}, \ldots, \frac{1}{2w-1}}_{2w-1\ terms}, \underbrace{0, \ldots, 0}_{(l-(2w-1))\ terms})$ and the corresponding entropy is given by $\lg(2w-1)$. This can be shown by a straightforward verification of the Kuhn-Tucker conditions for optimality of the following non-linear program:

$$\text{Maximize} \quad -\sum_{i=1}^{l} p_i \lg p_i$$

subject to the following constraints:

$$\begin{gathered} \sum_{i=1}^{l} ip_i = w \\ \sum_{i=1}^{l} p_i = 1 \\ p_i \geq 0,\ i = 1, \ldots, l \\ p_i - p_{i+1} \geq 0,\ i = 1, \ldots, l-1 \end{gathered}$$

Thus, if H is the entropy of any distribution D then, $H \leq \lg(\lceil 2w-1 \rceil)$, which yields the result. □

4 Prefetching

As in [27] we consider the following formalization of the prefetching problem: we have a collection $\mathcal{H}$ of pages in memory and a cache of size k, and typically $k \ll |\mathcal{H}|$. The system can prefetch k items to the cache prior to each page request. The fault rate is the average number of steps in which the requested item was not in the cache.

Let $l = |\mathcal{H}|$. Given a request sequence $< X >= X_1, X_2, \ldots X_n$, we are interested in the *minimal expected page fault rate* of a request sequence i.e., the minimum long term frequency of page faults that is possible for the sequence. We show the existence of this quantity when the request sequence is generated by a stationary ergodic process.

4.1 Lower Bound

We first prove the lower bound for a discrete memoryless source, generalizing the result in Feder and Merhav [11].

We observe that the optimal prefetching strategy in a discrete memoryless source is obvious (a consequence of Bayes' decision rule, for example see [16]):

Lemma 4.1 *Let $p(.)$ be a probability distribution on $\mathcal{H}$. Suppose each page in the sequence is drawn i.i.d with probability distribution $p(.)$. Then the minimal expected page fault rate can be obtained by picking the pages (in the cache) with the top k probabilities. Hence the minimal expected fault rate is given by $1 - \sum_{x \in T(p(.))} p(x)$, where $T(p(.))$ is the set of pages with the top k probabilities in the distribution $p(.)$.*

Our goal is to relate the fault rate of the above strategy to the entropy of the source. Consider a discrete random variable X, and let $p(i) = Pr\{X = i\}$ for $i \in \mathcal{H}$. Assume without loss of generality that $p(1) \geq p(2) \geq \cdots \geq p(l)$. Let $P = [p(1), \ldots, p(l)]$ be the probability vector and let $P_\pi = \{P \mid p(i) \geq 0,\ \forall i,\ \sum_{i=1}^{l} p(i) = 1 \text{ and } \sum_{i=1}^{k} p(i) = 1 - \pi\}$ Let $H(P)$ (or $H(X)$) be the entropy of the random variable having the distribution given by P. Given the minimal expected fault rate $\pi(X)$ (or π for simplicity) we would like to find a upper bound on the entropy as $H(X) \leq \max_{P \in P_\pi} H(P)$.

Lemma 4.2 *Let the minimal expected page fault rate be π. Then the maximum entropy $H(P_{max}(\pi))$ is given by $(1-\pi)\lg(\frac{k}{1-\pi}) + \pi \lg(\frac{l-k}{\pi})$.*

Proof: Given the minimal expected page fault rate π, the maximum entropy distribution $P_{max}(\pi)$ is given by $(\underbrace{\frac{1-\pi}{k}, \ldots, \frac{1-\pi}{k}}_{k\ terms}, \underbrace{\frac{\pi}{l-k}, \ldots, \frac{\pi}{l-k}}_{(l-k)\ terms})$
assuming $\pi \leq 1 - k/l$ (which is always true). This distribution maximizes the entropy because of the following argument. Let $p(x)$ be any probability distribution on $\mathcal{H}$. Then the relative entropy (or Kullback Leibler distance) between $p(x)$ and $P_{max}(\pi)$ is given by [6, definition 2.26]

$$\sum_{x \in \mathcal{H}} p(x) \lg(p(x)/P_{max}(\pi)) =$$
$$-H(X) + \sum_{x \in \mathcal{H}} p(x) \lg 1/P_{max}(\pi))$$

Since the relative entropy is always positive[6, Theorem 2.6.3] we have

$$H(X) \leq \lg(\frac{k}{1-\pi}) \sum_{x=1}^{k} p(x) + \lg(\frac{l-k}{\pi}) \sum_{x=k+1}^{l} p(x) =$$
$$(1-\pi)\lg(\frac{k}{1-\pi}) + \pi \lg(\frac{l-k}{\pi})$$

□

Corollary 4.1 $\pi \geq \frac{H - 1 - \lg k}{\lg(\frac{l}{k} - 1)}$

Proof: From lemma 4.2,

$$\begin{aligned} & H \leq \\ -(1-\pi)\lg(1-\pi) - \pi\lg\pi + (1-\pi)\lg k + \pi\lg(l-k) & \\ & = h(\pi) + (1-\pi)\lg k + \pi\lg(l-k) \end{aligned}$$

where, $h(\pi) = -\pi\lg\pi - (1-\pi)\lg(1-\pi)$ is the binary entropy function which takes values between 0 and 1. Hence, $H \leq 1 + \lg(k^{1-\pi}(l-k)^{\pi})$ which gives the result. □

We now show that the same lower bound holds for any stationary ergodic process generalizing the argument of [11, Theorem 1]. First we need to define the following. Let (X,Y) be a pair of discrete random variables (each with range $\mathcal{H}$) with joint distribution $p(x,y)$. For the following let $T(.)$ be defined as in 4.1. Then by lemma 4.1 the minimal expected fault rate that can be obtained (using a cache of size k) given that a page y of Y was observed is

$$\begin{aligned} \pi(X|Y) &= \sum_y [1 - \sum_{x \in T(p(.|y))} p(x|y)]p(y) = \\ & \sum_y \pi(X|Y=y)p(y) \end{aligned} \tag{10}$$

Let $\{X_i\}_{i=1}^{\infty}$ be a stationary ergodic process. Similar to (see equation 5) the entropy of a stationary process we define the fault rate of a stationary ergodic sequence as

$$\Pi(\mathcal{H}) = \lim_{n->\infty} \pi(X_n|X_{n-1},\ldots,X_1) \tag{11}$$

To show that the above limit exists, we need the following lemma which shows that conditioning cannot increase expected minimal fault rate.

Lemma 4.3 *Let (X,Y) be a pair of discrete random variables as defined above. Then, $\Pi(X|Y) \leq \Pi(X)$.*

Proof:

$$\Pi(X) = 1 - \sum_{x \in T(p(.))} p(x) \tag{12}$$

$$\Pi(X|Y) = \sum_y (1 - \sum_{x \in T(p(.|y))} p(x|y))p(y) \tag{13}$$

where $p(.|y)$ is the conditional probability distribution of X given y. Hence,

$$\begin{aligned} & \Pi(X) - \Pi(X|Y) = \\ \sum_y \sum_{x \in T(p(.|y))} p(x|y))p(y) - \sum_{x \in T(p(.))} p(x) & \\ = \sum_y \sum_{x \in T(p(.|y))} p(x,y) - \sum_{x \in T(p(.))} p(x) & \\ \geq \sum_{x \in T(p(.))} \sum_y p(x,y) - \sum_{x \in T(p(.))} p(x) = 0 & \end{aligned}$$

□

Lemma 4.4 *The limit defined in 11 exists for a discrete stationary ergodic process.*

Proof:

$$\begin{aligned}\Pi(X_{n+1}|X_n,\ldots,X_1) &\leq \Pi(X_{n+1}|X_n,\ldots,X_2)\\ &= \Pi(X_n|X_{n-1},\ldots,X_1) \qquad (14)\end{aligned}$$

where the inequality follows from the fact that conditioning cannot increase the minimal expected fault rate and the equality follows from the stationarity of the process. Since $\Pi(X_n|X_{n-1},\ldots,X_1)$ is a non-increasing sequence of non-negative numbers, it has a limit. □

An immediate corollary of the following lemma (in conjunction with equations 5 and 11) is that the same lower bound as in corollary 4.1 holds for stationary ergodic processes too.

Lemma 4.5 $\pi(X|Y) \geq \frac{H(X|Y)-1-\lg k}{\lg(\frac{l}{k}-1)}$

Proof: $H(X|Y=y)$ and $\pi(X|Y=y)$ are the entropy and the minimal expected fault rate of a discrete random variable that takes values in $\mathcal{H}$. Thus the lower bound of corollary 4.1 holds for every y, i.e.,

$$\begin{aligned}\pi(X|Y=y) \geq & \quad \frac{H(X|Y=y)-1-\lg k}{\lg(\frac{l}{k}-1)}\\ \Pi(X|Y) = & \quad \sum_y \pi(X|Y=y)p(y) \geq\\ & \quad \sum_y\left(\frac{H(X|Y=y)-1-\lg k}{\lg(\frac{l}{k}-1)}\right)p(y)\\ & \quad = \frac{H(X|Y)-1-\lg k}{\lg(\frac{l}{k}-1)}\end{aligned}$$

□

Thus we can state the following theorem where we have used $\pi(H,k)$ to emphasize the dependence of π on H and k.

Theorem 4.1 *The minimal expected page fault rate $\pi(H,k)$ on a request sequence generated by a stationary ergodic process with entropy H is lower bounded by $L(H,k) = \frac{H-1-\lg k}{\lg(\frac{l}{k}-1)}$.*

4.2 Upper bound

Our upper bound will use Rissanen's universal data compression system [24] which is a variant of the Ziv-Lempel's universal compression algorithm [28].

The Ziv-Lempel algorithm parses individual sequences $< X^n >= X_1, X_2, \ldots, X_n$ into phrases. Each phrase starts with a comma, and consists of a maximal length sequence that has occurred as an earlier phrase, followed by the next symbol. We denote by v_n the number of complete phrases when parsing the finite sequence $< X^n >$. For example, the binary string $< X^n >= 0101000100$ with length $n = 10$ is parsed as $,0,1,01,00,010,0$ and contains $v_n = 5$ complete phrases and an incomplete phrase at the end. The Ziv-Lempel parsing is obtained by maintaining a dynamically growing tree data structure. Initially this tree consists of a single node, the root. Edges of the tree are labeled with symbols of the alphabet $\mathcal{H}$. Processing of a new phrase starts at the root and proceeds down the tree through edges that match the symbols of the input sequence. When the process reaches a leaf it adds a new branch labeled with the next symbol of the input sequence, which is the last symbol of this phrase. Let T_n denote the tree after processing n symbols of the input.

Rissanen [24] has studied a variant of this algorithm which generates a tree $\tilde{T}_n$. The nodes of T_n are the internal nodes of $\tilde{T}_n$. An internal node of $\tilde{T}_n$ has all its $l = |\mathcal{H}|$ possible descendents. Thus, nodes in $\tilde{T}_n$ are either leaves or have l descendents. Thus, a processing of a phrase in $\tilde{T}_n$ ends when the process reaches a leaf. The leaf is then converted to an internal node, and its l descendents are added to the tree. Note that Rissanen's variant generates exactly the same phrases as the Liv-Zempel parsing. Let v_n be the number of phrases in the parsing of the input string. It is easy to verify that T_n contains $v_n + 1$ nodes, while $\tilde{T}_n$ contains $1 + l(v_n + 1)$ nodes, namely $v_n + 1$ interior nodes and $1 + (l-1)(v_n + 1)$ leaves. The advantage of Rissanen's version is that all leaves in the the tree $\tilde{T}_n$ have equal probability of being reached while searching for a new phrase [24, 2].

Consider the following prefetching algorithm using $\tilde{T}_n$: Assume that at step n the algorithm is at node z of the tree $\tilde{T}_n$. If z is a leaf we prefetch k symbols randomly and go to the root (after making the leaf an interior node and adding l children). If z is an interior node then we prefetch the k items that correspond to the k subtrees, rooted at z, with the maximum number of leaves. When the $(n+1)$th request is revealed the process proceeds through the corresponding branch.

To analyze the above prefetching algorithm we need the following basic results proven by Ziv and Lempel [20, 28].

Theorem 4.2 *[20] The number of phrases v_n in a distinct parsing of a sequence (from an alphabet of size l) $X_1, X_2, \ldots, X_n$ satisfies*

$$v_n \leq \frac{n \lg l}{(1-\epsilon_n)\lg n} \quad \textit{where} \quad lim_{n\to\infty}\epsilon_n = 0$$

Theorem 4.3 *[28] Let $\{X_n\}$ be a stationary ergodic process with entropy rate $H(\mathcal{H})$ and let v_n be the number of phrases in a distinct parsing of a sample of length n from this process. Then*

$$limsup_{n\to\infty}\frac{v_n \lg v_n}{n} \leq H(\mathcal{H})$$

We first show a simple upper bound which shows that π is bounded above by a linear function of H.

Theorem 4.4 *The minimal expected fault rate $\pi(H,k)$ of the prefetching algorithm on a request sequence generated by a stationary ergodic process with entropy H is upper bounded by $U(H,k) = \frac{(l-k)H}{l\lg(k+1)}$.*

Proof: We assume that $l \geq k+1$, otherwise the fault rate is 0. Since we prefetch the k items corresponding to the k largest subtrees, whenever we incur a fault the symbol corresponds to a branch with at most $1/(k+1)$ leaves of the current subtree. Since the total number of leaves in the completed tree is at most $v_n(l-1)+1$ the number of faults incurred while traversing from the root to a leaf is at most $\lg_{k+1}(v_n(l-1)+1)$. Since all leaves have equal probability, the probability of a fault at a given branch is at most $1 - k/l$. Thus, the expected number of faults while processing a phrase is at most $(\frac{l-k}{l})\lg_{k+1}(v_n(l-1)+1)$, and the expected number of faults incurred while processing a sequence of length n is at most

$$\begin{aligned}
&\frac{v_n}{n}(\frac{l-k}{l})\lg_{k+1}(v_n(l-1)+1)\\
\leq\ &\frac{l-k}{l\lg(k+1)}\frac{v_n}{n}(\lg(v_n+1)+\lg l)\\
\leq\ &\frac{(l-k)H}{l\lg(k+1)} \text{ as } n\to\infty
\end{aligned}$$

using theorems 4.2 and 4.3. □

The above bound shows that the fault rate is bounded above by a linear function of H, although it is weak when $H \geq \lg k$. We can get tighter upper bounds on the fault rate by a more careful analysis. For simplicity we first show the bound for a discrete memoryless source and then we can show that the same upper bound holds for any stationary ergodic source using arguments similar to lemma 4.5.

Theorem 4.5 *Consider a discrete memoryless source of entropy H and minimal expected fault rate π. Then we have the following bound: $\pi \leq \frac{H}{\lg k+\lg e}$. Further, if $H \geq \lg k$ then, $\pi \leq 1-\frac{k}{2^H}$.*

Proof: Using the same approach as in the proof of theorem 4.4 we have the expected number of faults incurred while processing a phrase is at most $\pi\lg_{\frac{k}{1-\pi}}(v_n(l-1)+1)$ for the following reasons. At each node the probability of not taking any of the k largest subtrees is at most π by using Rissanen's interpretation. Also whenever we incur a page fault the branch we choose has at most $(1-\pi)/k$ fraction of leaves of the parent subtree. Thus the minimal expected fault rate π is bounded above by

$$\pi \leq \frac{v_n}{n}\pi\lg_{\frac{k}{1-\pi}}(v_n(l-1)+1) \tag{15}$$

$$\leq \pi\frac{H}{\lg k-\lg(1-\pi)} \tag{16}$$

$$\leq \frac{H}{1/\pi\lg k-1/\pi\lg(1-\pi)} \tag{17}$$

$$\leq \frac{H}{\lg k+\lg e} \tag{18}$$

Further since $\pi \geq 0$ we have from equation 16, $\lg k - lg(1-\pi) \leq H$ which gives $\pi \leq 1-\frac{k}{2^H}$, provided $k \leq 2^H$. □

5 Caching

In this section we study online *caching* or *demand paging*, where a page is fetched into cache only when a page fault occurs. By comparing the fault rates of two request sequences with equal entropy we will show that entropy of the request sequence alone does not fully capture the performance of online caching algorithms. Our construction uses the following two facts:

A prefetching algorithm can "simulate" a caching algorithm by prefetching at each step the k elements that are in the cache of the caching algorithm at that step. Thus, a lower bound on the fault rate of any prefetching algorithm for a given request sequence is also a lower bound on the fault rate of any caching algorithm on that sequence.

Consider a request sequence generated by a discrete memoryless source. It can be shown that the optimal online algorithm for caching in this case always keeps the $k-1$ pages with the highest probability in the cache, and leaves one slot for cache miss [13]. Thus, we can state the following theorem which follows from theorems 4.1 and 4.4. (Note that $L(H,k)$ and $U(H,k)$ are monotonically decreasing functions of k, assuming H and l are fixed.)

Theorem 5.1 *The best expected fault rate for any caching algorithm with cache size k on a request sequence generated by a discrete memoryless source with entropy H, is*

$$L(H,k) \leq \pi(k) \leq U(H,k-1).$$

Our construction uses request sequences generated by a Markov source.

Definition 5.1 *[14] A probabilistic finite state automaton (probabilistic FSA) as a quintuple $(S, \mathcal{H}, g, p, z_0)$ where S is a finite set of states with $|S| = s$, $\mathcal{H}$ is a finite alphabet of size l, g is a deterministic "next state" function that maps $SX\mathcal{H}$ into S, p_z is a "probability assignment function" for each $z \in S$ that maps $\mathcal{H}$ into $[0,1]$ with the restriction that $\sum_{i \in \mathcal{H}} p_z(i) = 1$ and $z_0 \in S$ is the start state. A probabilistic FSA when used to generate strings is called a* Markov source. *A Markov source is* ergodic *if it is irreducible and aperiodic, meaning that each state can reach every other state, and the gcd of the possible recurrence times for each state is 1. A Markov source is stationary when the start state is chosen randomly according to the steady state probabilities of the states.*

A Markov source is a very general model and is not to be confused with a Markov chain on the page request sequence which is of first order. A Markov source can have infinite order. A stationary ergodic process can be approximated by a kth order Markov process, for large k [6]. We can define the entropy of a stationary Markov source as follows.

Definition 5.2 *[14] The entropy of a Markov source M denoted by $(S, \mathcal{H}, g, p, z_0)$ is given by*

$$H_M = \sum_{z=1}^{s} q(z) H(z)$$

where $q(z)$ is the stationary (steady state) probability corresponding to state z and $H(z)$ is the entropy of the state z defined as $-\sum_{x \in \mathcal{H}} p_z(x) \lg p_z(x)$.

Consider a two state Markov source with the same probability assignment function $p(.)$ for both states. Let H be the entropy of $p(.)$. Then the entropy of the Markov source is also H. We consider two cases:

Case 1 The pages corresponding to the top $k-1$ probabilities are the same in both states. In this case the best caching strategy is similar to the discrete memoryless case, that is keep the $k-1$ pages always in the cache. Hence, the fault rate $\pi(k)$ has the same bounds as in theorem 5.1.

Case 2 The set of $k-1$ pages with the highest probabilities in state 1 is disjoint from the set of $k-1$ pages with the highest probabilities in state 2. Suppose the stationary probabilities of the two states are $1/2$ each and the transition probability from each state to the other is also $1/2$. Then it can be shown that the best caching algorithm is to keep the top $(k-1)/2$ pages of each state (assuming k is odd) in the cache. Hence the minimal expected fault rate is (by theorems 4.1 and 4.4) in the range:

$$L(H, k/2) \leq \pi(k) \leq U(H, (k-1)/2)$$

It can be shown that the intervals corresponding in the above two cases are disjoint if k is sufficiently large. Thus, although the entropy in the two scenarios are equal, the fault rates are different.

6 Discussion

We briefly discuss how our approach may be used in certain situations to allocate resources more effectively. Consider the situation where we need to partition a (malleable) cache (for prefetching) for different data sources. One plausible way is to allocate more space in the cache for a data stream with lower entropy (assuming a common source alphabet). The motivation for this comes from our bounds on the fault rate for prefetching based on entropy of the source. Our bounds show that the fault rate grows linear in H. From our lower bound (theorem 4.1) we get $k \geq 2^{\frac{H-1-\pi lgl}{1-\pi}}$. If we fix π our bound tells us that we need a larger cache for a data stream with higher entropy to achieve the same fault rate as opposed to a stream with lower entropy. Thus our bounds can be used in practice to partition a malleable cache in a better way according to the desired fault rates for different streams.

A very interesting open question which arises from our work is how accurately can entropy characterize performance of caching or list accessing algorithms in our model where requests are generated by a stationary ergodic source. Can we give an upper bound on the average work done by any list accessing algorithm as a function of entropy alone? We feel that entropy *alone* does not characterize the performance of any (or the best) online list accessing algorithm. We know that prefetching gives a (weak) lower bound on the minimal expected fault rate for caching (demand paging). Here also it seems that entropy alone does not tightly capture the minimal expected fault rate. It would be interesting to come up with a parameter which along with entropy will determine or (at least tightly upper bound) the performance of a caching algorithm. Another interesting area of research is to explore whether entropy gives good performance bounds for other online problems known in literature.

Acknowledgments

We are thankful to John Savage and Ye Sun for useful discussions.

References

[1] S. Albers and M. Mitzenmacher, Average Case Analysis of List Update Algorithms, with Applications to Data Compression, *Algorithmica*, **21**, 1998, 312-329.

[2] P. Algoet, Universal Schemes for Prediction, Gambling and Portfolio Selection, *Annals of Probability*, **20**(2), 1992, 901-941.

[3] P. Algoet and T.M. Cover, Asymptotic Optimality and Asymptotic Equipartition Property of Log-Optimal Investment, *Annals of Probability*, **16**, 1988, 876-898.

[4] 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), 1986, 320-330.

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

[6] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[7] K. Curewitz, P. Krishnan and J.S. Vitter, Practical Prefetching Via Data Compression, In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1993, 257-266.

[8] Derek Chiou, Prabhat Jain, Srinivas Devadas and Larry Rudolph, Dynamic Cache Partitioning via Columnization, in *Proceedings of Design Automation Conference*, Los Angeles, June 2000.

[9] P. Elias, Universal Codeword Sets and the Representation of the Integers, *IEEE Transactions on Information Theory*, **21**(2), 1975, 194-203.

[10] A. Fiat, R.M. Karp, M. Luby, L. A. McGeoch, D.D. Sleator and N.E. Young, On Competitive Algorithms for Paging Problems, *Journal of Algorithms*, **12**, 1991, 685-699.

[11] M. Feder and N. Merhav, Relations between Entropy and Error Probability, *IEEE Transactions on Information Theory*, **40**(1), 1994, 259-266.

[12] M. Feder, N. Merhav and M. Gutman, Universal Prediction of Individual Sequences, *IEEE Transactions on Information Theory*, **38**, 1992, 1258-1270.

[13] P.A. Franaszek and T.J. Wagner. Some Distribution-free Aspects of Paging Performance, *Journal of the ACM*, **21**, 1974, 31-39.

[14] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

[15] G.H. Gonnet, J.I. Munro, and H. Suwanda. Exegesis of Self-organizing Linear Search, *SIAM Journal of Computing*, **10**, 1982, 613-637.

[16] M.E. Hellman and J. Raviv, Probability of Error, Equivocation and the Chernoff Bound, *IEEE Transactions on Information Theory*, **16**(4), 1970, 368-372.

[17] J.L. Hennessey and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd edition, Morgan Kaufmann, 1996.

[18] A.R. Karlin, S.J. Phillips and P. Raghavan, Markov Paging, In *Proc. of the 33rd IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1992, 208-217.

[19] J. Kelly, A New Interpretation of Information Rate, *Bell Sys. Tech. Journal*, **35**, 1956, 917-926.

[20] A. Lempel and J. Ziv, On the Complexity of Finite Sequences, *IEEE Transactions on Information Theory*, **22**, 1976, 75-81.

[21] The Malleable Caches Project at MIT, http://www.csg.lcs.mit.edu/mcache/index.html

[22] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[23] D. Ornstein, Guessing the Next Output of a Stationary Process, *Israel J. Math.*, **30**, 292-296.

[24] J. Rissanen, A Universal Data Compression System, *IEEE Transactions on Information Theory*, **29**(5), 1983, 656-664.

[25] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules, *Communications of the ACM*, **28**(2), 1985, 202-208.

[26] Edward Suh and Larry Rudolph, Adaptive Cache Partitioning, *CSG-Memo 432*, Lab. for Computer Science, MIT, June 2000.

[27] J.S. Vitter and P. Krishnan. Optimal Prefetching Via Data Compression, *Journal of the ACM*, **43**(5), 1996, 771-793.

[28] J. Ziv and A. Lempel, Compression of Individual Sequences via Variable Rate Coding, *IEEE Transactions on Information Theory*, **24**(5), 1978, 530-536.