Introduction

In the following lecture we will cover the following topics:

- Admission Control - prove that the exponent algorithm is $O(\log(D))$ competitive.
- Admission Control on the line - lower bounds
- Interval Scheduling - the Classify and Randomly Select algorithm has an expected competitive ratio of $O(\log D)$

1 Admission Control

Reminder: In the Admission Control problem we are given a graph $G = (V, E)$, capacities over the edges, $c : E \rightarrow \mathbb{R}^+$ and a sequence of requests. A request is in the form $(s_i, t_i, p_i, b_i)$ where:

- $s_i$ - source
- $t_i$ - target
- $p_i$ - price/demand/bandwidth required
- $b_i$ - benefit for servicing the request

Each request can be serviced or can be rejected and the goal is to maximize the benefit while maintaining a feasible solution: $\forall e \sum_{i \in Q_i} p_i \leq c(e)$.

In the previous lecture we saw the exponent algorithm for this problem: “Accept a request $i$ on a path $Q_i$ if $\sum_{e \in Q_i} X_e \leq D$, else, reject.”

Where:

$$X_e = (2D + 2)^{l_e} - 1$$

**Theorem 1.1.** The exponent algorithm is feasible

**Proof.** Assuming $p_i \leq \frac{1}{\log(2D+2)}$, overflow on edge $e$ may occur only if $l_e > 1 - \frac{1}{\log(2D+2)}$ but then:

$$X_e > (2D + 2)^{1 - \frac{1}{\log(2D+2)}} - 1$$

$$= \frac{2D + 2}{2} - 1$$

$$= D$$

But this implies that the algorithm would not have used this edge. \qed
In order to prove the competitive ratio, we are going to assume that:

- (mandatory) \( p_i \leq \frac{c(e)}{\log(2D+2)} \) \( (D \leq n - 1 \) is the maximal path length). 
- \( b_i = p_i \) - we try to maximize the throughput
- \( c(e) = 1 \)

**Theorem 1.2.** The exponent algorithm for Admission Control is \( O(\log D) \) competitive.

**Proof.** We will use the following lemmas.

**Lemma 1.3.** \( \sum_{i \in A} p_i \geq \sum_{e \in E} X_e \)

**Lemma 1.4.** \( \frac{1}{D} \sum_{i \in RA} p_i \leq \sum_{e \in E} X_e \) (RA are the requests we rejected and OPT accepted)

Lemmas 1.3, 1.4 prove the theorem:

\[
\sum_{i \in OPT} p_i \leq \sum_{i \in A} p_i + \sum_{i \in RA} p_i \\
\leq \sum_{i \in A} p_i + \frac{1}{D} \sum_{e \in E} X_e \\
\leq \sum_{i \in A} p_i + \frac{2D \cdot \log (2D + 2)}{D} \sum_{i \in A} p_i \\
= \sum_{i \in A} p_i \cdot \left( 1 + 2 \log (2D + 2) \right) \\
= O(\log D) \cdot \sum_{i \in A} p_i
\]

\[\square\]
Proof. Lemma 1.3
At time 0, \( \forall e \in E: X_e = 0 \) and the gain is 0.
We will show that if the algorithm accepts the \( i^{th} \) request, then,
\[
2D \cdot \log (2D + 2)p_i \geq \sum_{e \in E} \Delta X_e = \sum_{e \in Q_i} \Delta X_e
\]
We will look at the change in the exponent function, \( \sum_{e \in Q_i} \Delta X_e \):
\[
\sum_{e \in Q_i} \Delta X_e = \sum_{e \in Q_i} (2D + 2)^{\ell_e} \left[ (2D + 2)^p_i - 1 \right]
\]
\[
a = 2^{\log_2 a} \downarrow \sum_{e \in Q_i} (2D + 2)^{\ell_e} \left( 2 \log (2D + 2)p_i - 1 \right)
\]
\[
0 \leq x \leq 1 \Rightarrow 2^x - 1 \leq x
\]
\[
\leq \sum_{e \in Q_i} (2D + 2)^{\ell_e} \log (2D + 2)p_i
\]
\[
= \log (2D + 2)p_i \sum_{e \in Q_i} (2D + 2)^{\ell_e} - 1 + 1
\]
accepting the request
\[
\leq \log (2D + 2)p_i \cdot \left( \sum_{e \in Q_i} \ell_e + \sum_{e \in Q_i} \ell_e \right)
\]
\[
= 2D \log (2D + 2)p_i
\]

Proof. Lemma 1.4
\( Q_i^* \) - The path that OPT uses for request \( i \).
If the algorithm rejects request, specifically (because every path > \( D \)),
\[
D < \sum_{e \in Q_i^*} X_e
\]
Multiplying by \( p_i \),
\[
D \cdot p_i \leq p_i \sum_{e \in Q_i^*} X_e
\]
Summing over all \( i \in RA \)
\[
D \sum_{i \in RA} p_i \leq \sum_{i \in RA} p_i \cdot \sum_{e \in Q_i^*} X_e = \sum_{i \in RA} \sum_{e \in Q_i^*} X_e p_i
\]
By replacing the order of summation:

\[ = \sum_{e \in E} X_e \sum_{i \in Q_i} p_i \leq \sum_{i \in RA} \sum_{e \in Q_i^*} X_e \cdot 1 \]

where the last inequality follows since the weight that OPT puts on a single edge is \( \leq 1 \).

Remarks:

1. Variation: A request arrives with a path \( Q_i \). The algorithm can choose whether to accept or to reject. If \( \sum_{e \in Q_i} X_e \leq D \), accept, else, reject. The proof remains the same.

2. The request does not arrive with a path. The proof is valid without assuming that the algorithm chooses the shortest path, but just a path of weight \( \leq D \)

3. If \( p_i \leq b_i \leq \mu \cdot p_i \) (The benefit per cost unit is \( \in [1, \mu] \)), i.e. the ratio between a maximal benefit to a minimal benefit per bandwidth unit is \( \leq \mu \), then, the competitive ratio is \( O(\log D + \log \mu) \) using a more technical yet similar proof.

2 Lower Bounds

In the previous lecture we showed a lower bound of \( \Omega(\log \mu) \) for \( D = 1 \). We will now show a lower bound of \( \Omega(\log D) \) for any \( D \) and see that the online algorithm is optimal (up to a constant ratio).

**Theorem 2.1.** Any online algorithm for the Admission Control problem is at least \( \Omega(\log D) \)-competitive, "Even" in case of a line of length \( D \), Even if Random, Even if Fractional\(^1\).

\(^1\)Fractional algorithm - the algorithm can service even just a part of a request, or split it into several requests. Its benefit is proportional to the size it serviced.
In order to get a feasible solution, \( \sum \frac{X_i}{2^j} \leq 1 \), however, if we stop at phase \( j \), then \( \text{OPT} = 2^j \) (it takes only the requests in the last phase), so:

\[
S_j = \frac{\text{ON}}{\text{OPT}} = \frac{\sum_{i=0}^{j} X_i}{2^j}
\]

Summing over all possible \( S_j \) (all the possible phases where the sequence ends):

\[
\sum_{j=0}^{\log D} S_j = \sum_{j=0}^{\log D} \sum_{i=0}^{j} X_i \cdot 2^{-j} = \sum_{0 \leq i \leq j \leq \log D} X_i \cdot 2^{-j}
\]

replacing the order of summation:

\[
= \sum_{i=0}^{\log D} X_i \sum_{j \geq i} 2^{-j}
\]

by sum of geometric series:

\[
\leq \sum_{i=0}^{\log D} X_i \cdot 2 \cdot 2^{-i} = 2 \sum_{i=0}^{\log D} \frac{X_i}{2^i} \leq 2
\]

This implies that there exist \( j \) s.t. \( S_j \leq \frac{2}{\log D + 1} \).

For this \( j \) the sequence ends and we get a \( \Omega(\log D) \)-competitive ratio.

**Theorem 2.2.** If the requests have a demand (bandwidth) of \( \frac{1}{k} \), then, any deterministic algorithm is \( \Omega(D^\frac{1}{k}) \)-competitive, “Even” on a simple path.

This theorem shows that the assumption \( p_i \leq \frac{c(e)}{\log(2D+2)} \) is indeed mandatory.

**Proof.**

- \( k = 1 \):

  The sequence starts with a request of length \( D ([0,D]) \) which \( ON \) must accept, and blocks the network (because the capacity is 1), then, we give \( D \) requests of length 1 \(([0,1],[1,2],\ldots) \) that \( \text{OPT} \) accepts.

\[
\text{OPT} = D, \text{ON} = 1 \Rightarrow \Omega(D)
\]
• $k = 2$:
  Phase 1: 2 requests of length $D$ ([0, $D$]) and demand $\frac{1}{2}$. $ON$ accepts at least one.
  Phase 2: We give a prefix of the following sequence (two requests for every interval):
  
  $[0, D^\frac{1}{2}], [D^\frac{1}{2}, 2D^\frac{1}{2}], \ldots, [j \cdot D^\frac{1}{2}, (j + 1)D^\frac{1}{2}], \ldots, [D - D^\frac{1}{2}, D]$

  If $ON$ rejects all of these requests, then $OPT$ accepts all:
  \[
  OPT = D^\frac{1}{2}, ON = \frac{1}{2} \Rightarrow \frac{OPT}{ON} = \Omega(D^\frac{1}{2})
  \]

  Therefore, $ON$ must accept a request at some point.
  Assume it accepts the $j^{th}$:
  
  $[j \cdot D^\frac{1}{2}, (j + 1)D^\frac{1}{2}]$

  At this point we continue by giving the following requests (two requests for every interval):
  
  $[j \cdot D^\frac{1}{2}, j \cdot D^\frac{1}{2} + 1], [j \cdot D^\frac{1}{2} + 1, j \cdot D^\frac{1}{2} + 2], \ldots, [(j + 1)D^\frac{1}{2} - 1, (j + 1)D^\frac{1}{2}]$

  $OPT$ accepts all of them and we get: $ON = 1, OPT = D^\frac{1}{2} \Rightarrow \Omega(D^\frac{1}{2})$

• for any $k$:
  Each phase, we split the current interval into $D^\frac{1}{k}$ segments. We assume that $ON$ picks one (otherwise, we are done), and on the next phase, we split that segment into $D^\frac{1}{k}$ segments, and so on. We can continue doing this process for $k$ phases and on the last phase, $ON$ blocks this interval with a single request and $OPT$ gets a sequence of requests of length 1 and accepts them $\Rightarrow \Omega(D^\frac{1}{k})$.

\[\square\]

### 3 Interval Scheduling

We have an interval [0, $D$], all request $[a, b] (a < b \in N)$ have a demand=1 and the capacity is 1.

A feasible solution - disjoint paths (no overlapping requests).

A deterministic algorithm will be $\Omega(D)$-competitive: we will give requests [0, $D$] until it accepts one, and the interval is blocked, and then the sequence continues with [0, 1], [1, 2] … and $OPT$ will accept all.

**Definition** (bounded input). The input has only requests of length $2^{i-1} \leq l \leq 2^i$.

**Theorem 3.1.** For the bounded input case, the greedy algorithm (“accept when you can”) is 3-competitive.

**Proof.** for every segment $ON$ accept, $OPT$ can accept up to 3 segments instead. \[\square\]
Classify and Randomly Select Algorithm

The algorithm:

- **Classify**: Split the input into \( \log D \) sets, by length, i.e. the \( i^{th} \) set are the requests of interval length \( 2^{i-1} \) to \( 2^i \).

- **Randomly**: Choose \( 1 \leq i \leq \log D (i \in \mathbb{N}) \) randomly and uniformly.

- **Select**: Accept only requests from the \( i^{th} \) set.

**Theorem 3.2.** The Classify and Randomly Select(C&RS) algorithm has an expected competitive ratio of \( O(\log D) \).

**Proof.** Denote by \( B_i \) the number of requests accepted by \( OPT \) in the \( i^{th} \) set.

Using theorem 3.1,

\[
E(C&RS) \geq \frac{1}{\log D} \left( \frac{1}{3}B_1 + \frac{1}{3}B_2 + \cdots + \frac{1}{3}B_{\log D} \right)
\]

\[
= \frac{1}{3 \log D} (B_1 + B_2 + \cdots + B \log D)
\]

\[
= \frac{1}{3 \log D} OPT
\]

Remark: In general, if we classify the input into \( k \) classes and in each one we have a \( C \)-competitive algorithm, then by using C&RS we get a \( O(kC) \)-competitive algorithm.