Entropy-Based Bounds for 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. 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 classical online problems — list accessing, prefetching, and caching. Our bounds relate the performance of the best online algorithm to the *entropy*, a parameter intrinsic to the characteristics of the request sequence. This is in contrast to the *competitive ratio* parameter of competitive analysis which quantifies the performance of the online algorithm with respect to an optimal offline algorithm. For the prefetching problem, we give explicit upper and lower bounds for the performance of the best prefetching algorithm in terms of the entropy of the request sequence. In contrast, we show that the entropy of the request sequence alone does not fully capture the performance of online list accessing and caching algorithms.

Keywords: Online Algorithms, Performance Bounds, Entropy, Stochastic Process, Prefetching, Caching, List Accessing.

[*]Department of Computer Science, Purdue University, West Lafayette, IN 47907-2066, USA. E-mail: gopal@cs.purdue.edu. Part of this work was done while the author was at Brown University.

[†]Department of Computer Science, Brown University, Providence, RI 02912-1910, USA. E-mail: eli@cs.brown.edu.

The authors were supported in part by NSF grant CCR-9731477. Preliminary versions of this paper appeared in the proceedings of the 12th annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Washington D.C., 2001, and in the proceedings of the 38th Annual Conference on Information Sciences and Systems (CISS), Princeton University, NJ, 2004.

1 Introduction and Motivation

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"[21]. 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 caches [29, 38, 9]. 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 [6] 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 study bounds based on entropy for three classical online problems — list accessing, prefetching, and caching. Our bounds relate the performance of the best online algorithm to the *entropy*, a parameter intrinsic to the characteristics of the request sequence. This is in contrast to the *competitive ratio* parameter of competitive analysis which quantifies the performance of the online algorithm with respect to an optimal offline algorithm. To the best of our knowledge, this is the first work to focus on entropy-based bounds in the context of online algorithmic problems studied in computer science.

1.1 Related Work and Comparison

The three online problems considered here were extensively studied in the competitive analysis framework. It has been shown in [37] that the competitive ratio[1] of the *move to front (MTF) algorithm* for the list accessing problem is two. 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 [18] 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 distribution D and proved that for any distribution D, $\mathrm{TIMESTAMP}(D) \leq 1.34 \times \mathrm{SOPT}(D)$, and with high probability, $\mathrm{TIMESTAMP}(D) \leq 1.5 \times$

[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.

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, 18].

For online caching (or demand paging) the well known LRU (Least Recently Used) has a competitive ratio of k [37], where k is the cache size, while the randomized MARKER algorithm is $2 \log k$ competitive [12]. Franaszek and Wagner [16] studied a model in which every request is drawn from a discrete memoryless source. Karlin et al. [22] 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. Lund et al. [27] improve on the above result by proposing an efficient randomized 4-competitive online caching algorithm that works for *any* distribution $\mathcal{D}$ but it needs to know for (each pair of) pages p and q the probability that p will next be requested before q. Using the randomized algorithm of Lund et al., Pandurangan and Szpankowski [36] give a *universal* online caching algorithm based on pattern matching that works on a large class of models (including Markov sources) but does not need any knowledge of the input model.

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 [39] consider a model where the sequence of page requests is assumed to be generated by a *Markov source*, a model which is closest in spirit to our model of a stationary ergodic process. They show that the fault rate of a Ziv-Lempel [42] 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 $n \to \infty$. In a subsequent work [40], the same authors derive a randomized algorithm for prefetching and compare its performance to the optimal finite state prefetcher. Thus, their analysis can be considered to fall in the competitive analysis framework.

All the three problems considered here — list accessing, prefetching, and caching can be formulated as "sequential decision problems" [19, 31]. That is, we have a temporal sequence of observations (i.e., the request sequence) $x_1^n = x_1, x_2, \ldots, x_n$, for which corresponding actions $b_1, b_2, \ldots, b_n$ result in instantaneous losses $l(b_t, x_t)$, for each time instant t, $1 \le t \le n$, where $l(.,.)$ denotes a non-negative loss function. The action b_t, for all t, is a function of the previous observations x^{t-1} only; hence the sequence of actions can be considered as an online algorithm or strategy. A normalized loss

$$L = \frac{1}{n} \sum_{t=1}^{n} l(b_t, x_t) \tag{1}$$

accumulates instantaneous loss contributions from each action-observation pair. In the sequential decision problem the goal is to find an online strategy that approximates, in the long run and for an *arbitrary individual sequence* of observations, the performance of the best *constant* offline algorithm which has full knowledge of the given sequence of observations. In this setting, the quantity of interest is termed *regret* which is the additional loss incurred by the on-line strategy over the offline algorithm.

Statisticians and information theorists have extensively studied the sequential decision problem in two settings: a probabilistic framework, in which the sequence of requests is viewed as a sample of a random process, or using an individual sequence approach, i.e., *comparing* the performance of the online strategy for an *arbitrary* sequence with certain classes of *competing offline strategies* — such as in the sequential decision approach or with finite state machines. The latter setting is essentially in the framework of competitive analysis. Universal lossless data compression, gambling, prediction, and portfolio selection have been studied as sequential decision problems, see [19, 31]. In [42] for the problem of universal compression of individual sequences the class of competing offline strategies is extended to include all finite-state encoders (as opposed to being constant).

Other results in this setting are for data compression [14, 30], prediction [14], prefetching (under different loss functions) [41], and for general loss functions with memory [32].

In the probabilistic setting, most previous work (see the discussion of Algoet's work [3] below or the work of Vitter and Krishnan [39]) show *convergence* of certain online strategies to the optimum. A general result in this setting was shown by Algoet [3]: if the request sequence is generated by a stationary ergodic process then it is shown that the optimum strategy is to select an action that minimizes the conditional expected loss given the currently available information at each step and this strategy is shown to be asymptotically optimal in the sense of the strong law of large numbers. See [2] for further results on universal schemes for prediction, gambling and portfolio selection.

Our work assumes the probabilistic setting, i.e., the request sequence is assumed to be generated by a stationary ergodic process; however there is an important difference in our approach. Our work tries to *characterize* (i.e., obtain bounds on) the optimum attainable in terms of *entropy* of the sequence — a single parameter of the sequence. Why do we choose entropy? First, it has been shown that entropy can tightly characterize the optimal performance of many problems. For example a fundamental result in information theory is that entropy is the best achievable performance for lossless data compression in a probabilistic setting and there are well-known data compression algorithms that closely match this bound. Kelly [24] studied 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 [24, 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 [4, 7]. Second, since entropy is a measure of randomness of the request sequence, it is a reasonable to expect it to characterize the "intrinsic bottleneck" in the performance of any online algorithm.

However, we don't expect entropy to fully characterize (in the sense of the above mentioned examples such as horse race or data compression, i.e., either precisely or in terms of non-trivial lower and upper bounds) online performance for all problems. For example, the problems which are well characterized by entropy such as lossless data compression, prediction (prefetching is a generalization of prediction) can be thought of as sequential decision problems with memoryless loss functions i.e., the loss does not depend on previous action-request pairs. On the other hand, problems such as list accessing and caching involve loss functions that are not memoryless. In fact, we show that entropy alone is not the best parameter to characterize these problems. However, our non-trivial entropy-based lower bounds characterizes the "intrinsic bottleneck" of randomness in the performance of online algorithms for these problems. Our entropy-based results should thus be viewed as a first step in the above "characterization" approach. (See Section 6 for interesting questions in this regard.)

Our results on list accessing are based on the work of Bentley et al. [5] 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 [13] 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 k is 1. A tight lower bound on this error probability is given by Fano's inequality [7, 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 information theory literature on predicting binary sequences (corresponding to prefetching in a universe of two pages with cache of size 1) [14], 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 Ziv-Lempel universal algorithm for data compression [42]. Our approach is inspired by the work of Vitter and Krishnan [39] who show that a similar Ziv-Lempel based prefetching algorithm converges asymptotically to the optimal fault-rate when the sequence is generated by Markov sources (cf. Section 1.1, para 3).

Finally, we mention that since the publication of the preliminary version of this paper [35], the entropy-based bounds obtained here have been shown to be useful in understanding the performance of online algorithms in real applications. Fonseca et al. [15] use these bounds to explain the strong correlation between entropy and the performance of LRU algorithm in Web caches.

1.2 Our 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 performance of the best online algorithm for these problems.

We assume that the sequence of requests is generated by a *discrete stationary ergodic process* [17, definition 3.5.13], a very general stochastic source which is well-studied in information theory. It includes powerful models such as memoryless and (stationary) Markov sources [17, 39, 8].

For the list accessing problem we show that any online algorithm requires an average work of $\frac{2^H}{\lceil \lg l+1 \rceil}$ steps per item access, where H is the entropy of the input source and l is the total number of items. When the request sequence is generated by a discrete memoryless source, we show a somewhat better lower bound of $\frac{2^H}{e} - 1$. We then show that this bound can be quite weak in some cases. In particular, we give instances to show that two *different* memoryless sources with the same entropy can have very different work bounds.

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 a previously 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 different techniques [17, pages 520-521], [13, 20]. Our upper bound is derived by analyzing the performance of a prefetching algorithm based on Ziv-Lempel data compression algorithm.

Finally, we consider the online caching or demand paging problem, which is related to prefetching. We show that bounds similar to prefetching hold when requests are generated by a discrete memoryless source. However, unlike prefetching, for higher order Markov sources (in particular when requests are generated by a Markov chain) we show that different information sources with the same entropy can have very different minimal fault rates. Thus, in the case of caching and list accessing, entropy alone is not sufficient to characterize online performance.

2 Preliminaries

We model a request sequence for an online process as an indexed sequence of (discrete) random variables (also called a *stochastic process*), denoted by $\{X_i\}_{i=1}^{\infty}$ taking values in a finite alphabet set Σ. Let $l = |\Sigma|$. We use the notation x_m^n to denote a sequence $x_m, \ldots, x_n$ where $x_i \in \Sigma$. Similarly, X_m^n denotes the sequence $< X_m, \ldots, X_n >$ of random variables. We always assume that a probability measure exists and we write $\Pr(X_1^n) = \Pr(X_i = x_i, 1 \le i \le n, x_i \in \Sigma)$ for the probability mass, where we use lowercase letters for a realization of a stochastic process. 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 Σ and probability mass function $p(x) = \Pr\{X = x\}, x \in \Sigma$ is

$$H(X) = -\sum_{x\in\Sigma} p(x) \lg p(x). \tag{2}$$

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

$$H(X, Y) = -\sum_{x_1\in\Sigma}\sum_{x_2\in\Sigma} p(x_1, x_2) \lg p(x_1, x_2). \tag{3}$$

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

$$H(X_2|X_1) = \sum_{x_1\in\Sigma} p(x_1)H(X_2|X_1 = x) = -\sum_{x_1\in\Sigma}\sum_{x_2\in\Sigma} p(x_1, x_2) \lg p(x_2|x_1). \tag{4}$$

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

$$H_n(\Sigma) = \frac{1}{n}H(X_1, X_2, \ldots, X_n). \tag{5}$$

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

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

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.,*

$$\Pr\{X_1^n = x_1^n\} = \Pr\{X_{1+t}^{n+t} = x_1^n\}$$

for every shift t and for all $x_1^n \in \Sigma^n$.

It can be shown that [17, Theorem 3.5.1] for *stationary processes* (with finite $H_1(\Sigma)$) the limit $H(\Sigma)$ exists and

$$\lim_{n\to\infty} H_n(\Sigma) = \lim_{n\to\infty} H(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) = H(\Sigma); \tag{6}$$
$$H(X_n|X_{n-1}, \ldots, X_1) \text{ is non-increasing with } n; \tag{7}$$
$$H_n(\Sigma) \geq H(X_n|X_{n-1}, \ldots, X_1); \tag{8}$$
$$H_n(\Sigma) \text{ is non-increasing with } n. \tag{9}$$

An important special case of a stationary ergodic[2] process is when $X_1, X_2, \ldots$ are independent and identically distributed random variables (also called as a *discrete memoryless source*). For such a source,

$$H(\Sigma) = \lim_{n\to\infty} \frac{1}{n}H(X_1, X_2, \ldots, X_n) = \lim_{n\to\infty} \frac{nH(X_1)}{n} = H(X_1).$$

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).

[2]Informally, a process is ergodic if it cannot be "separated" into different persistent modes of behavior. For a formal (measure-theoretic) definition we refer to [17]. An important consequence of ergodicity is that the law of large numbers applies to such sources.

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 [6] 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_i of the ith item x_i is the number of links traversed by the algorithm to locate the item, starting at the head of the list. (The access cost might depend on the previous accesses.) 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_i be the total cost associated with servicing the ith item x_i. c_i includes both the access cost a_i and any transposition cost incurred before servicing x_i.

Following Bentley et al. [5] 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 reals).

Let $\Pr(x_1^n) = \Pr(X_1^n = x_1^n)$ denote the probability mass function of the request sequence from a finite alphabet set Σ and let $c(x_1^n)$ be the total cost of servicing a sequence x_1^n. Let $\bar{c_n} = \sum_{x_1^n \in \Sigma^n} c(x_1^n) \Pr(x_1^n)$ be the average cost of accessing an item by any deterministic algorithm $\mathcal{A}$ on a sequence of requests X_1^n generated by a stationary ergodic source of entropy H (or H_n for sequence of length n). Assume that 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. We have the following theorem.

Theorem 3.1 *$\bar{c_n} \geq f^{-1}(H)$.*

Proof: Since the total cost of servicing an item is at least the access cost we have,

$$\bar{c_n} \geq \sum_{x_1^n \in \Sigma^n} \left(\sum_{i=1}^{n} a_i \right) \Pr(x_1^n)$$

where a_i — the access cost of item x_i — is the distance 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_i using $f(a_i)$ bits, then by variable-length source coding theorem [17, theorem 3.5.2] and by equations 6 to 9,

$$\sum_{x_1^n \in \Sigma^n} \left(\sum_{i=1}^{n} f(a_i) \right) \Pr(x_1^n) \geq H_n \geq H. \tag{10}$$

Since f is concave, by Jensen's inequality, and by using 10,

$$f(\bar{c_n}) \geq f\left(\sum_{x_1^n \in \Sigma^n} \left(\sum_{i=1}^{n} a_i \right) \Pr(x_1^n) \right) \geq \sum_{x_1^n \in \Sigma^n} \left(\sum_{i=1}^{n} f(a_i) \right) \Pr(x_1^n) \geq H.$$

Hence, $\bar{c_n} \geq f^{-1}(H)$. □

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 [3] [11]. 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:

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 [33, 28]. Thus the lower bounds derived in the corollaries hold for randomized algorithms also.

The above lower bounds are for arbitrary stationary (ergodic) sources. We can show a somewhat better lower bound when the request sequence is generated by a discrete memoryless source. Consider a memoryless source with probability distribution $D = \{p_1, \ldots, p_l\}$ where p_i is the probability of accessing the ith symbol of the alphabet (w.l.o.g. assume that $p_1 \geq \cdots \geq p_l$). The best strategy for memoryless sources is to arrange the items in the list in decreasing order of request probabilities and never reorder them thereafter [6]. This algorithm is called SOPT (static offline algorithm). Although this is not strictly an online algorithm, it gives a lower bound on the performance of any online algorithm on memoryless sources; also it has been shown that several online algorithms such as MTF [18] and TIMESTAMP [1] achieve a performance which is within a small constant factor of SOPT. The expected access cost per item of SOPT is $\sum_{i=1}^{l} i p_i$. The following theorem gives a lower bound on the expected cost of accessing an item by SOPT (and hence any online algorithm) based on the entropy of the input distribution D.

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

Proof: Let w be the average cost of accessing an item by SOPT. Then the probability distribution that maximizes the entropy subject to a constraint on the average is the geometric distribution [7, Lemma 12.10.2]. Thus using the entropy of the geometric distribution with mean w as an upper bound we have

$$H \leq (w+1)\lg(w+1) - w \lg w \leq \lg(w+1) + \lg((1 + 1/w)^w)$$

$$\leq \lg(w+1) + \lg e.$$

This implies that $w \geq 2^{H - \lg e} - 1 = \frac{2^H}{e} - 1$. □

[3] 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.

The above lower bound is almost tight in some cases, but can be quite loose in general. For example, consider a discrete memoryless source with D being the uniform distribution. The average work needed per item for SOPT is $(l+1)/2$ (MTF is within a constant factor of this) and the above Theorem gives a bound which is within a constant factor. On the other hand, consider a memoryless source with the distribution $(1-\frac{1}{\sqrt{l}}, \underbrace{\frac{1}{(l-1)\sqrt{l}}, \ldots, \frac{1}{(l-1)\sqrt{l}}}_{l-1\ terms})$. The entropy of this distribution is $O(\lg(l)/\sqrt{l})$, but the average work per item is at least $\Theta(\sqrt{l})$ — much higher than the lower bound given by the above Theorem. Also, consider the distribution $(\underbrace{\frac{1-1/l}{\sqrt{l}}, \ldots, \frac{1-1/l}{\sqrt{l}}}_{\sqrt{l}\ terms}, \underbrace{\frac{1/l}{l-\sqrt{l}}, \ldots, \frac{1/l}{l-\sqrt{l}}}_{(l-\sqrt{l})\ terms})$: the work needed is $\Theta(\sqrt{l})$ — essentially same as the previous scenario, but the entropy is $\Theta(\lg l)$. Thus, *different* information sources with the same entropy can have very different fault rates; in this sense, entropy alone does not fully capture the performance of list accessing algorithms.

4 Prefetching

As in [39] we consider the following formalization of the prefetching problem: we have a collection Σ of pages in memory and a cache of size k, and typically $k \ll |\Sigma|$. 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 $l = |\Sigma|$. We assume that the request sequence $<X>= X_1, X_2, \ldots$ is generated by a stationary ergodic process with entropy rate H. Given a prefetching algorithm $\mathcal{A}$, let $c_i^{\mathcal{A}}$ be one or zero depending on whether a fault was incurred or not while servicing item X_i. We define

$$\Pi^{\mathcal{A}} = \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} E[c_i^{\mathcal{A}}] \tag{11}$$

to be the *long-term expected fault rate* of the sequence by a prefetching algorithm $\mathcal{A}$. Note that the expectation is taken with respect to both the random variable X_i (which in general can depend on $X_{i-1}, \ldots, X_0$) and the random choices made by the algorithm. We are interested in the *minimum long-term expected fault rate (MLEF)* of a request sequence i.e., the long-term expected page fault rate of the best prefetching algorithm possible for the request sequence. We show the existence of this quantity (henceforth denoted by Π) when the request sequence is generated by a stationary ergodic process. Our goal is to characterize MLEF in terms of the entropy of the source generating the sequence.

We will first show a lower bound for a discrete memoryless source. Then using techniques from [13] we will show that the same bound hold for a stationary ergodic source. We observe that the optimal prefetching strategy in a discrete memoryless source is the following obvious deterministic strategy:

Lemma 4.1 *Let $p(.)$ be a probability distribution on Σ. Suppose each page in the sequence is drawn i.i.d with probability distribution $p(.)$. Then the optimal strategy is to prefetch the pages (in the cache) with the top k probabilities. The expected fault-rate for this strategy for this discrete memoryless source is defined by $\pi = 1 - \sum_{x\in T(p(.))} p(x)$, where $T(p(.))$ is the set of pages with the top k probabilities in the distribution $p(.)$. The MLEF for the above discrete memoryless source is equal to π.*

Proof: Let c_i be an indicator random variable for the fault incurred while servicing X_i using the strategy of picking the top k pages. Then, $E[c_i] = \pi$, where π is as defined in the Lemma. Then the long-term expected fault rate of the strategy is

$$\limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} E[c_i] = \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \pi = \pi.$$

We show that π is a lower bound on the long-term expected fault rate of any strategy. This will imply that the above strategy is optimal. Let $c_i^{\mathcal{A}}$ be an indicator random variable for the fault incurred on X_i by any prefetching algorithm $\mathcal{A}$. Then $c_i^{\mathcal{A}}$ stochastically dominates c_i (a consequence of Bayes' decision rule, for example see [20]), thus

$$\limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} E[c_i^{\mathcal{A}}] \geq \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} E[c_i] = \pi.$$

□

4.1 Lower Bound

We first prove the lower bound for a discrete memoryless source, generalizing the result in Feder and Merhav [13].

Our goal is to relate the fault rate of the prefetching strategy of lemma 4.1 to the entropy of the source. Consider a discrete random variable X, and let $p(i) = Pr\{X = i\}$ for $i \in \Sigma$. 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 minimum 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 Σ. Then the relative entropy (or Kullback Leibler distance) between $p(x)$ and $P_{max}(\pi)$ is given by [7, definition 2.26]

$$\begin{aligned}&\sum_{x\in\Sigma} p(x)\lg(p(x)/P_{max}(\pi))\\ &= -H(X) + \sum_{x\in\Sigma} p(x)\lg 1/P_{max}(\pi)).\end{aligned}$$

Since the relative entropy is always positive [7, Theorem 2.6.3] we have

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

□

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 as in corollary 4.1 holds for any stationary ergodic process. First we need to define the following. Let (X,Y) be a pair of discrete random variables (each with range Σ) with joint distribution $p(x,y)$. For the following let $T(.)$ be defined as in Lemma 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

$$\begin{aligned} \Pi(X|Y) &= \sum_y [1 - \sum_{x \in T(p(.|y))} p(x|y)]p(y) \qquad (12) \\ &= \sum_y \Pi(X|Y=y)p(y). \qquad (13) \end{aligned}$$

Let $\{X_i\}_{i=1}^{\infty}$ be a stationary ergodic process. Similar to (see equation 6) the entropy of a stationary process we define the MLEF of a stationary ergodic process as

$$\Pi(\Sigma) = \lim_{n\to\infty} \Pi(X_n|X_{n-1},\ldots,X_1). \qquad (14)$$

As in the case of entropy, we can show that the above definition is equivalent to the one given in equation 11:

$$\Pi(\Sigma) = \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \Pi(X_i|X_{i-1},\ldots,X_1).$$

Theorem 4.1 *Let $\{X_i\}_{i=1}^{\infty}$ be a stationary ergodic process. Then,*

$$\lim_{n\to\infty} \Pi(X_n|X_{n-1},\ldots,X_1) = \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \Pi(X_i|X_{i-1},\ldots,X_1) = \Pi(\Sigma).$$

In the special case when $\{X_i\}_{i=1}^{\infty}$ is a discrete memoryless process, we have

$$\Pi(\Sigma) = \Pi(X_1) = \pi$$

where π is as defined in lemma 4.1.

Proof: We show that $\lim_{n\to\infty} \Pi(X_n|X_{n-1},\ldots,X_1)$ exists in Lemma 4.4 below. As in the case of entropy, since the above limit converges, the Cesaro mean [7][Theorem 4.2.3, page 64] implies that the two limits are equal.

If $X_1, X_2, \ldots$ are independent and identically distributed, then using equation 12,

$$\lim_{n\to\infty} \Pi(X_n|X_{n-1},\ldots,X_1) = \lim_{n\to\infty} \Pi(X_n) = \pi(X_1).$$

□

To show that the limit of 14 exists, we need the following lemma which shows that conditioning cannot increase minimal expected 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:

$$\Pi(X) = 1 - \sum_{x \in T(p(.))} p(x). \tag{15}$$

$$\Pi(X|Y) = \sum_y (1 - \sum_{x \in T(p(.|y))} p(x|y))p(y). \tag{16}$$

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 14 exists for a discrete stationary ergodic process.*

Proof:

$$\begin{aligned}
\Pi(X_{n+1}|X_n, \ldots, X_1) &\leq \Pi(X_{n+1}|X_n, \ldots, X_2) \\
&= \Pi(X_n|X_{n-1}, \ldots, X_1)
\end{aligned} \tag{17}$$

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 6 and 14) 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 Σ. Thus the lower bound of corollary 4.1 holds for every y, i.e.,

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

□

Thus we can state the following theorem where we have used $L(H, k)$ to emphasize the dependence of the lower bound on H and k.

Theorem 4.2 *The MLEF Π 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 uses Ziv-Lempel's universal data compression algorithm [42]. Our idea is to use Ziv-Lempel's algorithm to give a bound on the MLEF in terms of the entropy of the stationary ergodic source.

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 Σ. 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. 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.

Consider the following prefetching algorithm using T_n: Assume that at step n the algorithm is at node z of the tree T_n. If z is a leaf we prefetch k symbols randomly and go to the root (after adding a new branch labeled with the new symbol). If z is an interior node then we prefetch the k items that correspond to the k largest subtrees rooted at z. When the $(n+1)$th request is revealed the process proceeds through the corresponding branch. One simple way of finding out the k largest subtrees is to maintain a count on each node denoting the number of times that node was visited when searching for a phrase. (We start with a count of zero at the root initially, and a new leaf also gets initialized to count zero. Note that the count on a node is equal to the total number of nodes in the subtrees hanging from it.) Thus, the algorithm will choose the k symbols corresponding to the nodes having the k largest counts.

As mentioned in Section 1.1, Vitter and Krishnan [39] use essentially the above Ziv-Lempel based prefetching algorithm and show that the fault rate of this 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 $n \to \infty$. Feder et al. [14] also use the above Ziv-Lempel parsing scheme to design an efficient prediction procedure on *individual* binary sequences (corresponding to prefetching in a universe of two pages with cache of size 1) that is asymptotically optimal compared to any finite state predictor. There is also earlier work (e.g., see [25]) which show how the above Ziv-Lempel parsing scheme can be used for asymptotically optimal data compression. In contrast to these results that show *convergence* of the Ziv-Lempel based algorithms (whether for prefetching, prediction, or data compression, either in a probabilistic or individual sequence setting) to the optimum, our goal here is to analyze the above Ziv-Lempel scheme to obtain an *upper bound* on the optimum attainable in terms of *entropy* of the source.

To analyze the above prefetching algorithm we need the following basic results proven by Ziv and Lempel [26, 42].

Theorem 4.3 *[26] 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*

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

Theorem 4.4 *[42] Let $\{X_n\}$ be a stationary ergodic process with entropy rate $H(\Sigma)$ and let v_n be the number of phrases in a distinct parsing of a sample of length n from this process. Then*

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

with probability 1.

We show that the MLEF Π is bounded above by a linear function of H when the request sequence is generated by a stationary ergodic source.

Theorem 4.5 *The MLEF Π on a request sequence generated by a stationary ergodic process with entropy H is upper bounded by $U(H,k) = \frac{H}{\lg(k+1)}$.*

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 (k largest counts on the respective nodes), whenever we incur a fault the symbol corresponds to a branch with at most $1/(k+1)$ nodes of the current subtree. Since the total number of nodes in the tree is at most $v_n + 1$ the number of faults incurred while processing a phrase (i.e., while traversing from the root to a leaf) is at most $\lg_{k+1}(v_n+1)$. Note that this is a worst-case bound on the number of faults incurred for processing a phrase during the first n requests. Thus, the fault rate incurred while processing a sequence of length n is at most

$$\begin{aligned} &\frac{v_n}{n}\lg_{k+1}(v_n+1) \\ \leq\ &\frac{1}{\lg(k+1)}\frac{v_n}{n}\lg(v_n+1). \end{aligned}$$

Thus the fault rate is asymptotically upper bounded by

$$\begin{aligned} &\limsup_{n\to\infty}\frac{1}{\lg(k+1)}\frac{v_n}{n}\lg(v_n+1) \\ \leq\ &\frac{H}{\lg(k+1)} \text{ with probability 1} \end{aligned}$$

using theorems 4.3 and 4.4.

Thus, MLEF is upper bounded by $\frac{H}{\lg(k+1)}$ by the bounded convergence theorem ([10, page 16]). □

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 [6]. Similar to the prefetching problem we study entropy-based bounds on the MLEF. First we note that a lower bound on the fault rate of the best prefetching algorithm for a given request sequence is also a lower bound on the fault rate of any caching algorithm on that sequence. This is because, 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 the lower bound of Theorem 4.2 holds for any online caching algorithm. What about the corresponding upper bound? We can show an upper bound (that is essentially the same as that of prefetching) when the request sequence is generated by a discrete memoryless source. However, this bound (unlike prefetching) does not hold for more general sources. We show that two *different* information sources with the *same* entropy can have very different minimal fault rates. Thus entropy of the request sequence alone does not fully capture the performance of online caching algorithms as in the case of prefetching.

Consider a request sequence generated by a discrete memoryless source. We can state the following theorem which follows from theorems 4.2 and 4.5. (Note that $L(H,k)$, $U(H,k)$ are monotonically decreasing functions of k, assuming H and l are fixed.)

Theorem 5.1 *For the caching problem with cache size k, the MLEF Π on a request sequence generated by a discrete memoryless source with entropy H, is bounded as:*

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

Proof: Since the optimal prefetching strategy that always keeps the top k pages in the cache is a lower bound on the MLEF, we have $\Pi \geq L(H,k)$. On the other hand, the optimal caching strategy is to always keep the top $k-1$ pages (with the highest probability) in the cache, and leaves one slot for cache miss [16]. This is better than the best prefetching strategy using only $k-1$ cache slots. Thus, $\Pi \leq U(H,k-1)$. $\square$

However, the above upper bound does not hold for higher-order Markov sources (where the ith request depends on previous requests). For example, consider the sequence generated by the following (stationary) Markov chain M (which is a first-order source where the ith request depends only on the $(i-1)$th request). The page request sequence is generated in a natural way from the Markov chain by a random walk (according to the transition probabilities) on the states where there is a one-to-one correspondence between the set of states and the set of pages (see e.g., [22]).

Definition 5.1 *M has l states labeled as $s_0,\ldots,s_{l-1}$, corresponding to the l alphabets. The transition probabilities are (for $0 \leq i \leq l-1$):*

$$\Pr(s_i,s_j) = \begin{cases} 1-\frac{1}{l^2} & : \quad \textit{if } j = (i+1) \textit{ mod } l \\ \frac{1}{l^2(l-1)} & : \quad \textit{if } j \neq (i+1) \textit{ mod } l \end{cases}$$

Lemma 5.1 *The entropy rate of the above stationary Markov chain is $1/l^2 + O(1/l^4)$. However, MLEF of this chain is at least $1-(k-1)/l + O(1/l^2)$.*

Proof: The entropy of a stationary Markov chain can be computed using the following formula (see e.g., [7]):

$$H = \sum_i \mu_i \sum_j -\Pr(s_i,s_j)\lg\Pr(s_i,s_j)$$

where μ_i is the stationary probability of state s_i. For the Markov chain M, $\mu_i = 1/l$ for all s_i. A simple calculation yields the result.

To get a lower bound the long-term expected fault rate (and thus the MLEF), we partition the request sequence into disjoint subsequences of length l and compute a bound on each subsequence as follows. A length l subsequence has a probability of $(1-1/l^2)^l$ to have all distinct l items; any caching algorithm has to incur at least $l-k$ page faults when all the l requests are different. Thus the MLEF is at least $\frac{l-k}{l}(1-1/l^2)^l = 1-(k-1)/l + O(1/l^2)$. $\square$

From the above lemma, it is clear that the upper bound of Theorem 5.1 is not valid even for a first order Markov source. Furthermore, the above example, shows that entropy alone does not fully capture the MLEF. To see this, consider a source with entropy $\lg l$ (the highest possible) which is a discrete memoryless source with uniform distribution over the alphabet. It is easy to see that the MLEF (of the best caching algorithm) on this source is $1-k/l$, which is essentially the same as that of the above Markov chain M; however, their entropy rates are very different (one tends to zero while the other diverges as l increases).

6 Conclusion and Open Questions

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 4.2) 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 to partition a malleable cache in a better way according to the desired fault rates for different streams.

We conclude with a few open questions. Our upper bound for prefetching is interesting only when $H \leq \lg(k+1)$. Can one get tighter bounds when $H > \lg(k+1)$? For example, when $H = \lg l$, i.e., for the uniform distribution, the fault rate achievable is $1 - k/l = 1 - k/2^H$. It will be enough to show a tight bound for memoryless sources; again a Ziv-Lempel based approach might work.

We showed instances where entropy alone is not sufficient to capture the performance of online caching and list accessing algorithms. An interesting question is to come up with a parameter (which depends on the source) which along with entropy will characterize the performance of list accessing or caching algorithms. In the context of these instances, a possible candidate is a parameter which measures the average number of *different symbols* per fixed sequence length (say l) emitted by the source. Another interesting area of research is to explore whether entropy (or some other parameter intrinsic to the request sequence) gives good performance bounds for other online problems known in literature.

Acknowledgments

We thank Wojciech Szpankowski, John Savage, and Ye Sun for useful discussions. We thank the anonymous referees for their useful comments and suggestions for improving the presentation of the paper.

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. The Strong Law of Large Numbers for Sequential Decisions under Uncertainty, *IEEE Transactions on Information Theory*, **40**(3), 1994, 609-633.

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

[5] 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.

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

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

[8] 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.

[9] D. Chiou, P. Jain, S. Devadas, and L. Rudolph. Dynamic Cache Partitioning via Columnization, in *Proceedings of Design Automation Conference*, Los Angeles, June 2000.

[10] R. Durrett. *Probability: Thoery and Examples*, second edition, Duxbury Press, 1996.

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

[12] 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.

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

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

[15] R. Fonseca, V. Almeida, and M. Crovella. Localilty in a Web of Streams, *Comunications of the ACM*, **48**(1), 2005, 82-88. Conference version with B. Abrahao in *Proceedings of the IEEE INFOCOM*, 2003.

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

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

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

[19] J.F. Hannan. Approximation to Bayes risk in repeated plays, in *Contributions to the Theory of Games, Vol. 3, Annals of Mathematics Studies*, Princeton, NJ, 1957, 97-139.

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

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

[22] A.R. Karlin, S.J. Phillips and P. Raghavan. Markov Paging, *SIAM Journal on Computing*, **30**(3), 906-922, 2000.

[23] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes*, Academic Press, 1975.

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

[25] G. G. Langdon. A Note on the Ziv-Lempel Model for Compressing Individual Sequences, *IEEE Transactions on Information Theory*, **29**, 1983, 284-287.

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

[27] C. Lund, S. Phillips, and N. Reingold. Paging against a Distribution and IP Networking, *Journal of Computer and System Sciences*, **58**, 1999, 222-231.

[28] L. H. Loomis. On a Theorem of von Neumann. *Proceedings of the National Academy of Sciences of the USA*, bf 32, 1946, 213-215.

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

[30] N. Merhav and M. Feder. Universal Schemes for Sequential Decision from Individual Data Sequences, *IEEE Transactions on Information Theory*, **39**(4), 1993, 1280-1292.

[31] N. Merhev and M. Feder. Universal Prediction, *IEEE Transactions on Information Theory*, **44**, Oct. 1998, 2124-2147.

[32] N. Merhav, E. Ordentlich, G. Seroussi, and M. J. Weinberger. On Sequential Strategies for Loss Functions With Memory, *IEEE Transactions on Information Theory*, **48**(7), 1947-1958, 2002.

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

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

[35] G. Pandurangan and E. Upfal. Can Entropy Characterize Performance of Online Algorithms?, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001, 727-734.

[36] G. Pandurangan and W. Szpankowski. A Universal Online Caching Algorithm Based on Pattern Matching, in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2005.

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

[38] E. Suh and L. Rudolph. Adaptive Cache Partitioning, *CSG-Memo 432*, Lab. for Computer Science, MIT, June 2000.

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

[40] P. Krishnan and J.S. Vitter. Optimal Prediction for Prefetching in the Worst Case. *SIAM J. Comput.* **27**(6), 1998, 1617-1636.

[41] M. Weinberger and E. Ordentlich. On-line decision making for a class of loss functions via Lempel-Ziv Parsing, In *Proc. of the IEEE Data Compression Conference*, 2000, 163-172.

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