Chapter 9
Online Algorithms

Algorithm Theory
WS 2015/16

Fabian Kuhn

Competitive Ratio

- Let's again consider optimization problems
  - For simplicity, assume, we have a minimization problem

Optimal offline solution $\mathbf{OPT}(I)$:

- Best objective value that an offline algorithm can achieve for a given input sequence I

Online solution $\mathbf{ALG}(I)$:

- Objective value achieved by an online algorithm ALG on I

Competitive Ratio: An algorithm has competitive ratio $c \geq 1$ if

$$\mathbf{ALG}(I) \leq c \cdot \mathbf{OPT}(I) + \alpha.$$

- If $\alpha = 0$, we say that ALG is strictly c-competitive.

Self-Adjusting Lists

- Linked lists are often inefficient
  - Cost of accessing an item at position i is linear in i
- But, linked lists are extremely simple
  - And therefore nevertheless interesting
- Can we at least improve the behavior of linked lists?
- In practical applications, not all items are accessed equally often and not equally distributed over time
  - The same items might be used several times over a short period of time
- **Idea:** rearrange list after accesses to optimize the structure for future accesses
- **Problem:** We don't know the future accesses
  - The list rearrangement problems is an online problem!

Model

- Only find operations (i.e., access some item)
  - Let's ignore insert and delete operations
  - Results can be generalized to cover insertions and deletions

Cost Model:

- Accessing item at position i costs i
- The only operation allowed for rearranging the list is swapping two adjacent list items
- Swapping any two adjacent items costs 1

Rearranging The List

Frequency Count (FC):

- For each item keep a count of how many times it was accessed
- Keep items in non-increasing order of these counts
- After accessing an item, increase its count and move it forward past items with smaller count

Move-To-Front (MTF):

- Whenever an item is accessed, move it all the way to the front

Transpose (TR):

- After accessing an item, swap it with its predecessor

Cost

Cost when accessing item at position i:

- Frequency Count (FC): between i and $2i - 1$
- Move-To-Front (MTF): $2i - 1$
- Transpose (TR): $i + 1$

Random Accesses:

- If each item x has an access probability p_x and the items are accessed independently at random using these probabilities, FC and TR are asymptotically optimal

Real access patterns are not random, TR usually behaves badly and the much simpler MTF often beats FC

Move-To-Front

- We will see that MTF is competitive
- To analyze MTF we need competitive analysis and amortized analysis

Operation k:

- Assume, the operation accesses item x at position i
- c_k: actual cost of the MTF algorithm

$$c_k = 2i - 1$$

- a_k: amortized cost of the MTF algorithm
- c_k^*: actual cost of an optimal offline strategy
  - Let's call the optimal offline strategy OPT

Potential Function

- For the analysis, we think of running the MTF and OPT at the same time
- The state of the system is determined by the two lists of MTF and OPT
- Similarly to amortized analysis for data structures, we use a potential function which maps the system state to a real number
- If the MTF list and the list of OPT are similar, the actual cost of both algorithms for most requests is roughly the same
- If the lists are very different, the costs can be very different and the potential function should have a large value to be able to compensate for the potentially high cost difference
- We therefore use a **potential function** which measures the **difference between** the **MTF list** and the **optimal offline list**

Potential Function

Potential Function Φ_k:

- **Inversion:** pair of items x and y such that x precedes y in one list and y precedes x in the other list
- ***Twice*** the number of ***inversions*** between the lists of MTF and OPT after the first k operations
- Measure for the difference between the lists after k operations

Initially, the two lists are identical: $\boldsymbol{\Phi_0 = 0}$

For all k, it holds that $\boldsymbol{0 \leq \Phi_k \leq 2 \cdot \binom{n}{2} = n(n-1)}$

Potential Function

Potential Function Φ_k:

- **Inversion:** pair of items x and y such that x precedes y in one list and y precedes x in the other list
- ***Twice*** the number of ***inversions*** between the lists of MTF and OPT after the first k operations
- Measure for the difference between the lists after k operations

To show that MTF is α-competitive, we will show that

$$\forall k: \; a_k = c_k + \Phi_k - \Phi_{k-1} \leq \alpha \cdot c_k^*$$

Competitive Analysis

Theorem: MTF is 4-competitive.

Proof:

- Need that $a_k = c_k + \Phi_k - \Phi_{k-1} \leq 4c_k^*$
- Position of x in list of OPT: i^*
- Number of swaps of OPT: s^*
- In MTF list, position of x is changed w.r.t. to the $i-1$ preceding items (nothing else is changed)
- For each of these items, either an inversion is created or one is destroyed (before the s^* swaps of OPT)
- Number of new inversions (before OPT's swaps) $\leq i^* - 1$:
  - Before op. k, only $i^* - 1$ items are before x in OPT's list
  - With all other items, x is ordered the same as in OPT's list after moving it to the front

Competitive Analysis

Theorem: MTF is 4-competitive.

Proof:

- Need that $a_k = c_k + \Phi_k - \Phi_{k-1} \leq 4c_k^*$
- $c_k = 2i - 1, \quad c_k^* = i^* + s^*$
- Number of inversions created: $\leq i^* - 1 + s^*$
- Number of inversions destroyed: $\geq i - i^*$

Competitive Analysis

Theorem: MTF is 4-competitive.

Proof:

- Need that $a_k = c_k + \Phi_k - \Phi_{k-1} \leq 4c_k^*$
- $c_k = 2i - 1, \quad c_k^* = i^* + s^*$
- Number of inversions created: $\leq i^* - 1 + s^*$
- Number of inversions destroyed: $\geq i - i^*$