List Update with Locality of Reference: MTF Outperforms All Other Algorithms

Spyros Angelopoulos   Reza Dorrigiv   Alejandro López-Ortiz

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ont., N2L 3G1, Canada
{sangelop, rdorrigiv, alopez-o}@uwaterloo.ca

Technical Report CS-2006-46, School of Computer Science, University of Waterloo, November 2006

Abstract

It has been observed that in practice, typical request sequences for the list update problem exhibit a certain degree of locality of reference. We first extend the locality of reference model for the paging problem due to Albers et al [STOC 2002/JCSS 2005] to the domain of list update; this addresses the open question of defining an appropriate model for capturing locality of reference in the context of list update [Hester and Hirschberg ACM Comp. Surv. 1985]. We then apply this model in conjunction with a recent technique for comparing online algorithms, namely bijective analysis [SODA 2007] and analyze well known online algorithms for list update.

Using this framework, we prove that Move-to-Front (MTF) is the unique optimal algorithm for list update. This holds for both the standard cost function of Sleator and Tarjan [C. ACM 1985] and the refined cost function proposed independently by Martínez and Roura [TCS 2000] and Munro [ESA 2000]. Our work resolves an open conjecture of Martínez and Roura, namely proposing a measure which can successfully separate MTF from all other algorithms.

1 Introduction

List update is a fundamental problem in the context of on-line computation. Consider an unsorted list of l items. The input to the algorithm is a sequence of n requests that should be served in an on-line manner. Let $\mathcal{A}$ be an arbitrary on-line list update algorithm. To serve a request to an item x, $\mathcal{A}$ should linearly search the list until it finds x. If x is the i^{th} item in the list, $\mathcal{A}$ incurs cost i to access x. Immediately after accessing x, $\mathcal{A}$ can move x to any position closer to the front of the list at no extra cost. This is called a *free exchange*. Also $\mathcal{A}$ can exchange any two consecutive items at a cost of 1. These are called *paid exchanges*. An efficient algorithm should use free and paid exchanges so as to minimize the overall cost of serving a sequence. This is called the *standard cost model* [AW98].

The list update problem has been extensively studied due to its theoretical and practical importance. In practice, unsorted linear lists can be used to implement small dictionaries [BM85]. In addition, list update algorithms have been used as subroutines in computational geometry

[BCL93, Gol90] and data compression schemas [BSTW86, BW94, AM98, Sch98] among other applications. The problem was first studied by McCabe [McC65] in the context of maintaining a sequential file. In the early stages, list update algorithms were analyzed using the distributional or average-case model [Bit79, BK73, McC65, Riv76, GMS79]. In this model, it is assumed that the request sequence is generated by a probability distribution and the efficiency of an algorithm is related to the expected cost it incurs. A drawback of distributional analysis is that in practice the distribution of request sequences is usually not known in advance and it also changes during the execution of the algorithm.

Competitive analysis was a breakthrough in the analysis of on-line algorithms and made it possible to analyze on-line algorithms without any assumption about the distribution of the request sequences. The competitive ratio, first introduced formally by Sleator and Tarjan [ST85], has served as a practical measure for the study and classification of on-line algorithms. An algorithm is said to be α-competitive (assuming a cost-minimization problem) if the cost of serving any specific request sequence never exceeds α times the optimal cost (up to some additive constant) of an *off-line* algorithm which knows the entire request sequence. The competitive ratio, as a measure of comparison for on-line algorithms, has been applied to a variety of problems and settings, mainly due to its amenability to analysis: the measure is relatively simple to define yet powerful enough to quantify, to a large extent, the performance of an on-line algorithm.

List update algorithms were among the first algorithms studied using competitive analysis. Three well-known deterministic on-line algorithms are *Move-To-Front* (MTF), *Transpose*, and *Frequency-Count* (FC). MTF moves the requested item to the front of the list whereas Transpose exchanges the requested item with the item that immediately precedes it. FC maintains a frequency count for each item, updates this count after each access, and makes necessary moves so that the list always contains items in non-increasing order of frequency count. Sleator and Tarjan showed that MTF is 2-competitive, while Transpose and FC do not have constant competitive ratios [ST85]. Since then, several other deterministic and randomized on-line algorithms have been studied using competitive analysis. (See [Ira91, Alb98, AvW95, EY96] for some representative results.)

Notwithstanding its wide applicability, competitive analysis has some drawbacks. For certain problems, it gives unrealistically pessimistic performance ratios and fails to distinguish between algorithms that have vastly differing performance in practice. A well-known example is the paging problem. All three paging strategies least-recently-used (LRU), first-in-first-out (FIFO), and flush-when-full (FWF) have the same competitive ratio k, where k is the size of the cache in pages. In contrast, the performance ratio of LRU in practice is much better than k. For example, Sites and Agarwal provided experimental results showing that LRU achieves a performance ratio at most three [SA88]. Furthermore, it has long been empirically established that LRU (and variants thereof) are, in practice, preferable paging strategies to all other known paging algorithms [SGG02]. Such anomalies have lead to the introduction of many alternatives to competitive analysis of on-line algorithms (see [DLO05] for a comprehensive survey).

At first, it seems that competitive analysis of list update algorithms does not have these drawbacks and gives promising results: list update algorithms with better competitive ratio tend to have better performance in practice. For example, MTF behaves better than Transpose in practice [BEY98, BM85]. This is consistent with competitive analysis. However, the validity of the standard cost model has been debated. Martínez and Roura [MR00] and Munro [Mun00], independently addressed the drawbacks of the standard cost model. Let $(a_1, a_2, \ldots, a_l)$ be the list currently maintained by an algorithm $\mathcal{A}$. Martínez and Roura argued that in a realistic setting a complete rearrangement of all items in the list which precede item a_i to a given permutation would require time proportional to i, while $\mathcal{A}$ incurs a cost proportional to i^2 in the standard cost model to do this. Munro provided the example of accessing the last item of the list and then reversing the entire list. He observed that the real cost of this operation in an array or a linear link list should be $O(l)$, while it costs about $l^2/2$ in the standard cost model. As a consequence, their main objection to the standard model is that it prevents the algorithms from using their true power. They instead proposed a new cost model in which the cost of accessing the i^{th} item of the list plus the cost of reorganizing the first i items is linear in i. We will refer to this model as the *modified cost model*. They showed that under this modified cost model, every on-line algorithm has amortized cost of $\Theta(l)$ per access for some arbitrary long sequences, while there are some off-line algorithms with amortized cost of $\Theta(\log l)$ on every sequence. Therefore no on-line list update algorithm has constant competitive ratio in the modified cost model; more importantly, we cannot separate the behaviour of any on-line algorithms. Martínez and Roura observed this and posed the question: "an important open question is whether there exist alternative ways to define competitiveness such that MTF and other good online algorithms for the list update problem would be competitive, even for the [modified] cost model".

This situation is reminiscent of the basic objection to competitive analysis: It uses the concept of an optimal off-line algorithm, OPT, as a baseline in order to compare on-line algorithms. While this may be convenient, it is rather indirect: one could argue that in comparing two online algorithms $\mathcal{A}$ and $\mathcal{B}$ all the information we should need is the cost incurred by the algorithms on each request sequence. In various problems OPT is too powerful compared to on-line algorithms, causing all on-line algorithms to behave equally badly according to competitive analysis. Some alternative measures overcome this problem by allowing direct comparison of on-line algorithms. *Max-Max Ratio* [BDB94], *Relative Worst Order Ratio* [BF03], Bijective Analysis and Average Analysis [ADLO07] are examples of such measures which have been applied mostly to the paging problem as well as some other on-line problems. We are not aware of any result in the literature that applies the above measures to on-line list update algorithms.

Another issue in the analysis of on-line algorithms is that "real-life" sequences usually exhibit *locality of reference*. Informally, this property suggests that the currently requested item is likely to be requested again in the near future. For the paging problem, several models for capturing locality of reference have been proposed [Tor98, AFG05, Bec04]. Likewise, many researchers have pointed out that input sequences of list update algorithms in practice show locality of reference [HH85, Sch98, BEY98] and actually on-line list update algorithms try to take advantage of this property [HH85, RWS94]. Hester and Hirschberg [HH85] posed the question of providing a good definition of locality of accesses for the list update problem as an open problem. However, to the best of our knowledge, locality of reference for list update algorithms has not been formally studied. In addition, it has been commonly assumed, based on intuition and experimental evidence, that MTF is the best algorithm on sequences with high locality of reference, e.g., Hester and Hirschberg [HH85] claim: "move-to-front performs best when the list has a high degree of locality".

For the sake of simplicity, in this paper we only consider the *static* list update problem. This means that we only have accesses to list items and do not have any insert or delete operations. In particular, we have a set $S = \{a_1, a_2, \ldots, a_l\}$ of l items initially organized as a list $\mathcal{L}_0 = (a_1, a_2, \ldots, a_l)$. The results in this paper can easily be extended to the dynamic version of the problem. For an on-line algorithm $\mathcal{A}$ and a sequence σ, we denote by $\mathcal{A}(\sigma)$ the cost that $\mathcal{A}$ incurs to serve σ. We also denote by $\mathcal{I}_n$ the set of all request sequences of length n, and by $\mathcal{I}_{k+1}(\sigma)$ where σ is of length k, the set of the l sequences in $\mathcal{I}_{k+1}$ which have σ as their prefix.

Our results We begin by showing that all on-line list update algorithms are equivalent according to Bijective Analysis under the modified cost model. We then extend a model for locality of reference, proposed by Albers et al. [AFG05] in the context of the paging problem to the list update problem. The validity of the extended model is supported by experimental results obtained on the Calgary Corpus, which is frequently used as a standard benchmark for evaluating the performance of compression algorithms (and by extension list update algorithms, e.g. [BEY97]). Thus, we resolve the open problem posed by Hester and Hirschberg [HH85]. Our main result proves that under both the standard and the modified cost function MTF is never outperformed in our model, while it always outperforms *any other on-line list update algorithm*. As mentioned earlier, Martínez and Roura [MR00] posed the open problem of finding an alternative measure that shows the superiority of MTF in the modified cost model and suggested that this can be done by adding some restrictions over the sequences of requests. Our analysis technique allows us to resolve this problem as well.

2 Bijective Analysis

In this section, we first provide the formal definitions of Bijective Analysis and Average Analysis and then we show equivalence of all list update algorithms under the modified model according to these measures. We choose to employ these measures since they reflect certain desired characteristics for comparing online algorithms: they allow for direct comparison of two algorithms without appealing to the concept of the "optimal" cost (see [ADLO07]for a more detailed discussion). In addition, these measures do not evaluate the performance of the algorithm on a single "worst-case" request, but instead use the cost that the algorithm incurs on each and all request sequences. These two measures have already been successfully applied in the context of the paging problem [ADLO07] and have led to separation results for various paging algorithms.

Informally, Bijective Analysis aims to pair input sequences for two algorithms $\mathcal{A}$ and $\mathcal{B}$ using a bijection in such a way that the cost of $\mathcal{A}$ on input σ is no more than the cost of $\mathcal{B}$ on the image of σ, for all request sequences σ of the same length. In this case, intuitively, $\mathcal{A}$ is no worse than $\mathcal{B}$. On the other hand, Average Analysis compares the average cost of the two algorithms over all request sequences of the same length.

Definition 1 *[ADLO07] We say that an on-line algorithm $\mathcal{A}$ is* no worse *than an on-line algorithm $\mathcal{B}$ according to Bijective Analysis if there exists an integer $n_0 \geq 1$ so that for each $n \geq n_0$, there is a bijection $b : \mathcal{I}_n \leftrightarrow \mathcal{I}_n$ satisfying $\mathcal{A}(\sigma) \leq \mathcal{B}(b(\sigma))$ for each $\sigma \in \mathcal{I}_n$. We denote this by $\mathcal{A} \preceq_b \mathcal{B}$. Otherwise we denote the situation by $\mathcal{A} \not\preceq_b \mathcal{B}$. Similarly, we say that $\mathcal{A}$ and $\mathcal{B}$ are* the same *according to Bijective Analysis if $\mathcal{A} \preceq_b \mathcal{B}$ and $B \preceq_b A$. This is denoted by $\mathcal{A} \equiv_b \mathcal{B}$. Lastly we say $\mathcal{A}$ is* better *than $\mathcal{B}$ according to Bijective Analysis if $\mathcal{A} \preceq_b \mathcal{B}$ and $\mathcal{B} \not\preceq_b \mathcal{A}$. We denote this by $\mathcal{A} \prec_b \mathcal{B}$.*

Definition 2 *[ADLO07] We say that an on-line algorithm $\mathcal{A}$ is* no worse *than an on-line algorithm $\mathcal{B}$ according to Average Analysis if there exists an integer $n_0 \geq 1$ so that for each $n \geq n_0$, $\sum_{I \in \mathcal{I}_n} \mathcal{A}(I) \leq \sum_{I \in \mathcal{I}_n} \mathcal{B}(I)$. We denote this by $\mathcal{A} \preceq_a \mathcal{B}$. Otherwise we denote the situation by $\mathcal{A} \not\preceq_a \mathcal{B}$. $\mathcal{A} \equiv_a \mathcal{B}$, and $\mathcal{A} \prec_a \mathcal{B}$ are defined as for Bijective Analysis.*

Observation 1 *[ADLO07] If $\mathcal{A} \not\preceq_a \mathcal{B}$, then $\mathcal{A} \not\preceq_b \mathcal{B}$. In addition, if $\mathcal{A} \preceq_b \mathcal{B}$, then $\mathcal{A} \preceq_a \mathcal{B}$ and similar statements hold for $\mathcal{A} \equiv_b \mathcal{B}$ and $\mathcal{A} \prec_b \mathcal{B}$.*

The following theorem proves that under the modified cost model all list update algorithms are equivalent. This result parallels the equivalence of all lazy paging algorithms under Bijective Analysis as shown in [ADLO07].

Theorem 1 *Let $\mathcal{A}$ and $\mathcal{B}$ be two arbitrary on-line list update algorithms. Under the modified cost model, we have $\mathcal{A} \equiv_b \mathcal{B}$.*

Proof: We prove that for every $n \geq 1$ there is a bijection $b^n : \mathcal{I}_n \leftrightarrow \mathcal{I}_n$ so that $\mathcal{A}(\sigma) \leq \mathcal{B}(b^n(\sigma))$ for each $\sigma \in I_n$. We show this by induction on n, the length of sequences. Since $\mathcal{A}$ and $\mathcal{B}$ start with the same initial list, they incur the same cost on each sequence of length 1. Therefore the statement trivially holds for $n = 1$. Assume that it is true for $n = k$. Thus there is a bijection $b^k : \mathcal{I}_k \leftrightarrow \mathcal{I}_k$ so that $\mathcal{A}(\sigma) \leq \mathcal{B}(b^k(\sigma))$ for each $\sigma \in I_k$. Let σ be an arbitrary sequence of length k and $\sigma' = b^k(\sigma)$. We map $\mathcal{I}_{k+1}(\sigma)$ to $\mathcal{I}_{k+1}(\sigma')$ as follows. Let $\mathcal{L}(\mathcal{A}, \sigma) = (a_1, a_2, \ldots, a_l)$ be the list maintained by $\mathcal{A}$ after serving σ and $\mathcal{L}(\mathcal{B}, \sigma') = (b_1, b_2, \ldots, b_l)$ be the list maintained by $\mathcal{B}$ after serving σ'. Consider an arbitrary sequence $\sigma_1 \in \mathcal{I}_{k+1}(\sigma)$ and let its last request is to item a_i. We map σ_1 to the sequence $\sigma_2 \in \mathcal{I}_{k+1}(\sigma')$ that has b_i as its last request. Since $\mathcal{A}(\sigma) \leq \mathcal{B}(\sigma')$ and $\mathcal{A}$'s cost on the last request of σ_1 is the same as $\mathcal{B}$'s cost on the last request of σ_2, we have $\mathcal{A}(\sigma_1) \leq \mathcal{B}(\sigma_2)$. Therefore we get the desired mapping from $\mathcal{I}_{k+1}(\sigma)$ to $\mathcal{I}_{k+1}(\sigma')$. We obtain a bijection $b^{k+1} : \mathcal{I}_{k+1} \leftrightarrow \mathcal{I}_{k+1}$ by considering the above mapping for each sequence $\sigma \in \mathcal{I}_k$. Thus our induction statement is true and we have $\mathcal{A} \preceq_b \mathcal{B}$. Using a similar argument, we can show $\mathcal{B} \preceq_b \mathcal{A}$. Therefore we have $\mathcal{A} \equiv_b \mathcal{B}$. □

We will call a list update algorithm *economical* if it does not use paid exchanges. Since an economical list update algorithm does not incur any cost for reorganizing the list we can prove the following statement using an argument analogous to the proof of Theorem 1.

Corollary 1 *All economical on-line list update algorithms are equivalent according to Bijective Analysis under the standard cost model.*

These results show that so long as we consider all possible request sequences, all on-line list update algorithms are equivalent in a strong sense. However, as stated earlier, in practice request sequences tend to exhibit locality of reference. Therefore, the algorithm can focus on input sequences with this property. In the next section we show that we can use such an assumption to prove the superiority of MTF.

3 List Update with Locality of Reference

As stated in the Introduction, several models have been proposed for paging with locality of reference [Tor98, AFG05, Bec04]. In this paper, we consider the model of Albers et al. [AFG05], in which a request sequence has high locality of reference if the number of distinct requests in a window of size n is small. In Section 4 we will present experimental evidence which supports the validity of this model for the list update problem. Consider a function that represents the maximum number of distinct items in a window of size n, in a request sequence. For the paging problem, extensive experiments with real data show that this function can be bounded by a concave function for most practical request sequences [AFG05]. Let f be an increasing concave function. We say that a request sequence is *consistent* with f if the number of distinct requests in any window of size n is at most $f(n)$, for any $n \in \mathcal{N}$. We can model locality by considering only those request sequences that are consistent with f. Albers et al. consider a slightly more restrictive class of functions called concave* functions.

Definition 3 *[AFG05] A function $f : N \rightarrow R_+$ is concave* if*

1. $f(1) = 1$ *and*

2. $\forall n \in \mathcal{N} : f(n+1) - f(n) \geq f(n+2) - f(n+1)$.

We additionally require that f be surjective on the integers between 1 and its maximum value.

In order to model locality, we restrict the request sequences to those consistent with a concave* function f. Let $\mathcal{I}^f$ denote the set of such sequences. We can easily modify the definitions of Bijective Analysis and Average Analysis (Definition 1 and Definition 2) by replacing $\mathcal{I}$ with $\mathcal{I}^f$ throughout. We denote the corresponding relations by $\mathcal{A} \preceq_b^f \mathcal{B}$, $\mathcal{A} \preceq_a^f \mathcal{B}$, etc. Figure 1 shows the partition of the input space induced by the choice of f. Observe that the performance of list update algorithms are now evaluated within the subset of request sequences of a given length whose locality of reference is consistent with f, i.e. $\mathcal{I}_n^f$.

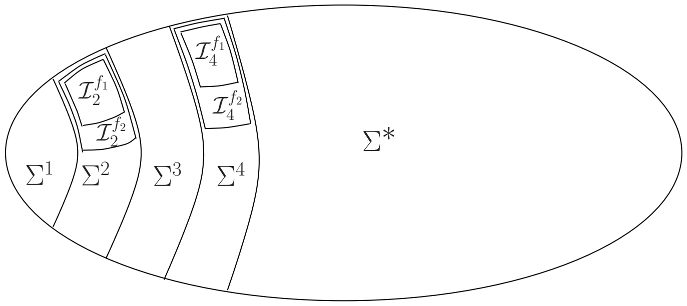


Figure 1: Partition of the input space induced by different choices of f [ADLO07].

Note that the inductive argument used to prove that all on-line list update algorithms are equivalent according to Bijective Analysis (Theorem 1) does not necessarily carry through under concave analysis because the bijection of the proof may map a sequence in $\mathcal{I}^f$ to one not in $\mathcal{I}^f$. Consider a fixed concave* function f. Let $\mathcal{I}_n^f$ denote sequences of length n in $\mathcal{I}^f$.

Definition 4 *Let $\mathcal{A}$ and $\mathcal{B}$ be list update algorithms, and f be a concave* function. $\mathcal{A}$ is said to* (m,f)-dominate *$\mathcal{B}$ for some integer m, if we have*

$$\sum_{\sigma \in \mathcal{I}_m^f} \mathcal{A}(\sigma) \leq \sum_{\sigma \in \mathcal{I}_m^f} \mathcal{B}(\sigma).$$

$\mathcal{A}$ is said to dominate *$\mathcal{B}$ if there exists an integer $m_0 \geq 1$ so that for each $m \geq m_0$ and every concave* function f, $\mathcal{A}$ (m,f)-dominates $\mathcal{B}$.*

Observation 2 *$\mathcal{A} \preceq_a^f \mathcal{B}$ if and only if there exists an integer $m_0 \geq 1$ so that $\mathcal{A}$ (m,f)-dominates $\mathcal{B}$ for each $m \geq m_0$.*

Lemma 1 *For every on-line list update algorithm $\mathcal{A}$, MTF dominates $\mathcal{A}$.*

Proof: Let f be an arbitrary concave* function and m be a positive integer. For any $1 \leq i \leq m$, let $\mathcal{F}_{i,m}(\mathcal{A})$ be the total cost $\mathcal{A}$ incurs on the i^{th} request of all sequences in $\mathcal{I}_m^f$. We will first show that $\mathcal{F}_{i,m}(MTF) \leq \mathcal{F}_{i,m}(\mathcal{A})$ for any $1 \leq i \leq m$. For $i = 1$, we have $\mathcal{F}_{1,m}(MTF) = \mathcal{F}_{1,m}(\mathcal{A})$, as all algorithms start with the same list. Now suppose that $i > 1$. Let σ be an arbitrary sequence of length $i-1$, T_σ denote the set of all sequences in $\mathcal{I}_m^f$ that have σ as their prefix, and $\mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma)$ be the total cost $\mathcal{A}$ incurs on the i^{th} request of all sequences in T_σ. Denote by $\mathcal{L}(MTF, \sigma) = (a_1, a_2, \ldots, a_l)$ and $\mathcal{L}(\mathcal{A}, \sigma) = (b_1, b_2, \ldots, b_l)$ the lists maintained by MTF and $\mathcal{A}$

after serving σ, respectively. Suppose that c_j (resp., d_j) sequences in T_σ have a_j (resp., b_j) as their i^{th} request, for $1 \le j \le l$. Note that $\sum_{1\le j\le l} c_j = \sum_{1\le j\le l} d_j = |T_\sigma|$ and $(d_1, d_2, \ldots, d_l)$ is a permutation of $(c_1, c_2, \ldots, c_l)$.

We first show that $c_{j+1} \le c_j$ for $1 \le j < l$. Let C_j and C_{j+1} denote the set of sequences in T_σ that have a_j and a_{j+1} as their i^{th} request. We provide a one-to-one mapping from C_{j+1} to C_j which proves that $|C_{j+1}| \le |C_j|$. We map every sequence τ in C_{j+1} to a sequence τ' in C_j by replacing every a_j with a_{j+1} and every a_{j+1} by a_j, starting from position i. Since a_j occurs before a_{j+1} in MTF's list after serving σ, we know that the last request to a_j occurs after the last request to a_{j+1} in σ. Therefore if τ is consistent with f, so is τ'. Thus every sequence in C_{j+1} is mapped to a unique sequence in C_j and we have $c_{j+1} = |C_{j+1}| \le |C_j| = c_j$.

Therefore we have

$$\mathcal{F}_{i,m}(MTF \,|\, \sigma) = \sum_{1\le j\le l} j \times c_j \le \sum_{1\le j\le l} j \times d_j = \mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma).$$

Now since

$$\mathcal{F}_{i,m}(MTF) = \sum_{\sigma\in\mathcal{I}_{i-1}} \mathcal{F}_{i,m}(MTF \,|\, \sigma) \text{ and } \mathcal{F}_{i,m}(\mathcal{A}) = \sum_{\sigma\in\mathcal{I}_{i-1}} \mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma),$$

we get $\mathcal{F}_{i,m}(MTF) \le F_{i,m}(\mathcal{A})$. We have

$$\sum_{\sigma\in\mathcal{I}_m^f} MTF(\sigma) = \sum_{1\le i\le m} \mathcal{F}_{i,m}(MTF),$$

and

$$\sum_{\sigma\in\mathcal{I}_m^f} \mathcal{A}(\sigma) = \sum_{1\le i\le m} \mathcal{F}_{i,m}(\mathcal{A}).$$

Therefore

$$\sum_{\sigma\in\mathcal{I}_m^f} MTF(\sigma) \le \sum_{\sigma\in\mathcal{I}_m^f} \mathcal{A}(\sigma).$$

Thus MTF (m, f)-dominates $\mathcal{A}$ for every concave* function f, and every integer $m \ge 1$. Hence MTF dominates $\mathcal{A}$. □

Corollary 2 *For any concave* function f and any on-line list update algorithm $\mathcal{A}$,*

$$MTF \preceq_a^f \mathcal{A}.$$

Therefore MTF is an optimal algorithm according to Average Analysis, when we classify the input sequences by locality of reference. A natural question is whether MTF is a unique optimum or not, i.e., is there an on-line list update algorithm $\mathcal{A}$ that dominates MTF?

Lemma 2 *No on-line list update algorithm (other than MTF itself) dominates MTF.*

Proof: Assume by way of contradiction that an on-line list update algorithm $\mathcal{A}$ dominates MTF and that $\mathcal{A}$ is different from MTF. According to the definition, there exists an integer $m_0 \ge 1$ so that for each $m \ge m_0$ and every concave* function f, $\mathcal{A}$ (m, f)-dominates MTF, i.e.,

$$\sum_{\sigma\in\mathcal{I}_m^f} \mathcal{A}(\sigma) \le \sum_{\sigma\in\mathcal{I}_m^f} MTF(\sigma).$$

Following the proof of Lemma 1, this holds only if $\mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma) = \mathcal{F}_{i,m}(MTF \,|\, \sigma)$ for every $m \geq m_0$, $2 \leq i \leq m$, and every sequence σ of length $i-1$. Let $\sigma \in \mathcal{I}^f_{i-1}$ be a sequence so that $\mathcal{L}(\mathcal{A}, \sigma)$ is different from $\mathcal{L}(MTF, \sigma)$, k be the largest index so that $y = a_k \neq b_k = x$ (for a_k and b_k defined as in Lemma 1, and p be the smallest index so that $\sigma[p..i-1]$ contains at most $k-1$ distinct items. Select the concave* function f so that $f(i-p) = f(i-p+1) = k-1$. Since $y \in \sigma[p..i-1]$ and $x \notin \sigma[p..i-1]$, we have $c_k = 0 < d_k$ (the sequence of length $m > i$ obtained by repeating y in any position starting from i^{th} position is consistent with f). Therefore

$$\mathcal{F}_{i,m}(MTF \,|\, \sigma) = \sum_{1 \leq j \leq l} j \times c_j < \sum_{1 \leq j \leq l} j \times d_j = \mathcal{F}_{i,m}(\mathcal{A} \,|\, \sigma),$$

which is a contradiction. □

Theorem 2 *Let $\mathcal{A}$ be an on-line list update algorithm other than MTF. Then $MTF \preceq^f_b \mathcal{A}$ and there exists at least one concave* function f so that*

$$\mathcal{A} \not\preceq^f_a MTF, \qquad \textit{which implies} \qquad \mathcal{A} \not\preceq^f_b MTF.$$

We can prove the separation with respect to Bijective Analysis between MTF and specific algorithms, e.g., Transpose, for a much larger family of concave* functions.

Theorem 3 *For all concave* functions f such that $f(l) < l$ (l is the size of list),*

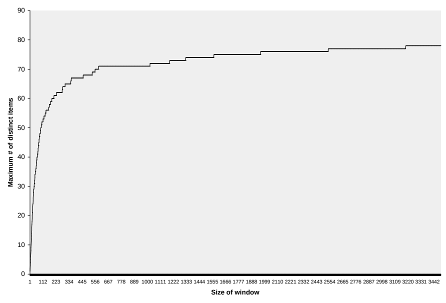
$$\textit{Transpose} \not\preceq^f_b MTF.$$

Proof: Let $\mathcal{L}_0 = (a_1, a_2, \ldots, a_l)$ be the initial list. Assume by way of contradiction that $Transpose \preceq^f_b MTF$. Therefore there is an integer $n_0 \geq 1$ so that for each $n \geq n_0$, there is a bijection $b : \mathcal{I}^f_n \leftrightarrow \mathcal{I}^f_n$ satisfying $Transpose(\sigma) \leq MTF(b(\sigma))$ for each $\sigma \in \mathcal{I}^f_n$. Now consider a sequence σ of length $m \geq n_0$ obtained by considering the prefix of the infinite sequence $a_l a_{l-1} a_l a_{l-1} \ldots$. Transpose incurs a cost of l on each request and we have $Transpose(\sigma) = m \times l$. Note that σ is consistent with f, because it has two distinct items.[1] Thus $\sigma \in \mathcal{I}^f_m$ and from the assumption there should exist some sequence $\sigma' \in \mathcal{I}^f_m$ so that $m \times l = Transpose(\sigma) \leq MTF(\sigma')$. Therefore MTF should incur a cost of l on each request of σ'. Hence σ' should be a prefix of the sequence $a_l a_{l-1} a_{l-2} \ldots a_1 a_l a_{l-1} a_{l-2} \ldots a_1 \ldots$. Now any window of size l in σ' has l distinct items. Since we started with $f(l) < l$, σ' is not consistent with f and this contradicts the assumption that $\sigma' \in \mathcal{I}^f_m$. □

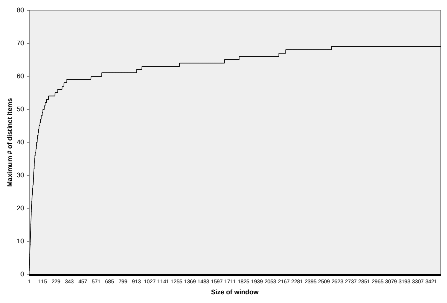
4 Experimental Results and Analysis

In this section we test the validity of the locality of reference assumption we described in Section 3 against experimental data. For our experiments, we considered the fourteen files of the Calgary Compression Corpus [WB] which are frequently used as a standard benchmark for file compression. Recall that list update algorithms can be used in a very direct way in file compression. For each file, we computed the maximum number of characters in windows of all possible sizes, up to the

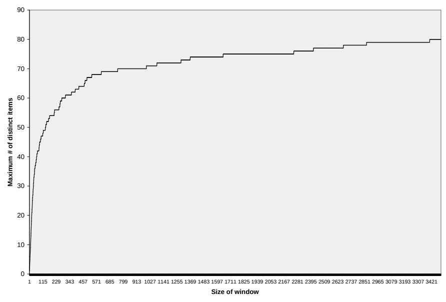
[1]We can assume that $f(2) = 2$ because otherwise we are restricted to sequences that contain only one item.



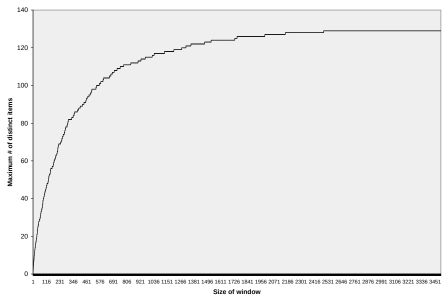
(a) Results for file bib



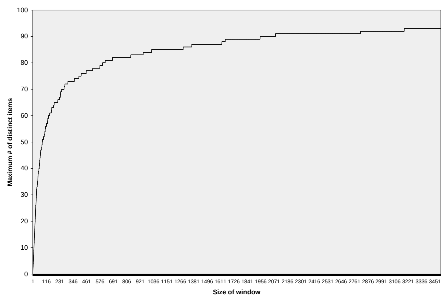
(b) Results for file book1



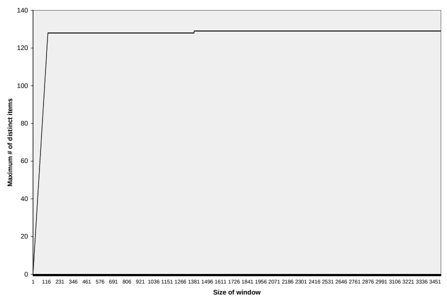
(c) Results for file book2



(d) Results for file geo



(e) Results for file news

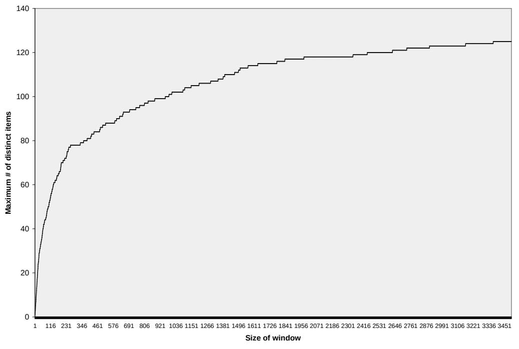


(f) Results for file obj1

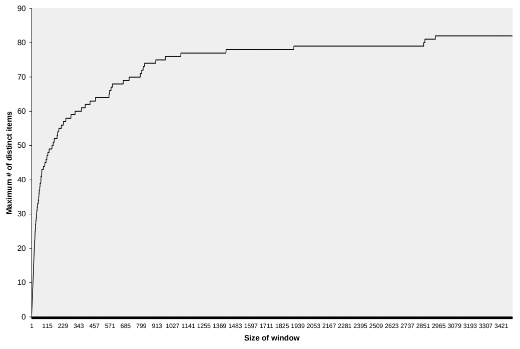
Figure 2: Maximum number of distinct characters in windows of size up to 3500 for the files in Calgary Compression Corpus.

size of the whole file. Figures 2 and 3 show the resulting graphs. Note that since we observed that the maximum number of distinct items does not change much as we increase the size of window to values more than 3500, we only show the results for windows of size up to 3500.

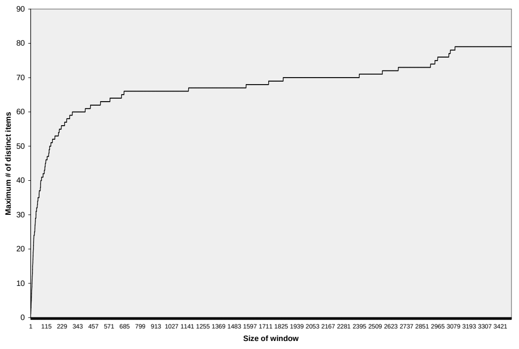
As can be seen from these graphs, the curves have an overall concave shape. We should note that for some of the input files, the function we obtained is not concave for some intervals. However, this is not a major concern, since we can bound said function by any concave function f which is such that $f(i)$ is an upper bound on the maximum number of distinct items in windows of size i. For instance, we can take the upper convex hull of the data points. In fact, Albers et al. [AFG05] observed that similar non-concavity (mostly localized within small intervals) was present in their experimental results concerning locality of reference in typical request sequences for the paging problem. Albers et al. put forth this argument to justify the fact that local small deviations from concavity do not impose a serious problem.



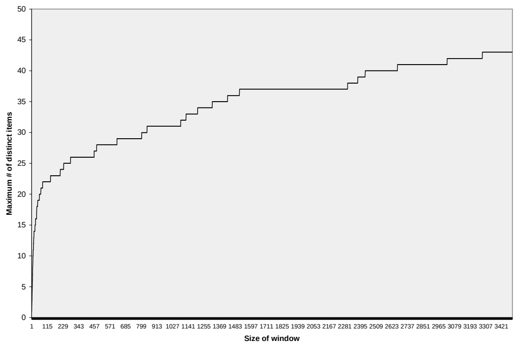
(a) Results for file obj2



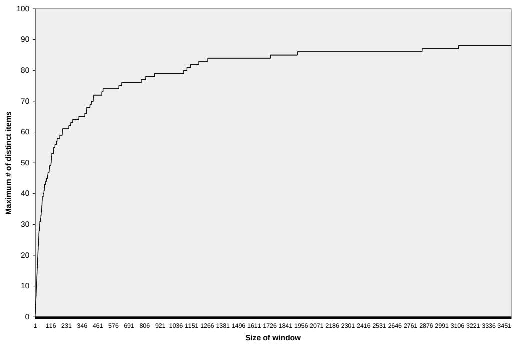
(b) Results for file paper1



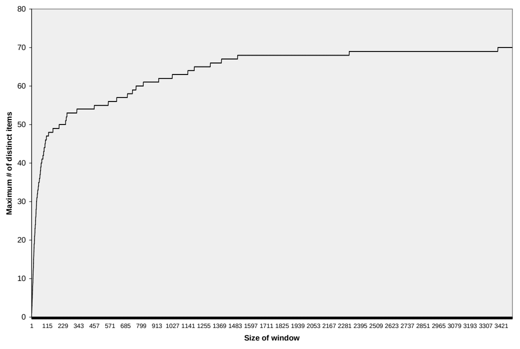
(c) Results for file paper2



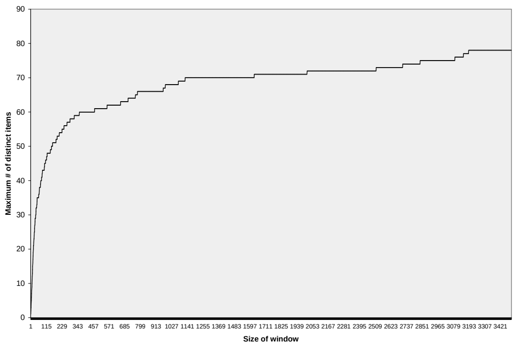
(d) Results for file pic



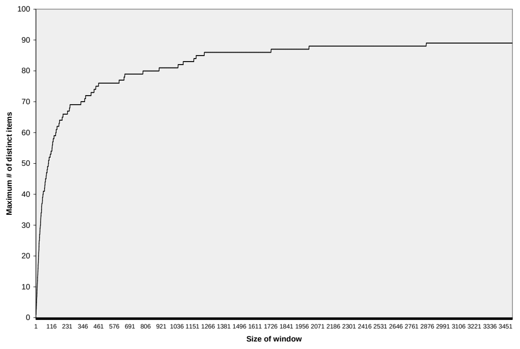
(e) Results for file progc



(f) Results for file progl



(g) Results for file progp



(h) Results for file trans

Figure 3: Calgary Compression Corpus (continued).

5 Conclusions

In this paper we addressed certain open questions concerning the well-studied list update problem. We first considered the issue of modeling locality of reference for typical request sequences for this problem. We provided experimental evidence which suggests that the concave-function model of Albers et al., originally devised for the context of paging algorithms, can satisfactorily model locality of reference within the domain of list update. We then combined this model with two recently proposed measures for comparing online algorithms, namely Bijective Analysis and Average Analysis. Our choice was based on the fact that these measures allow direct comparison of two online algorithms, by considering their relative performance on all requests sequences of the same length, rather than on some specific pathological sequences. These measures have been previously applied with success in separating several paging algorithms, a situation which has long been known but cannot be resolved by resorting solely to competitive analysis.

Using the above framework, we showed that whereas all list update algorithms are equivalent for the modified-cost model, when locality of reference is considered, MTF emerges as the sole best-possible online algorithm for the problem. This resolves an open problem posed by Martínez and Roura. We believe that our techniques might well be applicable to other problems in which competitive analysis has failed to yield satisfactory results such as the online bin packing problem, but this remains the subject of future work.

References

[ADLO07] S. Angelopoulos, R. Dorrigiv, and A. López-Ortiz. On the separation and equivalence of paging strategies. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*, 2007. to appear.

[AFG05] S. Albers, L. M. Favrholdt, and O. Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70(2):145–175, 2005.

[Alb98] S. Albers. Improved randomized on-line algorithms for the list update problem. *SIAM Journal on Computing*, 27(3):682–693, June 1998.

[AM98] S. Albers and M. Mitzenmacher. Average case analyses of list update algorithms, with applications to data compression. *Algorithmica*, 21(3):312–329, 1998.

[AvW95] S. Albers, B. von Stengel, and R. Werchner. A combined BIT and TIMESTAMP algorithm for the list update problem. *Information Processing Letters*, 56:135–139, 1995.

[AW98] S. Albers and J. Westbrook. Self-organizing data structures. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms — The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 13–51. Springer-Verlag, 1998.

[BCL93] J. L. Bentley, K. L. Clarkson, and D. B. Levine. Fast linear expected-time algorithms for computing maxima and convex hulls. *Algorithmica*, 9(2):168–183, 1993.

[BDB94] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11:73–91, 1994.

[Bec04] L. Becchetti. Modeling locality: A probabilistic analysis of LRU and FWF. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA '04)*, volume 3221 of *LNCS*, pages 98–109, 2004.

[BEY97] R. Bachrach and R. El-Yaniv. Online list accessing algorithms and their applications: Recent empirical evidence. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, pages 53–62, 1997.

[BEY98] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[BF03] J. Boyar and L. M. Favrholdt. The relative worst order ratio for on-line algorithms. In *CIAC: Italian Conference on Algorithms and Complexity*, 2003.

[Bit79] J.R. Bitner. Heuristics that dynamically organize data structures. *SIAM Journal on Computing*, 8:82–110, 1979.

[BK73] P.J. Burville and J.F.C. Kingman. On a model for storage and search. *Journal of Applied Probability*, 10:697–701, 1973.

[BM85] J.L. Bentley and C.C. McGeoch. Amortized analyses of self-organizing sequential search heuristics. *Communications of the ACM*, 28:404–411, 1985.

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

[BW94] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, DEC SRC, 1994.

[DLO05] R. Dorrigiv and A. López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 36(3):67–81, September 2005.

[EY96] R. El-Yaniv. There are infinitely many competitive-optimal online list accessing algorithms. Manuscript, 1996.

[GMS79] G. H. Gonnet, J. I. Munro, and H. Suwanda. Towards self-organizing linear search. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science (FOCS '79)*, pages 169–174. IEEE, 1979.

[Gol90] M. J. Golin. *Probabilistic analysis of geometric algorithms*. PhD thesis, Princeton University, Princeton, NJ, USA, 1990. Available as Computer Science Department Technical Report CS-TR-266-90.

[HH85] J. H. Hester and D. S. Hirschberg. Self-organizing linear search. *ACM Computing Surveys*, 17(3):295, September 1985.

[Ira91] S. Irani. Two results on the list update problem. *Information Processing Letters*, 38:301–306, 1991.

[McC65] J. McCabe. On serial files with relocatable records. *Operations Research*, 12:609–618, 1965.

[MR00] C. Martínez and S. Roura. On the competitiveness of the move-to-front rule. *Theoretical Computer Science*, 242(1–2):313–325, July 2000.

[Mun00] J. I. Munro. On the competitiveness of linear search. In *Proceedings of the 8th Annual European Symposium on Algorithms (ESA '00)*, volume 1879 of *Lecture Notes in Computer Science*, pages 338–345, 2000.

[Riv76] R. Rivest. On self-organizing sequential search heuristics. *Communications of the ACM*, 19:63–67, 1976.

[RWS94] N. Reingold, J. Westbrook, and D.D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11:15–32, 1994.

[SA88] R. L. Sites and A. Agarwal. Multiprocessor cache analysis using ATUM. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*, pages 186–195, 1988.

[Sch98] F. Schulz. Two new families of list update algorithms. In *Proceedings of the 9th International Symposium on Algorithms and Computation (ISAAC '98)*, volume 1533 of *Lecture Notes in Computer Science*, pages 99–108, 1998.

[SGG02] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. John Wiley & Sons, 2002.

[ST85] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[Tor98] E. Torng. A unified analysis of paging and caching. *Algorithmica*, 20(2):175–200, 1998.

[WB] I. H. Witten and T. Bell. The Calgary/Canterbury text compression corpus. Anonymous ftp from ftp.cpsc.ucalgary.ca /pub/text.compression/corpus/text.compression.corpus.tar.Z.