

List Update Problem

CS 422 / CS 536

Vamsi Addanki

Purdue University
vaddank@purdue.edu

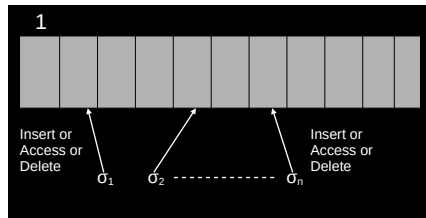
March 05, 2026

Overview

- 1 Preliminaries
 - Model
 - Potential function
 - Amortized costs
 - competitiveness
- 2 Deterministic online algorithms
 - Transpose
 - Frequency count
 - Move-to-Front
- 3 c-competitiveness lower bound
- 4 Packet Classification
 - Move-Recursively-Forward

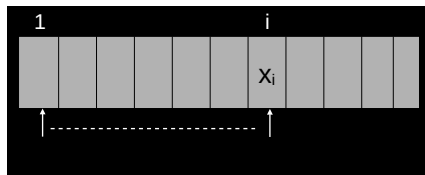
Preliminaries

- Maintain a dictionary
- Unsorted linear list L
- Request sequence σ
 - *access* an item in the list
 - *insert* an item into the list
 - *delete* an item in the list



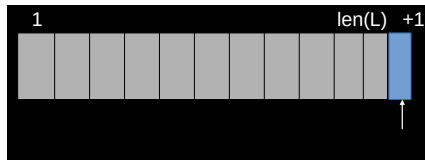
Preliminaries

- *access*: A list update algorithm starts at the front of the list and searches **linearly** through the items until the desired item is found
- cost is i , where i is the position of requested item



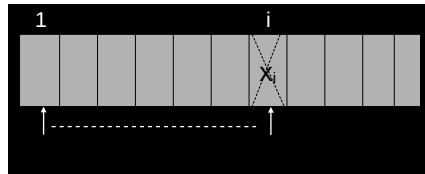
Preliminaries

- *insert*: To insert a new item, the algorithm first **scans the entire list** to verify that the item is not already present and then inserts the item at the **end of the list**
- cost is $\text{len}(L) + 1$



Preliminaries

- *delete*: To delete an item, the algorithm scans the list to search for the item and then deletes it
- cost is i , where i is the position of requested item



Preliminaries

- *free exchanges*: **After access or insert** of an item, moving it to the front costs 0.
- *paid exchanges*: At any time, adjacent items in L may be exchanged at cost 1.
For example, $\text{swap}(L[i], L[i + 1])$ costs 1.

Preliminaries

Two models,

- *Static list accessing*: only *access* operation is allowed
- *Dynamic list accessing*: any operation is allowed

Preliminaries

Problem: Minimize the cost of servicing a request sequence σ

- List update algorithm ALG
- Optimal offline algorithm OPT (for the same problem as ALG)
- $ALG(\sigma)$ is the cost incurred by ALG to service request sequence σ

Potential function Φ

- S_{ALG} and S_{OPT} denote the set of possible configurations of ALG and OPT
- $\Phi : S_{ALG} \times S_{OPT} \rightarrow \mathbb{R}$

Amortized costs

Let ALG_i denote the cost incurred by ALG during the i^{th} event and denote **amortized cost** of ALG for i^{th} event as a_i

- $a_i = ALG_i + \Phi_i - \Phi_{i-1}$

c-competitive

Definition (c-competitive)

$$\forall len(L), \forall \sigma, \exists \alpha \mid ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha$$

In order to prove ALG is c-competitive, it is enough to show¹

- $\forall i, a_i \leq c \cdot OPT_i$
- $\exists b \mid \forall i, \Phi_i \geq b$

¹ $a_i = ALG_i + \Phi_i - \Phi_{i-1}$

Deterministic online algorithms

- Transpose
- Frequency count
- Move-to-Front

Transpose

- After accessing or inserting an item at index i , $swap(L[i], L[i - 1])$

Frequency count

- Let f_i denote the frequency count of an item k in L
- Whenever i is requested, f_k++ ;
- Maintain L in nonincreasing order of frequency count

Move-to-Front (MTF)

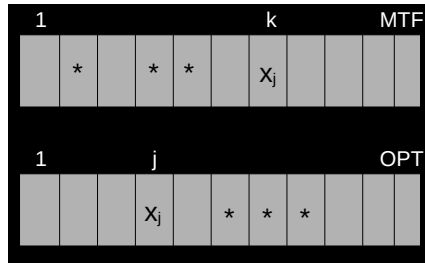
- Upon a request to item at index i , move it to front of L

Move-to-Front (MTF)

Theorem

The Move-to-Front algorithm is 2-competitive.

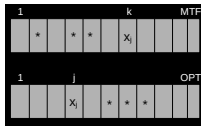
Move-to-Front (MTF)



Move-to-Front (MTF)

Proof (continued...):

- for any $t \mid 1 \leq t \leq m$, let x denote the item requested by $\sigma(t)$
- $k = \#$ of items that precede x in L_{MTF} and L_{OPT}
- $l = \#$ of items that precede x in L_{MTF} but follows x in L_{OPT}
- $MTF(t) = k + l + 1$
- $OPT(t) \geq k + 1$
- When MTF serves $\sigma(t)$ and moves x to front, l inversions are destroyed and at most k new inversions are created.
- Thus, $MTF(t) + \Phi(t) - \Phi(t-1) \leq MTF(t) + k - l$



Move-to-Front (MTF)

Proof (continued...):

$$\begin{aligned}
 MTF(t) + \Phi(t) - \Phi(t-1) &\leq MTF(t) + k - l = 2 \cdot k + 1 \\
 MTF(t) + \Phi(t) - \Phi(t-1) &\leq 2 \cdot OPT(t) - 1
 \end{aligned} \tag{1}$$

$$\left(\sum_{t=1}^m MTF(t) \right) + \Phi(m) - \Phi(0) \leq \sum_{t=1}^m 2 \cdot OPT(t) - m$$

Move-to-Front (MTF)

Proof (continued...):

$$MTF(\sigma) \leq 2 \cdot OPT(\sigma) - m + \Phi(m) \quad (2)$$

Since $\Phi(m)$ is non-negative, the proof follows from the definition of c – *competitive*
Def. 1

Move-to-Front (MTF)

Proof (continued...): Extension to allow insertion and deletion:

- On an insertion, $MTF(t) = OPT(t) = n + 1$, where $n = \text{len}(L_{MTF}) = \text{len}(L_{OPT})$ before insertion
 - at most n new inversions are created
- On a deletion, l inversions are removed and no new inversion is created

non-competitiveness of Transpose and FC

Theorem

Both Transpose and Frequency count are not c -competitive for any constant c .

non-competitiveness of Transpose and FC

Proof (Transpose):

- let L_0 denote our list initially
- consider a request sequence that asks for $L_0[\text{len}(L)]$ and $L_0[\text{len}(L - 1)]$ repeatedly
- for each pair of such requests, Transpose incurs a cost of $2 \cdot \text{len}(L)$
- OPT would simply move these two items to the front of the list, there by incurring a cost of 3 for each pair of requests
- similarly for a request sequence of size n , the cost ratio, online to opt is,
$$\frac{n \cdot \text{len}(L)}{(\frac{n}{2} - 1) \cdot 3 + 2 \cdot \text{len}(L)}$$

non-competitiveness of Transpose and FC

Proof (Transpose):

- $\lim_{n \rightarrow \infty} \frac{n \cdot \text{len}(L)}{(\frac{n}{2} - 1) \cdot 3 + 2 \cdot \text{len}(L)} = \frac{2 \cdot \text{len}(L)}{3}$
- since $\text{len}(L)$ has no a priori bound in dynamic list accessing problem, Transpose is non-competitive

non-competitiveness of Transpose and FC

Proof (FC):

- let $x_1, x_2, \dots, x_l$ be the items in the list
- let k be any integer such that $k > l$
- σ is such that, first x_1 is requested k times, then x_2 is requested $k - 1$ times and so on
 - x_i is requested $k + 1 - i$ times
 - notice, x_i are requested in the decreased order of relative frequencies
- upon requesting x_i , FC moves it to i^{th} position and never changes it's position

non-competitiveness of Transpose and FC

Proof (FC):

- $FC(\sigma) = \sum_{i=1}^l i \cdot (k+1-i) = \frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}$
- now, to prove a lower bound, we need some upper bound on $OPT(\sigma)$
- Assume the list is initially in the worst-case for *MTF*
 - x_i is located at last position when it is first requested (cost of l), moved to front and stays there for $k-i$ subsequent requests

$$MTF(\sigma) \leq \sum_{i=1}^l [l + (k-i)] = l \cdot (l+k) - \frac{l \cdot (l+1)}{2}$$

$$\Rightarrow \frac{FC(\sigma)}{OPT(\sigma)} \geq \frac{\frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}}{l \cdot (l+k) - \frac{l \cdot (l+1)}{2}}$$

non-competitiveness of Transpose and FC

Proof (FC):

$$\begin{aligned}\Rightarrow \frac{FC(\sigma)}{OPT(\sigma)} &\geq \frac{\frac{k \cdot l \cdot (l+1)}{2} + \frac{l \cdot (1-l^2)}{3}}{l \cdot (l+k) - \frac{l \cdot (l+1)}{2}} \\ \Rightarrow \lim_{k \rightarrow \infty} &= \frac{l+1}{2}\end{aligned}$$

Since $l = \text{len}(L)$, which has no a priori bound, FC is non-competitive

Lower bound for any deterministic online algorithm

Theorem

Let A be a deterministic online algorithm for the list update problem. If A is c -competitive, then $c \geq 2$

Lower bound for any deterministic online algorithm

Proof:

- let $x_1, x_2, \dots, x_l$ be the items in the list
- construct a request sequence of size m such that each request is made to the item that is at the last position in L_A
- $A(\sigma) = m \cdot n$
- OPT will first sort items in the nonincreasing order of request frequencies and then serves σ without any exchanges
 - in rearranging OPT incurs a cost of at most $n \cdot \frac{n-1}{2}$
 - in serving σ , OPT incurs a cost of at most $m \cdot \frac{n+1}{2}$
- $OPT(\sigma) \leq \frac{m \cdot (n+1)}{2} + \frac{n \cdot (n-1)}{2}$

Lower bound for any deterministic online algorithm

Proof ²:

- $OPT(\sigma) \leq \frac{m \cdot (n+1)}{2} + \frac{n \cdot (n-1)}{2}$
- $OPT(\sigma) \leq \frac{m \cdot (n+1)}{2}$ (for $m \gg n$)

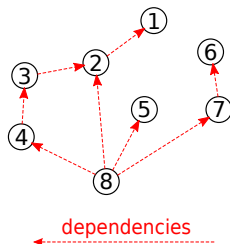
$$A(\sigma) \geq \frac{2 \cdot n}{n+1} \cdot OPT(\sigma)$$

We conclude the proof, since the RHS in the above inequality approaches 2 as n increases and the bound holds for any n .

² $A(\sigma) = m \cdot n$

Packet Classification

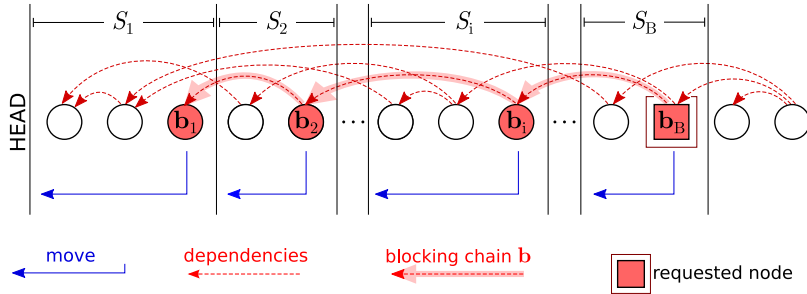
- Packet classification and firewall rule lists cannot be arbitrarily rearranged
- Each rule has a priority over other rules: Can be represented as a DAG



N	Proto	Src IP	Dst IP	Src Port	Dst Port	Action
1	TCP	10.12.12.0/24	20.0.0.1/32	ANY	80	DENY
2	TCP	0.0.0.0/0	20.0.0.1/32	ANY	80	ACCEPT
3	IP	0.0.0.0/0	20.0.0.1/32			DENY
4	UDP	0.0.0.0/0	0.0.0.0/0	1000-2000	1000-2000	ACCEPT
5	UDP	20.0.0.0/24	10.0.10.0/24	ANY	3306	ACCEPT
6	TCP	10.12.12.0/24	0.0.0.0/0	21	21	DENY
7	IP	10.0.0.0/16	20.0.0.0/20			ACCEPT
8	IP	0.0.0.0/0	0.0.0.0/0			DENY

Move-Recursively-Forward

- The idea is to move to front, **but**, only until the priorities are not violated
 - Move-recursively-forward
 - 4-competitive [3]



Move-Recursively-Forward

Static-List MRF Efficuts
Hypercuts Hicuts Cutsplit

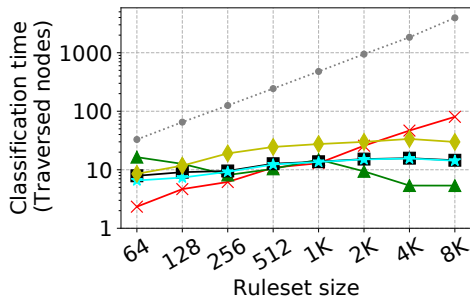


Figure: Classification time under high-locality in traffic

Move-Recursively-Forward

Static-List MRF Efficuts
Hypercuts Hicuts Cutsplit

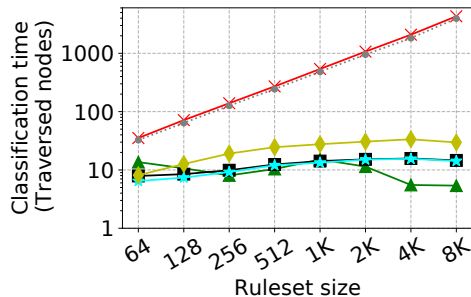


Figure: Classification time under low-locality in traffic

Move-Recursively-Forward

- Traffic locality can improve classification time, motivating the need for self-adjusting datastructures in packet classification
- Can we do better than 4-competitive?
- Can we generalize to tree-based data structures e.g., splay trees?

References

- [1] Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized efficiency of list update and paging rules. Commun. ACM 28, 2 (Feb. 1985), 202–208. URL: <https://doi.org/10.1145/2786.2793>
- [2] Albers S. Competitive online algorithms. BRICS; 1996 Sep. URL: <https://imada.sdu.dk/~joan/dm19/albers.pdf>
- [3] V. Addanki, M. Pacut, A. Pourdamghani, G. Rétvári, S. Schmid and J. Vanerio, "Self-Adjusting Partially Ordered Lists," IEEE INFOCOM 2023 - IEEE Conference on Computer Communications, New York City, NY, USA, 2023, pp. 1-10, URL: <https://ieeexplore.ieee.org/abstract/document/10228937>.
- [4] Azar Y, Lewkowicz S, Suriyanarayana V. List update with prediction. In Proceedings of the AAAI Conference on Artificial Intelligence 2025 Apr 11 (Vol. 39, No. 15, pp. 15436-15444). URL: <https://doi.org/10.1609/aaai.v39i15.33694>

The End