A New Family of Randomized Algorithms for List Accessing

Theodoulos Garefalakis

Department of Computer Science
University of Toronto
Toronto, Canada
M5S 1A4
e-mail: theo@cs.toronto.edu

Abstract. Sequential lists are a frequently used data structure for implementing dictionaries. Recently, self-organizing sequential lists have been proposed for "engines" in efficient data compression algorithms. In this paper, we investigate the problem of *list accessing* from the perspective of *competitive analysis*. We establish a connection between randomized list accessing algorithms and Markov chains, and present *Markov-Move-To-Front*, a family of randomized algorithms. To every finite, irreducible Markov chain corresponds a member of the family. The family includes as members well known algorithms such as *Move-To-Front, Random-Move-To-Front, Counter,* and *Random-Reset*.
First we analyze *Markov-Move-To-Front* in the standard model, and present upper and lower bounds that depend only on two parameters of the underlying Markov chain. Then we apply the bounds to particular members of the family. The bounds that we get are at least as good as the known bounds. Furthermore, for some algorithms we obtain bounds that, to our knowledge, are new.
We also analyze *Markov-Move-To-Front* in the paid exchange model. In this model, the cost of an element transposition is always paid, and costs d. We prove upper and lower bounds that are relatively tight. Again, we apply the bounds to known algorithms such as *Random-Move-To-Front* and *Counter*. In both cases, the upper and lower bounds match as the parameter d tends to infinity.

1 Introduction

In this paper we consider the *static list accessing* (also known as the *list update*) problem. The problem is to maintain an unsorted list of items in such a way that the cost of successive accesses is kept small. More specifically, an initial list of items is given, and a request sequence is generated. A request specifies an item in the list, and is serviced by accessing the item. In the on-line setting, each request has to be serviced before the next request is made known. The cost incurred by an access to an item is equal to the position of the element in the current list (as maintained by the on-line algorithm). In order to reduce the cost of future requests, the algorithm is allowed to reorganize the list between

requests. One can view this reorganization as a sequence of exchanges between consecutive items in the list. The cost of such a rearrangement is measured in terms of the minimum number of exchanges needed for the rearrangement. We will consider two different cost models for the exchanges that are performed in order to rearrange the list. In the standard model, the algorithm may move the item just accessed to a position closer to the front of the list *free of charge*. Those exchanges are called *free*. All other exchanges are called *paid*, and cost 1 each. In the paid exchange model every exchange has a cost. One can justify the absence of free transposition, if one thinks of the list as an unsorted array. In order to move an item in the list, we have to perform a series of transpositions. The constant d is typically greater than 1, and reflects the fact that link traversals and exchanges are different operations, and therefore may have a different cost.

1.1 Competitive Analysis

For the analysis of on-line algorithms, we use *competitive analysis* [9]. In this framework, the on-line algorithm is compared with an optimal off-line algorithm. An optimal off-line algorithm knows the entire request sequence in advance, and can make optimal choices incurring minimum cost. The performance measure of the on-line algorithm is the *competitive ratio*.

Definition 1. A deterministic on-line algorithm ALG is said to be c-competitive, if there exists a constant α such that for every request sequence σ,

$$ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha.$$

where OPT(σ) is the cost incurred by an optimal off-line algorithm with full knowledge of the request sequence σ. The competitive ratio of ALG, denoted by $R(ALG)$ is the infimum of c, such that ALG is c-competitive.

A useful way to view the problem of analyzing an on-line algorithm is as a game between an *on-line player* and a cruel *adversary* [3]. The on-line player runs the on-line algorithm that services the request sequence created by the adversary. The adversary, on the other hand, based on knowledge of the algorithm used by the on-line player, constructs the input that maximizes the competitive ratio. Usually, we identify the adversary and the off-line algorithm, and think of it as the *off-line player*. For randomized algorithms, the knowledge on which the adversary bases the construction of the cruel sequence is important. In this work we consider only *oblivious* adversaries, which are allowed to examine the on-line algorithm, but have to generate the entire request sequence in advance, without any knowledge of the random bits of the on-line player. The adversary services the request sequence off-line, incurring the optimal off-line cost.

Definition 2. A randomized on-line algorithm ALG is said to be c-competitive against an oblivious adversary, if there exists a constant α such that for every request sequence σ generated as described above,

$$E[ALG(\sigma)] \leq c \cdot OPT(\sigma) + \alpha$$

where OPT(σ) is the cost incurred by an optimal off-line algorithm on σ, and the expectation is taken over the random choices made by ALG. The infimum of c, such that ALG is c-competitive, is called the competitive ratio of ALG against an oblivious adversary, and is denoted by $R_{OBL}(ALG)$.

1.2 Competitive Algorithms

Sleator and Tarjan [9] have shown that the well-known deterministic *Move-To-Front* algorithm (MTF) is 2-competitive. Karp and Raghavan then observed that no deterministic list accessing algorithm can be better than 2-competitive, and therefore MTF is competitive-optimal. Later, Albers [1] presented *Timestamp*, and proved that it is also 2-competitive. In 1996, El-Yaniv [5] presented an infinite family of optimal deterministic algorithms, and showed that MTF and *Timestamp* are members of this family.

While the best performance of deterministic on-line algorithms is known, this is not the case for randomized algorithms against oblivious adversaries. Irani [6, 7] gave the first randomized list-accessing algorithm, SPLIT, and proved it to be $\frac{15}{8}$-competitive, thus beating the deterministic lower bound. Then Reingold, Westbrook, and Sleator [8] presented two families of randomized algorithms, *Counter* and *Random-Reset*. The best member of the two families achieves a competitive ratio of $\sqrt{3}$, and for several years it was the best randomized list accessing algorithm known. Later, Albers [1] presented *Timestamp(p)*, a randomized algorithm whose competitive ratio is the golden ratio $\frac{1+\sqrt{5}}{2}$. Finally, in 1995, Albers *et al.* [2] gave the *Combination* algorithm that is 1.6-competitive, the best upper bound known so far. The best lower bound known for the problem is due to Teia [10]. He proved that no randomized algorithm can be better than 1.5-competitive.

The paid exchange model, which was also defined in [9], received much less attention than the standard model. Most of the results concerning this model were given by Reingold, Westbrook, and Sleator in [8]. The best known deterministic algorithm, due to Reingold, Westbrook, and Sleator is 5-competitive, independent of d, while the best known deterministic lower bound is 3, [8]. In the same paper, the authors considered the *Random-Reset* algorithm in the paid exchange model, and were able to beat the deterministic lower bound. As expected, for each value of d, there is a different member of the family that has a better performance. Quite surprisingly though, the competitive ratio starts from 2.64 (for $d = 1$), and drops as d grows. No lower bound against oblivious adversaries is known for this model.

1.3 Results

The work in [8] suggests that a greedy algorithm, in the sense that the decision "where to place" the requested item is made without considering any other item, can attain a good competitive ratio. In the present work we further exploit such greedy algorithms. We establish a connection between randomized list accessing

algorithms and Markov chains. To every finite, irreducible Markov chain corresponds a list accessing algorithm. We name this family of randomized algorithms *Markov-Move-To-Front*(MMTF). We consider *Markov-Move-To-Front* in both the standard and the paid exchange models, and prove upper and lower bounds for its competitive ratio (against an oblivious adversary) that depend only on two parameters of the underlying Markov chain: π_0, the stationary probability of state 0, and S, the expected time to hit state 0, when in the stationary distribution.

The *Markov-Move-To-Front* family includes many well-known algorithms such as *Move-To-Front*, BIT, *Random-Move-To-Front*, the *Counter* and the *Random-Reset* families. The bounds we achieve are at least as good as the known bounds, thus in this aspect *Markov-Move-To-Front* presents a unified analysis for a wide class of interesting algorithms. Furthermore, the application of the general bounds to specific members of the family – such as *Random-Move-To-Front* and *Random-Reset* – yield some new, to our knowledge, results.

2 The MMTF Family

We introduce a family of algorithms that simplify the decision *where to place* the item just requested to *when to move it to front*. The engine of each algorithm is a Markov Chain M with a finite set of states $S_M = \{0, 1, ..., s\}$. A copy of the Markov chain is associated with each item in the list. The initialization of the Markov chains is made according the stationary distribution, and therefore, they will remain at the stationary distribution thereafter.
MMTF algorithm: *Upon a request for item x, serve the request, and then make a step in the Markov chain associated with x. If the state at which the algorithm just arrived is 0 move x to front; otherwise do nothing.*

Note that MMTF is *not* a mixed strategy, i.e., it is *not* a distribution over a set of deterministic algorithms, but it is rather a behavioral randomized algorithm [4].

3 The Standard Model

In this section we analyze the MMTF family in the standard model, and obtain upper and lower bounds against oblivious adversaries.

3.1 Upper Bound

For all the upper bounds in this paper we use the potential function method. Potential function arguments are quite common in the analysis of deterministic algorithms. In the randomized setting the same argument holds with some small adjustments. Let Φ be a potential function, and consider an event sequence $e_1, e_2, \ldots, e_n$. Let Φ_i be the value of Φ after the ith event. Φ_0 is the value of

Φ before the first event. Also let ALG and OPT be the on-line and optimal off-line algorithms respectively. Denote by ALG_i (OPT_i) the cost incurred by ALG (OPT) during the ith event, and define the *amortized cost* of ALG during the ith event to be $a_i = ALG_i + \Phi_i - \Phi_{i-1}$. Then we can prove the following lemma, where all the expectations are taken over the random choices made by the on-line algorithm[1].

Lemma 3. *Suppose there exists a constant c such that, with respect to all possible event sequences, for each event e_i, $E[a_i] \leq c \cdot OPT_i$, and $E[\Phi]$ is lower bounded by some constant. Then ALG is c-competitive against an oblivious adversary.*

Let M be an irreducible Markov Chain with set of states $S_M = \{0, 1, ..., s\}$, and transition probabilities $P = (p_{ij})$ that has a stationary distribution $\pi = (\pi_0, \pi_1, ..., \pi_s)$. For $i, j \in S_M$, h_{ij} is the hitting time from state i to state j in M. In our analysis, the hitting time *to* state 0 is of special interest, therefore for brevity we denote by h_i the hitting time h_{i0} [2]. By S we denote the expected hitting time *to* state 0, i.e, $S = \sum_{i=0}^{s} \pi_i h_i$.

Theorem 4. *Let M be an irreducible Markov Chain that has a steady-state distribution $\pi = (\pi_0, \pi_1, ..., \pi_s)$, and transition probabilities $P = (p_{ij})$. The MMTF algorithm that operates on M has competitive ratio that is upper bounded by* $\max\{1 + \pi_0 S, S\}$.

Proof. We start by defining the notion of an inversion. An inversion is an ordered pair $\langle y, x\rangle$ of items such that x appears before y in OPT's list, but it appears after y in the list maintained by MMTF (i.e., the pair indicates the relative order of the items in the on-line algorithm's list). We define the type $T(x)$ of an item x, whose Markov chain is at state i, to be the hitting time h_i. The type of an inversion $\langle y, x\rangle$ is then defined to be the type of x. As our potential function, we define $\Phi = \sum T(x)$, where the sum is taken over all inversions $\langle y, x\rangle$.

Consider now a request for item x, and assume that x is in position k in OPT's list. We will distinguish between two types of events. One is the service of the request by OPT and MMTF, including the possible free exchanges. The other is a paid exchange made by OPT.

Event 1: The cost of OPT for this event is k. The amortized cost of MMTF is $a = MMTF + \Delta\Phi \leq k + R + A + B + C$, where R is the number of inversions $\langle y, x\rangle$ at the time of the access, A is the change in the potential due to new inversions created, B is the change in the potential due to old inversions destroyed, and C is the change in the potential due to old inversions that change type. For the rest of the proof we will fix the value of R to r. Suppose that the state of x

[1] We consider finite request sequences of any length n. The algorithm uses a constant number, say c, of random bits to serve each request, therefore the total number of random bits needed to serve the whole request sequence is cn. The expectations are taken over those random strings of length cn.

[2] Recall that h_{ii} is the expected time to return to state i. Therefore $h_0 = h_{00} > 0$.

before the access is $I = i$, and after the access it is $J = j$. There are two cases to consider.
Case 1: If $j = 0$ then x is moved to front. In that case, exactly r inversions $\langle y, x\rangle$ are destroyed. The type of x was h_i. So $B = -rh_i$. Since there are no old inversions that change type, $C = 0$ and therefore

$$E[R + B + C \mid I = i, J = 0] = r - rh_i.$$

Case 2: If $j \neq 0$ then x is not moved to front. In that case, no inversions are destroyed, so $B = 0$. But because of the change of x's type, some inversions $\langle y, x\rangle$ might change type. The change in the potential due to this change is $C = r(h_j - h_i)$, so that

$$E[R + B + C \mid I = i, J = j \neq 0] = r + r(h_j - h_i).$$

And therefore, we can compute $E[R + B + C]$ as

$$\begin{aligned}
&\sum_{i=0}^{s}\sum_{j=0}^{s} E[R + B + C \mid I = i, J = j]Pr(I = i, J = j) \\
&= \sum_{i=0}^{s} E[R + B + C \mid I = i, J = 0]Pr(I = i, J = 0) + \\
&\quad \sum_{i=0}^{s}\sum_{j=1}^{s} E[R + B + C \mid I = i, J = j]Pr(I = i, J = j) \\
&= \sum_{i=0}^{s} \pi_i p_{i0} r(1 - h_i) + \sum_{i=0}^{s} \pi_i \sum_{j=1}^{s} p_{ij} r(1 + h_j - h_i) \\
&= r\sum_{i=0}^{s} \pi_i (p_{i0} - p_{i0}h_i + \sum_{j=1}^{s} p_{ij} + \sum_{j=1}^{s} p_{ij}h_j - \sum_{j=1}^{s} p_{ij}h_i) \\
&= r\sum_{i=0}^{s} \pi_i (1 - h_i + \sum_{j=1}^{s} p_{ij}h_j) \\
&= r(1 - S + \sum_{j=1}^{s} h_j \sum_{i=0}^{s} \pi_i p_{ij}) \\
&= r(-S + \sum_{j=0}^{s} \pi_j h_j) \\
&= 0
\end{aligned}$$

We turn now to the estimation of $E[A]$. Assume that after servicing the request, OPT moves x forward to position k'. Let y_i, $i = 1, ..., k-1$ be the items that preceded x in OPT's list before the access. Finally, let Y_i be a random variable that measures the change in the potential due to each pair $\{y_i, x\}$. Again we consider the same two cases as above.
Case 1: If $j = 0$, i.e, x is moved to front, then $k' - 1$ new inversions are created

$(\langle x, y_1\rangle, ..., \langle x, y_{k'-1}\rangle)$. Since each element in the list has its own copy of the Markov chain, each of the above inversions has expected type $\sum_{i=0}^{s} \pi_i h_i = S$, so

$$E[Y_j] = \begin{cases} S \text{ for } 1 \le j \le k'-1 \\ 0 \text{ for } k' \le j \le k-1 \end{cases}$$

In that case we can compute

$$E[A \mid case1] = \sum_{j=1}^{k-1} E[Y_j] = \sum_{j=1}^{k'-1} S = (k'-1)S$$

Case 2: If $j \neq 0$, i.e., x is not moved, then at most $k - k'$ new inversions are created due to the move by OPT $(\langle y_{k'}, x\rangle, ..., \langle y_{k-1}, x\rangle)$. The expected type of x is $\sum_{i=1}^{s} \pi_i h_i = S - 1$, so

$$E[Y_j] = \begin{cases} 0 & \text{for } 1 \le j \le k'-1 \\ S-1 & \text{for } k' \le j \le k-1 \end{cases}$$

$$E[A \mid case2] \le \sum_{j=1}^{k-1} E[Y_j] = \sum_{j=k'}^{k-1} (S-1) = (k-k')(S-1)$$

Case 1 occurs with probability π_0, and case 2 with probability $1 - \pi_0$. Hence, $E[A]$ is at most

$$\begin{aligned} &\pi_0(k'-1)S + (1-\pi_0)(k-k')(S-1) \\ &= (1 + 2\pi_0 S - S - \pi_0)k' + (1-\pi_0)(S-1)k - \pi_0 S \end{aligned}$$

E[A] is a linear function of k' and achieves a maximum either for $k' = 1$, or for $k' = k - 1$ (depending on the coefficient of k'). Therefore, $E[A] \le \max\{(1-\pi_0)(S-1), \pi_0 S\} \cdot k$. We are able now to compute an upper bound for the amortized cost for the request for x.

$$\begin{aligned} E[a] &\le E[k + R + A + B + C] \\ &= k + E[R + B + C] + E[A] \\ &\le (1 + \max\{(1-\pi_0)(S-1), \pi_0 S\}) \cdot k \\ &= \max\{1 + (1-\pi_0)(S-1), 1 + \pi_0 S\} \cdot OPT \end{aligned}$$

Event 2: A paid exchange made by OPT can create at most one inversion. The type of the inversion is $\sum_{i=0}^{s} \pi_i h_i = S$ on the average. On the other hand OPT pays 1 for the exchange. Hence

$$E[a] = S \cdot 1 = S \cdot OPT$$

Combining event 1 and event 2 we obtain

$$R_{OBL}(MMTF) \le \max\{1 + (1-\pi_0)(S-1), 1 + \pi_0 S, S\}.$$

In order to eliminate the term $1 + (1-\pi_0)(S-1)$ notice that

$$S \ge 1 \quad \Longrightarrow \quad S \ge 1 + (1-\pi_0)(S-1)$$

Therefore, $R_{OBL}(MMTF) \le \max\{1 + \pi_0 S, S\}$.

3.2 Lower Bound

We will determine now a lower bound for MMTF that operates an any Markov chain M. As it will become clear, the lower bound is non-trivial only for Markov chains with $S \geq \frac{3}{2}$.

Theorem 5. *Let MMTF operate on a Markov chain M. Then* $\forall \epsilon > 0$, $R_{OBL}(MMTF) > S - \epsilon$.

Proof. We will prove the theorem by describing a request sequence that enforces the above lower bound. More specifically, we will show that for any given $\epsilon > 0$, there exists a sufficiently large list, such that $E[MMTF(\sigma)] > (S-\epsilon) \cdot MTF(\sigma)$.

Assume that the initial configuration of MTF's list is $\langle x_1, x_2, ..., x_\ell \rangle$ with x_1 at the front, and at the same time MMTF's list is a permutation ρ, i.e., for each i, $1 \leq i \leq \ell$ the item x_i is at position $\rho(i)$ in MMTF's list. Let k be some integer whose value will be determined later, and consider the following request sequence:

$$\sigma = (x_1)^k, (x_2)^k, ..., (x_\ell)^k.$$

The cost incurred by MTF is

$$MTF(\sigma) = (1 + 2 + ... + \ell) + \ell(k-1) = \frac{\ell(\ell+1)}{2} + \ell(k-1).$$

MMTF, on the other hand, will move each requested item to front every S requests on the average, so its expected cost $E[MMTF(\sigma)]$ is at most

$$\begin{aligned} & (\rho(1)S + \rho(2)S + ... + \rho(\ell)S) + \ell(k-S) \\ = & S \cdot \frac{\ell(\ell+1)}{2} + \ell(k-S) \end{aligned}$$

because, when MMTF services a request for x_i, the fact that it is (eventually) moved to front does not affect future requests for elements that are behind x_i in the list, but it does make a difference for elements that are positioned in front of x_i. The cost to access those elements increases by one because of the move-to-front action. Thus the cost of $\rho(i)$ attributed to the S first requests for x_i is clearly a lower bound of the expected cost. Note, that the right hand side of the above inequality is independent of the initial configuration of MMTF's list. Then

$$\begin{aligned} \frac{E[MMTF(\sigma)]}{MTF(\sigma)} &= \frac{S\ell(\ell+1) + 2\ell(k-S)}{\ell(\ell+1) + 2\ell(k-1)} \\ &= \frac{S + 2\frac{k-S}{\ell+1}}{1 + 2\frac{k-1}{\ell+1}} \end{aligned}$$

If we choose $\ell \gg k$, formally if we let $\frac{k}{\ell} \longrightarrow 0$, then the ratio goes to S

$$\frac{E[MMTF(\sigma)]}{MTF(\sigma)} \longrightarrow S \tag{1}$$

By definition, 1 means that

$$\forall \epsilon > 0 \quad S - \epsilon < \frac{E[MMTF(\sigma)]}{MTF(\sigma)} < S + \epsilon$$

The left inequality gives the lower bound. To complete the proof, it remains to say that this procedure can be repeated enough times to outweigh any additive constant.

It is interesting to mention here that the above result is not only expected, but also occurs with high probability. In order for MMTF to incur a (significantly) different cost, it has to move the requested elements to front with significantly different rate, than once every S requests. That is, the time T_0 to hit state 0 has to deviate from its expected value $E[T_0] = S$. From Chebyshev's inequality for the random variable T_0 we get the high probability result, namely

$$Pr(\mid T_0 - S \mid \geq \frac{S}{\sqrt{\delta}}) \leq \delta.$$

3.3 Some Members of the Family

In this section, we present some well-known algorithms that are members of the MMTF family, and we apply the general bounds of the previous sections.

Random-Move-To-Front(p). RMTF(p) algorithm works as follows. Upon a request for an item x, RMTF(p) serves the request, and then with probability p moves x to front. For $p = 0$, items are never moved, so RMTF(0) clearly does not achieve a bounded competitive ratio. We can prove the following theorem[3].

Theorem 6. *Let* $0 < p \leq 1$. *Then* $\frac{1}{p} \leq R_{OBL}(RMTF(p)) \leq \max\{2, \frac{1}{p}\}$.

Proof. RMTF(p) can be formulated as an MMTF algorithm that operates on the simple two-state Markov Chain shown in Figure 1.

The transition matrix of the chain is the following

$$P = \begin{pmatrix} p & 1-p \\ p & 1-p \end{pmatrix}$$

It is not difficult to see that for this Markov chain $(\pi_0, \pi_1) = (p, 1-p)$, and $h_0 = h_1 = \frac{1}{p}$. Therefore, $S = \pi_0 h_0 + \pi_1 h_1 = \frac{1}{p}$. We can now apply Theorem 4 and Theorem 5 to obtain the result.

Counter(k, {0}) family. The *Counter(k,{0})* family, presented by N. Reingold, J. Westbrook, and D. Sleator [8], is a special case of the MMTF family. The upper bound of Theorem 4 when applied to this subclass of algorithms gives the same bounds as computed by Reingold *et al.*

[3] J. Westbrook has obtained a better bound for RMTF, namely $\max\{\frac{1}{p}, \frac{2}{2-p}\}$.

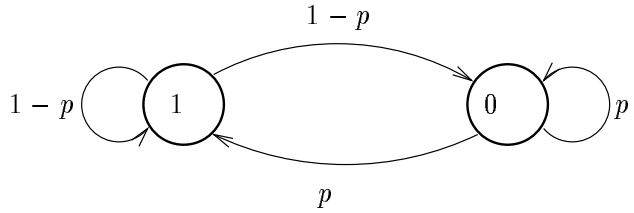

Fig. 1. The Markov chain for RMTF

A good two-state algorithm. Consider now the general form of a two-state algorithm, shown in Figure 2, with transition matrix

$$P = \begin{pmatrix} 1-q & q \\ p & 1-p \end{pmatrix}$$

The vector of the stationary probabilities of the above chain is $(\pi_0, \pi_1) = (\frac{p}{p+q}, \frac{q}{p+q})$, and the hitting times are $h_0 = \frac{p+q}{p}$, and $h_1 = \frac{1}{p}$, therefore $S = 1 + \frac{q}{p(p+q)}$. Using Theorem 4, we can express now an upper bound on the competitive ratio as a function of p and q. Interestingly enough, it turns out that the optimal[4] values are $p = q = 1$, i.e., under our analysis the best two-state algorithm is BIT (introduced by Reingold, Westbrook, and Sleator [8]), which achieves a competitive ratio of 1.75. Recall that BIT is randomized because the starting state is randomly chosen. BIT is a good example of a mixed randomized strategy. It is a distribution over two deterministic algorithms: one that starts at state 0, and moves an element to front every second time it is requested, and one that starts at state 1, and moves an element to front every second time it is requested. RMTF(p), on the other hand, is a behavioral randomized algorithm. The behavior of the two algorithms is quite different [5] , and, as shown in the previous sections, the mixed strategy (i.e., BIT) achieves a better competitive ratio than the behavioral algorithm (i.e., RMTF(p) for $p \leq \frac{1}{2}$). For a discussion on the differences between mixed and behavioral randomized strategies, the reader is referred to [4].

Random-Reset. In [8], Reingold *et al.*, argued that the best possible member of their *Random-Reset* family has three states, and they determined the best choice for the probabilities to reset to state 1 and to state 2. Subsequently, they proved an upper bound of $\sqrt{3}$ for the competitive ratio of this particular member. For many years this was the best randomized algorithm for list accessing known (see Figure 3). It is interesting to see now, that this algorithm is a member of the MMTF family, and its analysis follows from the general analysis. In particular, for this algorithm $S = \sqrt{3}$, and also $1 + \pi_0 S = \sqrt{3}$, and therefore,

[4] optimal under this analysis.

[5] J. Westbrook was the first to point out the difference in the behavior of the two algorithms, and proved that for $p = 2$ the competitive ratio of RMTF is lower bounded by 2, and thus worse than the competitive ratio of BIT.

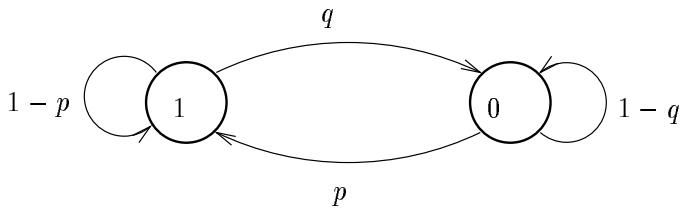

Fig. 2. The MMTF with two states

from theorems 4 and 5 we conclude that $\sqrt{3}$ is both an upper and a lower bound, and therefore the competitive ratio of this algorithm.

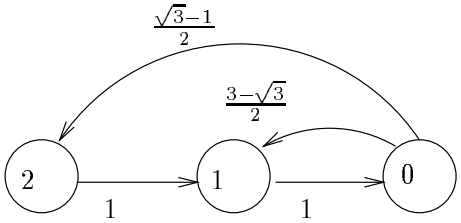

Fig. 3. The optimal *Random-Reset* algorithm

4 Paid Exchange Model

It is possible to prove similar upper and lower bounds for the performance of MMTF in the paid exchange model. Recall that in this cost model, every exchange has a cost d, where d is a constant typically greater than 1. Again, as in the standard model, the cost of accessing the ith item in the list costs i.

4.1 Upper Bound

Let M be an irreducible Markov chain with transition probabilities $P = (p_{ij})$, and a stationary distribution $\pi = (\pi_0, \pi_1, ..., \pi_s)$. $S = \sum_{i=0}^{s} \pi_i h_i$ as in the previous section. Then we can prove the following theorem.

Theorem 7. *The MMTF algorithm that operates on M has competitive ratio that is upper bounded by* $\max\{1 + \pi_0(2d + S), 1 + \frac{S}{d}\}$.

Proof. The analysis is very similar to that for the standard model. The notions of the inversion and the type of an inversion are defined as in the proof of Theorem 4. We define now our potential function to be $\Phi = \sum(d + T(x))$, where the sum is taken over all inversions $\langle y, x \rangle$. Intuitively, $T(x)$ pays for the increased access

cost because of the inversion, and d pays for the cost of removing the inversion, when the item is moved to front.

Consider a request for item x, which is at position k in OPT's list. In this model, since all exchanges are paid, the types of events we consider are slightly different. The first type is the service of a request by MMTF or OPT, and the exchanges made by MMTF. The second type is an exchange made by OPT.
Event 1: OPT will pay k to service this request. The amortized cost for MMTF is $a = MMTF + \Delta\Phi \leq k + R + A + B + C + D$, where R is the number of inversions of the form $\langle y, x\rangle$ at the time of the access, A is the change in the potential due to new inversions created, B is the change in the potential due to old inversions destroyed, C is the change in the potential due to old inversions that change type, and D is the cost of (paid) exchanges made by MMTF. For the rest of the proof we will fix the value of R to r.

Suppose that the state of x is $I = i$ before the access and $J = j$ after the access. We consider the following two cases:
Case 1: $j = 0$, so x is moved to front. In this case exactly r inversions of the form $\langle y, x\rangle$ are destroyed. The type of x was h_i, so $B = -r(d+h_i)$. There are not any old inversions that change type, so $C = 0$. Also, $D \leq (k + r - 1)d$. Finally, at most $k - 1$ new inversions of the form $\langle x, y\rangle$ are created. Each such inversion has expected type $\sum_{i=0}^{s} \pi_i h_i = S$, so $A \leq (k - 1)(d + S)$. Therefore,

$$E[R + A + B + C + D \mid I = i, J = 0] \leq r(1 - h_i) + (k - 1)(2d + S)$$

Case 2: $j \neq 0$, so x is not moved to front. In this case no new inversions are created, and no old inversions are destroyed, so $A = 0$ and $B = 0$. However, x might change type, causing a change of type to the old inversions of the form $\langle y, x\rangle$. The change in the potential due to this change is $C = r(h_j - h_i)$. Also, $D = 0$, since MMTF does not perform any exchanges.

$$E[R + A + B + C + D \mid I = i, J = j \neq 0] = r + r(h_j - h_i)$$

We can now see that the expected value of $R + A + B + C + D$ equals

$$\begin{aligned}
&\sum_{i=0}^{s} E[R + A + B + C + D \mid I = i, J = 0]\pi_i p_{i0} + \\
&\sum_{i=0}^{s}\sum_{j=1}^{s} E[R + A + B + C + D \mid I = i, J = j]\pi_i p_{ij} \\
&= \sum_{i=0}^{s} \pi_i p_{i0}(r(1 - h_i) + (k - 1)(2d + S)) + \\
&\sum_{i=0}^{s} \pi_i \sum_{j=1}^{s} p_{ij} r(1 + h_j - h_i) \\
&= r(\sum_{i=0}^{s} \pi_i p_{i0}(1 - h_i) + \sum_{i=0}^{s} \pi_i \sum_{j=1}^{s} p_{ij}(1 + h_j - h_i)) +
\end{aligned}$$

$$\sum_{i=0}^{s} \pi_i p_{i0}(k-1)(2d+S)$$

Note that the expression r is multiplied with is the same as in the previous section, and is identically 0, therefore

$$\begin{aligned} E[R+A+B+C+D] &= \sum_{i=0}^{s} \pi_i p_{i0}(k-1)(2d+S) \\ &= (k-1)(2d+S)\sum_{i=0}^{s} \pi_i p_{i0} \\ &= \pi_0(k-1)(2d+S) \end{aligned}$$

We can now bind the amortized cost of MMTF for this event:

$$E[a] \leq k + E[R+A+B+C+D] \leq k[1+\pi_0(2d+S)]$$

Event 2: A transposition by OPT can create at most one inversion. The expected change in the potential due to that inversion is at most

$$\sum_{i=0}^{s} \pi_i(d+h_i) = d + \sum_{i=0}^{s} \pi_i h_i = d+S$$

On the other hand, OPT pays d. Therefore,

$$E[a] = E[\Delta\Phi] \leq d+S = (1+\frac{S}{d})\cdot d = (1+\frac{S}{d})\cdot OPT$$

From the above we conclude that the competitive ratio of MMTF against an oblivious adversary is upper bounded by $\max\{1+\pi_0(2d+S), 1+\frac{S}{d}\}$.

4.2 Lower Bound

For the lower bound we describe two request sequences, the first of which enforces the competitive ratio to be at least $1+\frac{S-1}{d+1}$, and the second enforces it to be at least $1+\frac{d}{S}$.

Theorem 8. *Let MMTF operate on a Markov chain M. Then $R_{OBL}(MMTF) \geq \max\{1+\frac{d}{S}, 1+\frac{S-1}{d+1}\}$.*

Proof. For the first sequence we will compare MMTF with MTF. Let $\langle x_1, x_2, ..., x_\ell\rangle$ be the MTF's list, and let item x_1 be in position $\rho(i)$ in the list maintained by MMTF. Consider the request sequence $\sigma = (x_1)^k, (x_2)^k, ..., (x_\ell)^k$. The value of k will be determined later. The cost of MTF to service σ is

$$\begin{aligned} &(1+2+...+\ell) + \ell(k-1) + d(0+1+...+(\ell-1)) \\ &\leq (d+1)\frac{\ell(\ell+1)}{2} + \ell(k-1) \end{aligned}$$

On the other hand, MMTF will move each requested item to front once every S request, on the average. Also notice that a move-to-front action does not affect the elements that are behind the moved element, but will increase the access cost of elements that are in front of it in the list and have not been requested yet. Therefore, to assume that item x_i is in position $\rho(i)$ when it is first requested gives a lower bound to the access cost incurred by MMTF. Therefore, $MMTF(\sigma)$ is at least

$$\begin{aligned}&(\rho(1)+\rho(2)+...+\rho(\ell))S+\ell(k-S)+(\rho(1)+\rho(2)+...+\rho(\ell)-\ell)d\\&=(d+S)\frac{\ell(\ell+1)}{2}+\ell(k-S)-\ell d\end{aligned}$$

If we choose $\ell \gg k$, formally $\frac{k}{\ell} \longrightarrow 0$, then we get

$$\frac{E[MMTF(\sigma)]}{MTF(\sigma)} \longrightarrow \frac{d+S}{d+1}$$

which by definition means that

$$\forall \epsilon > 0 \quad \frac{d+S}{d+1} - \epsilon < \frac{E[MMTF(\sigma)]}{MTF(\sigma)} < \frac{d+S}{d+1} + \epsilon$$

The left inequality gives the lower bound of $\frac{d+S}{d+1}$.

Consider now the request sequence $\sigma = (x_1, x_2, ...x_l)^k$. The cost incurred by OPT is at most the cost incurred by the algorithm that services the request sequence without moving any item, i.e., $OPT(\sigma) \leq k\frac{\ell(\ell+1)}{2}$. MMTF will move an item once every S requests, on the average, so its cost on σ is

$$(\rho(1)+...+\rho(\ell))k+\frac{k}{S}(\rho(1)+...+\rho(\ell)-\ell)d = k\frac{\ell(\ell+1)}{2}(1+\frac{d}{S})-\ell d$$

Taking $\frac{k}{\ell} \longrightarrow 0$, the ratio approaches $1+\frac{d}{S}$.

It remains to say that both request sequences can be repeated sufficiently many times to outweigh any additive constant.

4.3 Some Members of the Family

In this section, we apply the general bounds determined for the paid exchange model to some well known members of the family. Those bounds, as expected, depend on the scaling parameter d, and their limit behavior, as $d \longrightarrow \infty$ is of interest.

Counter(k, {0}). In [8], Reingold *et al.*, analyzed the Counter(k, {0}) algorithm in the paid exchange model, and were able to determine an upper bound of its competitive ratio in terms of the maximum counter value k and d. Subsequently, they observed that as d tends to infinity, the best ratio (i.e., the ratio that corresponds to the best choice of k for the particular d) tends to $(5+\sqrt{17})/4$.

In this section we will show that the above limit is tight. We begin by describing *Counter*(k, $\{0\}$) as an MMTF algorithm. As shown in Figure 4, *Counter*(k, $\{0\}$) is a simple directed cycle of $k+1$ nodes. In the stationary distribution, $\pi_0 = \pi_1 = ... = \pi_k = \frac{1}{k+1}$. Also, $h_0 = k+1$, and for $1 \leq i \leq k$, $h_i = i$. Therefore, $S = \frac{k+2}{2}$ and by Theorem 7 we get an upper bound for the competitive ratio $R_{OBL}(COUNTER)$:

$$R_{OBL}(Counter) \leq \max\{1 + \frac{k+2}{2d}, 1 + \frac{1}{k+1}(2d + \frac{k+2}{2})\}$$

Now fix d, and set $1 + \frac{k+2}{2d} = 1 + \frac{1}{k+1}(2d + \frac{k+2}{2})$. This equation has one non-negative solution

$$k = -\frac{3}{2} + \frac{d}{2} + \frac{1}{2}\sqrt{17d^2 + 2d + 1}$$

Since $R_{OBL}(Counter)$ has only one local minimum, which is also global, we conclude that the optimal choice for k is either the floor or the ceiling of the above expression. If we take the limit now, as d tends to infinity, $R_{OBL}(Counter)$ is upper bounded by

$$\lim_{d\to\infty}(1 + \frac{1}{2d}\lceil -\frac{3}{2} + \frac{d}{2} + \frac{1}{2}\sqrt{17d^2 + 2d + 1}\rceil) = \frac{5+\sqrt{17}}{4} \qquad (2)$$

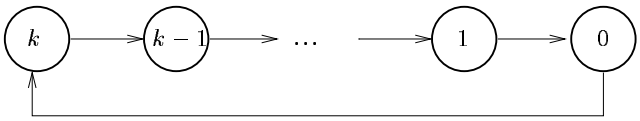

Fig. 4. The *Counter* algorithm

Consider now the lower bound for the competitive ratio for the above, optimal choice of k

$$\lim_{d\to\infty}(1 + \frac{1}{d+1}(\lfloor -\frac{3}{2} + \frac{d}{2} + \frac{1}{2}\sqrt{17d^2 + 2d + 1}\rfloor - 1)) = \frac{5+\sqrt{17}}{4} \qquad (3)$$

From 2 and 3 we conclude that

$$\lim_{d\to\infty} R_{OBL}(Counter) = \frac{5+\sqrt{17}}{4}$$

Random-Move-To-Front(p). Consider now the RMTF(p) algorithm. In Section 3.3 we determined that for RMTF(p), $S = \frac{1}{p}$. It is not difficult to compute now the optimal choice of the probability p, given the value of d. From Theorem 7 we get that

$$R_{OBL}(RMTF) \leq \max\{1 + \frac{1}{pd}, 2 + 2pd\}$$

For a particular d, the best choice for p is when the two expressions are equal, that is

$$1 + \frac{1}{pd} = 2 + 2pd \quad \Longleftrightarrow \quad p = -\frac{1}{d} \quad or \quad \frac{1}{2d} \tag{4}$$

As $p \geq 0$, the only choice is $p = 1/(2d)$, and then the competitive ratio is 3, independent from d. The lower bound, on the other hand, depends on d.

$$\begin{aligned} R_{OBL}(RMTF) &\geq \max\{1 + \frac{1}{pd}, 1 + \frac{\frac{1}{p} - 1}{d+1}\} \\ &= \max\{1.5, \frac{3d}{d+1}\} \end{aligned} \tag{5}$$

From 4 and 5 we get

$$\frac{3d}{d+1} \leq R_{OBL}(RMTF) \leq 3$$

which is relatively tight, and clearly

$$\lim_{d \to \infty} R_{OBL}(RMTF) = 3$$

5 Concluding Remarks

This paper leaves open several questions. One is to determine the correct competitive ratio for the *Markov-Move-To-Front* family. This would imply the correct competitive ratio for a number of interesting algorithms such as *Random-Move-To-Front* and BIT. Although the presented family includes a wide variety of algorithms, we were not able to find one that outperforms *Random-Reset*. It is interesting to further investigate the family, and see if a better member exists, or alternatively prove that *Random-Reset* is the best member of the family. In this work, we required that each element in the list has a copy of the *same* Markov chain. It is interesting to consider an algorithm, where each element is allowed to have a different chain. This would allow "important" items to move to front more frequently. Although in an adversarial setting this might not improve the performance, it is possible that in practice, or in a model that captures *locality of reference*, such an algorithm would have a better performance.

6 Acknowledgments

The author would like to thank Allan Borodin, Adi Rosén and Mike Molloy for many interesting discussions.

References

1. Albers, S.: Improved Randomized On-line Algorithms for the List Update Problem. Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (1995) 412-419
2. Albers, S., von Stengel, B., Werchner, W.: A Combined BIT and TIMESTAMP Algorithm for the List Update Problem. TR-95-039 (1995), International Computer Science Institute, Berkeley
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Draft (1996)
4. Borodin, A., El-Yaniv, R.: On Randomization in Online Computation. To appear in 12th IEEE Conference on Computational Complexity (1997).
5. El-Yaniv, R.: There are Infinitely Many Competitive-Optimal Online List Accessing Algorithms. Submitted to SODA 97.
6. Irani, S.: Two Results on the List Update Problem. Information Processing Letters **38** (6) (1991) 202-208.
7. Irani, S.: Corrected Version of the SPLIT Algorithm for the List Update Problem. (1996) ICS Department, U.C. Irvine. Technical Report 96-53. Note: Corrected version of the SPLIT algorithm appearing in [6].
8. Reingold, N., Westbrook, J., Sleator, D.: Randomized Competitive Algorithms for The List Update Problem. Algorithmica **11** (1994) 15-32.
9. Sleator, D., Tarjan, R.: Amortized Efficiency of List Update and Paging Rules. Communications of the ACM **28** (2) (1985) 202-208.
10. Teia, B.: A Lower Bound for Randomized List Update Algorithms. Information Processing Letters **47** (1993) 5-9.

This article was processed using the LaTeX macro package with LLNCS style