On list update with locality of reference ()

Susanne Albers and Sonja Lauer, 2015

Juan Vanerio

Communication Technologies Group
Faculty of Computer Science
University of Vienna

June 7, 2021

Table of Contents

Introduction

Introduction

Introduction

- **List Update Problem**
- A fundamental online algorithmic problem.
- Early approaches based on stochastic analysis.
- Last 30 years of research focused on *competitive analysis*.

$$A(\sigma) \leq c \cdot OPT(\sigma) + \alpha$$

  - Arbitrary request sequences (σ).
  - (Usually) Oblivious adversary model generates sequence.
- **Empirical** performance is **better** than suggested by competitive analysis.
- Reason: In practice sequences exhibit **locality of reference**
  - A small subset of items is referenced at any point in time.

Introduction

- **List Update Problem**
- Maintain a set of items L as linear list.
- Serve a sequence of item requests σ
  - Online: No knowledge of future requests.
  - $\sigma(t) \in L$
- **Standard Cost Model**:
  - Accessing the i-th item has cost i.
  - *Free exchanges*: The accessed item can be moved to another position at no cost.
  - *Paid exchanges*: All other exchanges of consecutive items has cost 1.
  - $A(\sigma)$: Total cost incurred by algorithm A in serving σ.

Important Algorithms

- **OPT**: The ideal offline optimum algorithm to compare against.
- **Move-To-Front(MTF)**: $c = 2$: Move the requested item to the front of the list.
- **TimeStamp(TS)**: $c = 1.62$: Place requested item x in front of the first item preceding x that was requested at most once since the last request to x. On error don't move.
- **BIT(BIT)**: $c = 1.75$: Initialize randomly a bit $b(x)$ for each $x \in L$. On access, complement the item's bit. If value changes to 1, move item to front.
- **COMB(COMB)**: $c = 1.6$: With probability 4/5 use BIT, else TS.

Model of Locality of Reference

Model of Locality of Reference

Concept

- **Intuition**: A sequence that references a small subset of items exhibits high locality.
- **Intuition**: If σ requests the same item many times in a row (high locality) then moving the item to the front becomes a very good strategy.
    - How many times is not relevant for the cost.
- When a different item is requested (a **change**), all algorithms should rearrange their lists to be competent.
- This paper proposes a formalization of the locality concept addressing the intuitive elements.

Definitions

- **Run**: Subsequence of requests to the same item.
  - **Short**: A single request.
  - **Long**: More than two requests.
    - **Prefixed**: Preceded by one or more short runs.
    - **Independent**: Not prefixed.
    - **Change**: If the previous log run reference a different item. The first long run is (usually) also a change.
- **Intuition: High degree of locality if there are relatively many long runs!**

Parameters

For any subsequence $\sigma' \subseteq \sigma$ let:

Parameter	Count
$r(\sigma')$	Runs
$s(\sigma')$	Short runs
$l(\sigma')$	Long runs
$l_p(\sigma')$	Prefixed long runs
$l_i(\sigma')$	Independent long runs
$l_c(\sigma')$	Long run changes
$f_b(\sigma')$	1 if item requested first is in front.
$f_e(\sigma')$	1 if σ' ends with short run and item is not in front.

Properties

- $r(\sigma') = s(\sigma') + l(\sigma')$
- $l(\sigma') = l_i(\sigma') + l_p(\sigma')$
- $l_i(\sigma') \leq l_c(\sigma') \leq l(\sigma')$

Example

independent long run — short run — prefixed long run — prefixed long run

$$\sigma_{xy} = xxyyyxyxyxxxxyxxx$$

long run change — NO change!

$$r(\sigma_{xy}) = 9 \rightarrow s(\sigma_{xy}) = 5$$

$$r(\sigma_{xy}) = 9 \rightarrow l(\sigma_{xy}) = 4$$

$$l(\sigma_{xy}) = 4 \rightarrow l_i(\sigma_{xy}) = 2$$

$$l(\sigma_{xy}) = 4 \rightarrow l_p(\sigma_{xy}) = 2$$

$$l(\sigma_{xy}) = 4 \rightarrow l_c(\sigma_{xy}) = 3$$

Initial config: $y|x$

$$\rightarrow f_b(\sigma_{xy}) = 0$$

$$\rightarrow f_e(\sigma_{xy}) = 0$$

λ-locality

- A class Σ of requests sequences has $\lambda-$ *locality* if $\forall\,\sigma \in \Sigma$,

$$\frac{l_c(\sigma)}{r(\sigma)} \geq \lambda$$

- **The number of long run changes represent at least a fraction λ of all runs**.
- $0 \leq \lambda \leq 1$
  - If a sequence consists of long runs only, $\lambda = 1$
  - $\lambda = 0$ if there are no long runs.

Find new bounds as a function of λ for the competitiveness of online algorithms

Analysis

Analysis

Basic Cost Analysis

Overview:

- Given an algorithm A, decompose its cost over all subsequences comprehended by a pair of items.
- Draw relationships between the projected cost and the cost of subsequences.
- Bound the cost for each phase ended by a long run.
- Add all up to get the resulting cost.
- Compare against OPT to get the competitiveness.

The cost A of an algorithm on σ can be evaluated novelty: incorporate paid exchanges

Basic Cost Analysis

At time t a request i made for item $\sigma(t)$.

Let:

- $A_x(t,\sigma) = 1$ if $x < \sigma(t)$.
- $A_p(t,\sigma)$ the number of paid exchanges performed by A.

$$
\begin{aligned}
A(\sigma) &= \sum_{t=1}^{|\sigma|} \left(1 + A_p(t,\sigma) + \sum_{x \in L} A_x(t,\sigma) \right) \\
&= |\sigma| + \sum_{t=1}^{|\sigma|} A_p(t,\sigma) + \sum_{x \in L} \sum_{t=1}^{|\sigma|} A_x(t,\sigma) \\
&= |\sigma| + \sum_{t=1}^{|\sigma|} A_p(t,\sigma) + \sum_{\substack{\{x,y\} \subseteq L \\ x \neq y}} \sum_{\substack{t: \\ \sigma(t) \in \{x,y\}}} A_x(t,\sigma) + A_y(t,\sigma)
\end{aligned}
$$

Basic Cost Analysis

Now let:

- $A_{p,xy}(\sigma)$ the number of paid exchanges performed by A to change the relative order between x and y while serving σ.
- $A_{xy}(\sigma) = \sum_{\substack{\{x,y\}\subseteq L \\ x\neq y}} A_{p,xy}(\sigma) + \sum_{\substack{t: \\ \sigma(t)\in\{x,y\}}} (A_x(t,\sigma) + A_y(t,\sigma))$ is the cost of A projected over $\{x, y\}$.

Then,

$$A(\sigma) = |\sigma| + \overbrace{\sum_{\substack{\{x,y\}\subseteq L \\ x\neq y}} A_{xy}(\sigma)}^{\text{Partial cost model}} \qquad (1)$$

Basic Cost Analysis - Projection

Let σ_{xy} be the subsequence obtained by projecting σ on $\{x, y\}$.

- Most algorithms satisfy $A_{xy}(\sigma) = A(\sigma_{xy})$
- $OPT_{xy}(\sigma) \geq OPT(\sigma_{xy})$

Decomposes σ_{xy} in p_{xy} phases ending with a long run (except perhaps the last phase).

- Let's $\pi(i)$ be the i-th phase.
- WLOG $\pi(i)$ starts with x.
- There are only two possible structures:
  - $$(xy)^k x^l \quad k \geq 0, \quad l \geq 1 \tag{2}$$
  - $$(xy)^k y^l \quad k \geq 1, \quad l \geq 0 \tag{3}$$

Basic Cost Analysis - Final observations

- Definitions f_b and f_e apply for any σ_{xy}
- Eventually they values must be added over all item pairs.
- Their added values don't depend on $|\sigma|$
- They will allow finding bounds when considering the first and last phases respectively.
- On σ_{xy} a long run change occurs only if phase structure is $(xy)^k x^l$ with $l \geq 2$.

Optimal Offline Algorithm

OPT for two item lists

- Move to front only on the first request of a long run.

Claim

$OPT_{xy}(\sigma) \geq OPT(\sigma_{xy}) \ \forall\ x, y \in L, x \neq y$

Proof:

- Assume we serve σ_{xy} with algorithm algorithm O' such that:
  - O' changes the relative order of x and y only when OPT does so when servicing σ over L.
- By design O' incurs (partial) cost $OPT_{xy}(\sigma)$
- By definition its cost can't be lower than the optimum for the sequence being served.
- Then $OPT_{xy}(\sigma) \geq OPT(\sigma_{xy})$ □

Optimal Offline Algorithm

Lemma 1

$OPT(\sigma) \geq \frac{1}{2}(r(\sigma) + l_c(\sigma) + f_e(\sigma)) - f_b(\sigma) + |\sigma|$

Proof:

- Using eq.1 and the Claim we can find a lower bound just by adding the item pairs decomposition (σ_{xy}).
- The idea is to study many cases. There are actually a lot.

Case 1 $|\sigma| = 1 \Rightarrow r = 1, l_c = 1$

Case 1.1 x in front of y ("$x|y$") $\Rightarrow OPT = 0, f_b = 1, f_e = 0$. Direct.

Case 1.2 $y|x \Rightarrow OPT = 1, f_b = 0, f_e = 1$. Direct.

Case 2 $|\sigma| > 1$. Split in phases ending in long runs.

- Note: Long run change occurs only if structure is $(xy)^k x^l$ with $l \geq 2$.

Optimal Offline Algorithm

Consider then that there are many phases.

- If $f_e(\sigma_{xy}) = 1$ then $p_{xy} > 1$.
  1. Phase starts with request to x and finishes to request to x.
  2. $l_c(\pi(p_{xy})) = l(\pi(p_{xy})) = 0$
  3. $r = 2k + 1$
  4. $$OPT = k + 1 = \frac{1}{2}(r + l_c + f_e) \tag{4}$$
- else if $f_e(\sigma_{xy}) = 0$ or just $\pi(i)$ for $1 < i \leq p_{xy}$:
  - Initial list: $y|x$
  - If structure is $(xy)^k x^l \Rightarrow r = 2k + 1 \Rightarrow OPT = k + 1 = \frac{1}{2}(r + l_c)$
  - If structure is $(xy)^k y^l \Rightarrow l_c = 0, r = 2k \Rightarrow OPT = k = \frac{1}{2}(r + l_c)$
- first phase
  - If $y|x \Rightarrow f_b = 0 \Rightarrow$ same as before $\Rightarrow OPT = k \geq \frac{1}{2}(r + l_c - f_b)$
  - If $x|y \Rightarrow f_b = 1 \Rightarrow OPT = \lfloor r/2 \rfloor \geq \frac{1}{2}(r + l_c - f_b)$
- Add up over all phases and pairs (as per 1). □

Move-to-Front - Claim

- Move the requested item to the front of the list.

Claim

$MTF_{xy}(\sigma) = MTF(\sigma_{xy}) \; \forall \; x, y \in L, x \neq y$

Proof:

- On both the original and the xy-pair lists, x precedes y iif the last request to either of them was to x.
- Say σ has $|\sigma_{xy}|$ requests to $\{x, y\}$
- Any request i s.t $1 \leq i \leq |\sigma_{xy}|$ contributes 1 to $MTF_{xy}(\sigma)$ iif $MTF(\sigma_{xy}) = 1$
- There are no paid exchanges.
- $MTF_{xy}(\sigma) = MTF(\sigma_{xy})$ □

Move-to-Front Algorithm - Lemma

Lemma 2

$MTF(\sigma) \leq r(\sigma) - f_b(\sigma) + |\sigma|$

Proof:

- Using eq.1 and the Claim we can find an upper bound just by adding the item pairs decomposition (σ_{xy}).

1. On σ_{xy}, the first request of each run the referenced item is moved to the front with cost 1.
2. Exception is the first run if item was already in front ($f_b = 1$)
3. $\Rightarrow MTF(\sigma_{xy}) = r - f_b$
4. Add up over all phases and pairs (as per 1). □

Move-to-Front Algorithm

Theorem 1

$$\frac{MTF(\sigma)}{OPT(\sigma)} \leq \frac{2 + 2\alpha(\sigma)}{1 + 2\alpha(\sigma) + \beta(\sigma)}$$

Where:

- $\alpha(\sigma) = \frac{|\sigma| - (f_b(\sigma))}{r(\sigma)}$ and $\beta(\sigma) = \frac{l_c(\sigma)}{r(\sigma)}$

Corollary 1

If σ has λ-locality, then

$$\frac{MTF(\sigma)}{OPT(\sigma)} \leq \frac{2}{1 + \lambda}$$

This results in a much better performance guarantee under high locality!

BIT - Claims

- Initialize randomly a bit $b(x)$ for each $x \in L$. On access, complement the item's bit. If value changes to 1, move item to front.
- Note: Assume $y|x$ while servicing σ_{xy} at time $t-1$. Then $\mathbb{E}[BIT(\sigma_{xy}(t))] = 1/2$
- Note: Assume last requests were xyx. Then $\mathbb{E}[BIT(\sigma_{xy}(t))|\sigma_{xy}(t) = x] = 1/4$ and $\mathbb{E}[BIT(\sigma_{xy}(t))|\sigma_{xy}(t) = y] = 3/4$.

Claim

$BIT_{xy}(\sigma) = BIT(\sigma_{xy}) \ \forall \ x, y \in L, x \neq y$

Proof:

- On the i-th request to x or y, x precedes y in the original list iif it does on the xy-pair list.
- The rest of the argument is analogous to that of MTF.
- $BIT_{xy}(\sigma) = BIT(\sigma_{xy})$ □

BIT - Lemma

Lemma 2

(On expectation) $BIT(\sigma) \leq \frac{3}{4}r(\sigma) + \frac{1}{4}l(\sigma) + \frac{1}{2}l_i(\sigma) + \frac{1}{4}f_e(\sigma) + |\sigma|$

Proof Outline:

- Using eq.1 and the Claim we can find an upper bound just by adding the item pairs decomposition (σ_{xy}).
- Decompose in phases ending in long runs. Term for f_e is due to last phase only.
- Non-last phases satisfy: $BIT(\sigma) \leq \frac{3}{4}r(\sigma) + \frac{1}{4}l(\sigma) + \frac{1}{2}l_i(\sigma)$

1. A generic phase $\pi(i)$ starts with $y|x$.
2. Then first request costs 1, and the second has expected cost 1/2(Claim note 1).
3. Any further short runs in the phase have expected value 3/4(Claim note 2).
4. The long run (if any) has worst case cost 1 $(3/4 + 1/4$, Claim note 2).
5. Go through the initial and final cases to introduce adjustments.
6. Add up over all phases and pairs (as per 1).

□

BIT - Theorem

Theorem 3

$$\frac{BIT(\sigma)}{OPT(\sigma)} \leq \frac{1.5 + 2\alpha(\sigma) + \delta(\sigma)}{1 + 2\alpha(\sigma) + \beta(\sigma)}$$

Where:

- $\alpha(\sigma) = \frac{|\sigma|-(f_b(\sigma))}{r(\sigma)}$, $\beta(\sigma) = \frac{l_c(\sigma)}{r(\sigma)}$ and $\delta(\sigma) = \frac{l(\sigma)/2+l_i(\sigma)+2f_b(\sigma)}{r(\sigma)}$.

Corollary 2

If σ has λ-locality, then

$$\frac{BIT(\sigma)}{OPT(\sigma)} \leq \min\left\{1.75, \frac{2}{1+\lambda}\right\}$$

This results in a better performance guarantee for $\lambda > 1/3$

Other Results

- No improvement in the upper bounds for TIMESTAMP or COMB.
- They can't exploit locality.
- Empirical competitiveness against pairwise offline optimum confirmed expected results.
  - Particularly accurate results achieved with MTF.

Conclusions

Conclusions

Conclusions

- Useful model of locality.
  - Mainly based in detailed characterization of runs in σ.
  - Captures intuition.
  - Allows new analysis of online algorithms.
- Found new, better bounds for MTF and BIT under locality.
  - Particularily for MTF that approaches OPT for large λ
- The paper also provides an experimental study backing the results.

Thank You

Thank You for listening!
Questions?