DIMACS Technical Report 98-44

September 1998

List Update Problem

by

Béla Csaba[1] Sachin Lodha[2]

[1]**Email:** bcsaba@paul.rutgers.edu
Address: Dept. of Computer Science
Rutgers University, New Brunswick, New Jersey 08854
[2]**Email:** lodha@paul.rutgers.edu
Address: Dept. of Computer Science
Rutgers University, New Brunswick, New Jersey 08854

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Bellcore and NEC Research Institute.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We present an optimal on-line algorithm for the *List Update Problem* when the request sequence has some *particular* structure.

1 Introduction

1.1 The General Setting

We investigate the *List Update Problem.* The problem can be briefly stated as follows :

There are n different items in a doubly-linked list. Every item has its own unique key. Access requests are coming for these items, one at a time, in the form of their key values. We satisfy the request by finding an item of the corresponding key value in the doubly-linked list. If the position of the item requested is i in the list, then the cost of this operation is i units. While satisfying such a request, we are allowed to move the requested item in any of the first i positions for no cost. The goal is to satisfy all requests at minimum possible cost.

In the off-line setting, we know the request sequence in advance. On the other hand, in the on-line setting, we have to satisfy a request without the knowledge of any future request. Let ρ denote a finite length request sequence coming from $\{1, \ldots, n\}^*$. Let $C_{OPT}(\rho)$ denote the cost of the optimal satisfaction of ρ[1]. Let A be a deterministic on-line algorithm and let $C_A(\rho)$ denote the cost of A while satisfying ρ. Now if we could find a constant K such that

$$\forall \rho \quad C_A(\rho) \leq \quad c \cdot C_{OPT}(\rho) + K \tag{1}$$

then we say A is c-competitive. This notion of competitiveness first appeared in [ST-85] and since then it has been most useful in analyzing the performance of on-line algorithms.

1.2 Previous Work

Sleator and Tarjan, [ST-85], were first to show that *Move-To-Front* is a 2 competitive strategy. Then came Karp and Raghavan, who proved in [KR-90] that $c \geq 2$ for any deterministic on-line algorithm implying MTF is an optimal on-line strategy. Later Suzanne Albers gave *Timestamp(0)* algorithm in [A-95] which is again optimal in on-line setting.

In the case of randomized on-line algorithm setting, Teia showed a lower bound of 1.5 for the competitive ratio c in [T-93], but with an oblivious adversary who is the least powerful among all standard adversaries.

1.3 Motivation

It is natural to think whether knowing the structure of the request sequence could be of any help or not, *ie* beating the barrier of 2 on the competitive ratio of on-line algorithms. This assumption may not be impractical, since in practice we may expect some kind of "pattern" in the incoming requests.

1.4 Results

Hence, in this report, we consider two different kind of restrictions on the requests. For them, we give two different optimal on-line strategies.

[1]since the request sequence is of finite length, we can do so in a finite amount of time [RW-96]

2 Preliminaries

Let L denote a doubly linked list containing n different items. Without loss of generality, we can assume that their keys are unique and they range from 1 to n.[2]

Every order of the items in the list defines a configuration. Thus, there are $M = n!$ different configurations. We'll refer to them as $C_1, C_2, \ldots, C_M$.

Swapping two adjacent items in the list is called *transposition* operation. Let $C(i,j)$ denote the minimum number of transpositions needed to change from C_i to C_j. We say configurations C_i and C_j are *adjacent* if $C(i,j) = 1$.

Claim 1 *For any two configurations C_i and C_j, $C(i,j) \leq \binom{n}{2}$.*

Proof : Move the item x , last in C_j, from its current position in C_i to the last position. This takes $\leq (n-1)$ transpositions. Use induction for the two configurations on $n-1$ items in the first $n-1$ positions.

□

We use α, β and γ to denote request sequences of finite lengths l_α, l_β and l_γ respectively. The set $\{1, 2, \ldots, n\}$ is denoted by V. With this notation, $\alpha, \beta, \gamma \in V^*$.

3 ONREG Languages

Our request sequences come from V^*. We define a subclass ONREG over V^*. Let's describe this class with following two rules

1. If $\alpha \in V^*$, then $\{\alpha\}$ and $\{\alpha\}^*$ are in ONREG;
2. If $L, L' \in ONREG$, then LL' and $L \bigcup L'$ are in ONREG.

Note that $LL' = \{xy | x \in L, y \in L'\}$ is the concatenation of L and L', $L \bigcup L'$ is their union. We will refer to $L \subset \{\alpha^i | i \in \mathcal{N}\} \in \mathcal{V}^*$ as "atomic" languages for obvious reasons.

Theorem 1 *For any language in ONREG, there is a* 1*-competitive deterministic on-line algorithm.*

Remark : It is easy to pick up a language L from ONREG for which MTF and Timestamp(0) are 2-competitive. For this, let L be α^* where $\alpha = 123 \ldots n$, and initial configuration C_{init} be $123 \ldots n$. From the definition of MTF and Timestamp(0), it follows that each of them will pay average cost of n for each request. If we keep this configuration static, then we can pay not more than $n/2$ on the average, which can be proven as the optimal cost.

We will prove *theorem* 1 only for "atomic" languages. It will follow for the remaining cases from [CD-97].

[2] Otherwise, we can discard all but one item with the same key value.

3.1 Algorithm and Proof

Lemma 1 *If L is a finite subset of V^*, then there is a 1-competitive deterministic on-line algorithm for L.*

Proof : Let $L = \{w_1, w_2, \ldots, w_{|L|}\}$ and let $H = \max_{1 \leq i \leq |L|} |w_i|$. Then, letting $K = n \cdot H$ in equation 1 will give us the desired result.

□

Lemma 2 *Let $L \subset \{\alpha^i | i \in \mathcal{N}\}$, then there is a 1-competitive deterministic on-line algorithm for L.*

Proof : We will prove this result for $L = \{\alpha^i | i \in \mathcal{N}\}$ and then the statement of *lemma* 2 will follow immediately.

Here is our algorithm - First, we construct a table T of size $M \times M$ where $T(i,j) = \frac{cost(\alpha^i, q_j)}{i \cdot l_\alpha}$. Note that $cost(\alpha^i, q_j) =$ The optimal cost of satisfying α^i starting and ending in q_j.

Let $T(i_0, j_0)$ be the minimal entry in T. We break tie by choosing first pair (i_0, j_0) in their lexicographic ordering. Given the list in initial configuration I, we first transform it into configuration q_{j_0} at cost $C(I, q_{j_0})$. By *lemma* 1, $C(I, q_{j_0}) \leq \binom{n}{2}$. Then, follow the optimal satisfaction of α^{i_0} repetitively.

Claim 2 *Let $p \in \mathcal{N}$ and q be any configuration. Then, $T(i_0, j_0) \leq \frac{cost(\alpha^p, q)}{p \cdot l_\alpha}$.*

Proof : If $p \leq M$, then the statement is trivially true. Let's suppose that it's true for $p < k$. Let's consider any satisfaction which starts and ends in q for α^k. Since $k \geq M + 1$, there is a q' such that it appears twice on the first letter of α in that satisfaction. Due to induction hypothesis, we know that the average cost of satisfaction between the two q's is not smaller than $T(i_0, j_0)$.

If the statement of the lemma were not true for k and q, then it means that the average cost of satisfaction for the request sequence without the part between the two q's is smaller than $T(i_0, j_0)$. But this contradicts with the induction hypothesis.

□

Now we are ready to prove *lemma* 2. Given α^p and initial configuration I, consider any satisfaction S of it. If $p \leq M$, then we are trivially done since we can let our $K = n \cdot l_\alpha \cdot M$ in *equation* 1 and we are done.

So let $p \geq M + 1$. Therefore, there is a q such that it appears twice on the first letter of α in that satisfaction. There may be many such possible q's, choose that q which is not contained in any such repetition. Call this part of the request sequence $\beta = \alpha^k$. This choice of q implies that the length of the remaining part is less than $M \cdot l_\alpha$.

Let's denote the cost of our algorithm on β by C' and the cost of the satisfaction S be C. It is enough if we prove that $C' \leq C + M \cdot n \cdot l_\alpha$. Then, we can choose $K = \binom{n}{2} + 2 \cdot n \cdot M \cdot l_\alpha$ in *equation* 1 and the *lemma* 2 follows. By the definition of cost function, we have

$$cost(\alpha^k, q) \;\; \leq \;\; C \tag{2}$$

By *claim* 2, it follows that

$$T(i_0, j_0) \;\; \leq \;\; cost(\alpha^k, q)/(k \cdot l_\alpha) \tag{3}$$

Simple observation shows that

$$C' \leq \lfloor \frac{k}{i_0} \rfloor cost(i_0, j_0) + n \cdot M \cdot l_\alpha \quad (4)$$

$$C' \leq \frac{k}{i_0} cost(i_0, j_0) + n \cdot M \cdot l_\alpha \quad (5)$$

$$C' \leq k \cdot l_\alpha \cdot T(i_0, j_0) + n \cdot M \cdot l_\alpha \quad (6)$$

But from equations 3 and 2 respectively

$$C' \leq C + n \cdot M \cdot l_\alpha \quad (7)$$

□

Proof of theorem 1 We derive our notion of ONREG from the one of $ONREG_0$ in the paper [CD-97]. In that paper, it is proved that for the union and concatenation of two $ONREG_0$ languages there is a 1-competitive deterministic on-line algorithm (refer to *Lemma* 3.2 and *Lemma* 3.3 in [CD-97]). Therefore, the statement of *theorem* 1 follows from these lemmas and *lemma* 1 and *lemma* 2 proven above.

□

4 Access Graphs

Knowing the pattern in the request sequence, can one hope to improve the performance of an on-line algorithm? Such question has already been investigated for the *paging problem* in [BIRS-95]. In that paper, they make use of "access graphs" to enhance the working of an on-line algorithm.

4.1 Model and Algorithm

Our assumption is that once an item i is requested, the next requested item j comes from a "small" pool of items depending only on i. So one can associate a "neighborhood" $N(i)$ with every item i. We can use graphs to describe these dependencies.

Let's define the "access graph" based on these dependencies. $\vec{G} = (V, \vec{E})$, where

$$\begin{aligned} V &= \{1, 2, \ldots, n\}, \\ \vec{E} &= \{(i,j) | i, j \in V; j \in N(i)\} \end{aligned}$$

Such a graph can be constructed by empirical observations or a-priori knowledge about the requests.

Here is the algorithm - At each node i in the access graph G, we store the key of i and the item i itself. Besides this, we also store the keys from $N(i)$ in a sorted order along with pointers to the corresponding nodes. Thus, a request can be fulfilled in time $\log |N(i)| + 1$. We always "point" to the last requested item.

4.2 Comparison with MTF

MTF is known to be the best deterministic on-line algorithm for the *list update problem.* But by working with another, possibly very slightly modified, structure we show that we can achieve a huge improvement in performance.

Let's consider an access graph $\vec{G}$ which is a directed cycle on n nodes. Therefore our request sequence will be in accordance with the cycle. For each request, we incur unit cost. On the other hand, average cost of MTF is $n/2$.

In case of a general graph G, if we can find a large directed cycle C, say of length L, with maximum degree of a vertex on the cycle D, then there is a request sequence on which we spend $L \log D$, but the MTF spends $L^2/2$. Define the request sequence to be in accordance with the cycle C.

It can be easily shown that a graph G with minimum degree d has a cycle of length at least $d+1$. To prove this, just consider a path of maximum length in G and the outgoing edges of one end-point.

Thus, for a general graph G, with minimum and maximum degrees d and D, we can outperform MTF by a factor of $O(\frac{d}{\log D})$ in some cases. If the access graph is a balanced degree graph *ie* the minimum degree and maximum degree don't differ "too" much, then this could be a huge improvement.

References

[A-95] S. Albers, "Improved randomized on-line algorithms for the list update problem", *In Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 412–419, 1995.

[BIRS-95] A. Borodin, S. Irani, P. Raghavan, B. Schieber, "Competitive paging with locality of reference", *Journal of Computer and System Sciences*, 50, pp. 244–258, 1995.

[CD-97] B. Csaba, G. Dányi, "Server problems and regular languages", *Acta Cybernetica*, 13, pp. 23–39, 1997.

[KR-90] R. Karp, P. Raghavan, Personal communication transmitted through [RWS-94].

[RW-96] N. Reingold, J. Westbrook, "Off-line algorithms for the list update problem", *Information Processing Letters*, 60, pp. 75–80, 1996.

[RWS-94] N. Reingold, J. Westbrook, D. Sleator, "Randomized competitive algorithms for the list update problem", *Algorithmica*, 11, pp. 15–32, 1994.

[ST-85] D.D. Sleator, R.E. Tarjan, "Amortized efficiency of list update and paging rules", *Communications of the ACM*, 28, pp. 202–208, 1985.

[T-93] B. Teia, "A lower bound for randomized list update algorithms", *Information Processing Letters*, 47, pp. 5–9, 1993.