**Online Algorithms** 21/12/2011

# Lecture Notes 8: Benefit problems and Admission Control

*Professor: Yossi Azar* *Scribe:Omri Zomet*

## 1 Introduction

Until today, we have dealt with minimization problems, where graph edges had non-retricted capacity and every request was served. In this lecture will look at Benefit Problems, in which we maxmize the benefit. For example:

- One-way Trading - How to sell shares to gain maximum benefit
- Admission Control and Routing - Recalling the General Routing problem, now each request has benefit attribute ($b_i$). Given request i either return route $Q_i$ or reject. We would like to maximize $\sum_{i \in A} b_i$

## 2 Benefit Problems

In these problems, we maximize the benefit instead of minimizing the cost.
Competitive ratio is defined a bit different than before -
**Competitive ratio:** An algorithm A is called c-competitive if $A(\sigma) \geq \frac{1}{c}$ OPT($\sigma$)

### Example: One Way Trading (Shares)

You have $n$ (finite) shares that you like to sell. Each share has value$\in [1\, \mu]$.
At each moment you are offered a price to sell a share. Either you sell the share or not.
Oh - and if someone recognize you as the next Madoff - the share's value is 0, for good.
**Target:** Maximize the total selling price (benefit).

### Bad strategies

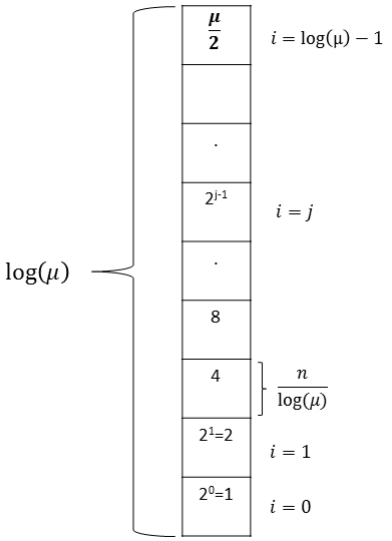
- Sell only at $\mu$ - ON will be offered to sell at price $(\mu - 1)$ for $x$ times. ON will never sell, thus earning nothing. OPT will earn $x \cdot (\mu - 1)$. Ratio - $\infty$
- Sell at any price - ON will be offered to sell at price of 1 for $n$ times and then to sell at price $\mu$ for $n$ times. ON will earn $n$ while OPT=$n \cdot \mu$. Ratio - $\mu$

### Exponent Algorithm

We will show an $O(\log \mu)$ algorithm. General idea - start selling at low prices and slowly jack up the price. When you are left with few shares - sell them at the highest price.

**Exponent Algorithm:** Let us assume $n \geq \log \mu$ (reasonable amount of shares). After selling $\frac{n}{\log \mu} \cdot i$ shares, minimum sell price is updated to $2^i$.

One way to think about the algorithm is as if the shares are divided to $\log \mu$ groups, each group of size $\frac{n}{\log \mu} (\geq 1)$. Then it assigns a minimum sell price for group i to be $2^i$, for $0 \leq i \leq \log \mu - 1$.




**Theorem:** Exponent algorithm is $O(\log \mu)$ competitive
**Proof:** Let $i^*$ be the highest full group - which means ON sold more than $\frac{n}{\log \mu} \cdot (i^* + 1)$ shares but less than $\frac{n}{\log \mu}(i^* + 2)$ shares.
- Notice that even if the first group ($i = 0$) isn't full - we get $ON(\sigma) = OPT(\sigma)$, since ON didn't reject anything (minimum price is 1). Hence $i^* \geq 0$ is defined well.

Now we compute how much OPT earned:
Following the definition of $i^*$, $ON \geq \frac{n}{\log \mu} \cdot 2^{i^*}$.
Let $RA$ be the requests where OPT sell but ON didn't. Then:

$$OPT(RA) \leq n \cdot 2^{i^*+1}$$

$$\begin{aligned} OPT(\sigma) &\leq ON(\sigma) + OPT(RA) && (1) \\ &\leq ON(\sigma) + n \cdot 2^{i^*+1} && (2) \\ &\leq ON(\sigma) + 2 \log \mu \cdot ON(\sigma) && (3) \\ &\leq (2 \cdot \log \mu + 1) \cdot ON(\sigma) && (4) \end{aligned}$$

This means that the exponent algorithm is $2 \cdot \log \mu + 1$ competitive

**Theorem:** Every algorithm is $\Omega(\log \mu)$ competitive.
**Proof:** We will give the following sequence of requests -

There are $\log \mu + 1$ iterations. In iteration $i$, $0 \leq i \leq \log \mu$, $n$ requests arrive at price $2^i$.
ON sells part of his shares at each phase $i$ - denoted by $X_i$:
Sells $X_0$ of his shares at price $2^0$
Sells $X_1$ of his shares at price $2^1$
Sells $X_2$ of his shares at price $2^2$
...
Sells $X_{\log \mu}$ of his shares at price $\mu$ ,where:

$$\sum_{i=0}^{\log \mu} X_i \leq 1$$

The basic idea is that we can stop this sequence at any iteration. If the sequence ends at iteration $i$, then OPT sells all the shares at the price of the iteration, which is $2^i$, and ON earns $\sum_{j \leq i} X_j \cdot 2^j$.

The ratio of each iteration:
1st level: $\frac{ON}{OPT} = X_0$
2nd level: $\frac{ON}{OPT} = \frac{X_0 \cdot 2^0 + X_1 \cdot 2^1}{2^1}$
3rd level: $\frac{ON}{OPT} = \frac{X_0 \cdot 2^0 + X_1 \cdot 2^1 + X_2 \cdot 2^2}{2^2}$
...
Last level: $\frac{ON}{OPT} = \frac{X_0 \cdot 2^0 + X_1 \cdot 2^1 + ... + X_{\log \mu + 1} \cdot 2^{\log \mu}}{2^{\log \mu}}$

At some point - one of these ratios will be small enough and that's where our sequence will end. Let us find the minimum ratio.
Let $r_k$ be the ratio at the k'th level:

$$r_k = \frac{ON}{OPT} = \frac{\sum_{i=0}^{k} X_i \cdot 2^i}{2^k}$$

To find the minimum - lets some up all the levels:

$$\begin{aligned} \sum_{k=0}^{\log \mu} r_k &= \sum_{k=0}^{\log \mu} \sum_{i=0}^{k} X_i \cdot \frac{2^i}{2^k} && (5) \\ &\leq 2 \cdot (X_0 + X_1 + ... + X_{\log \mu}) && (6) \\ &\leq 2 \cdot 1 && (7) \end{aligned}$$

(This is a sum of geometrical progressions, where each is $\leq 2$)
Hence exist a k such that $r_k \leq \frac{2}{\log \mu + 1}$ and we get competitive ratio $\geq \frac{\log \mu + 1}{2}$.

# 3 Addmission Control and Routing

We have a graph $G = (V, E)$
Capacity $C : E \to R^+$

Each request is a tuple $(s_i, t_i, p_i, b_i)$, where:
$s_i$ - source vertex
$t_i$ - target vertex
$p_i$ - bandwidth of request
$s_i$ - benefit of request
The aim is to maximize the total benefit: $\sum_{i \in A} b_i$ where $A$=Accepted requests
For each request i - either return path $Q_i$ or reject it if there's no such path.
We might not be able to find a path for each request since we must maintain feasibility:

- *Feasible Solution:* the accumulated bandwidth on each edge, over all paths, doesn't exceed the capacity of the edge. i.e.

$$\forall e \sum_{i|e \in Q_i} p_i \leq C(e)$$

Before continuing with the solution, we should mention that this problem generalize the Shares (One Way Trading) problem we've seen before - simply by looking at a graph consisting of one edge with capacity=n (each share is a request i with $p_i$=1 and sell price is $b_i$)

Following are some definitions and assumptions we need in order to show a logarithmic competitive ratio:
- Let $D \leq n-1$ be the maximum length of a path in the graph. If we allow any simple path without any restrictions, then $D = n-1$
- For each $e$, let $l_e$ be that accumulated load on e:

$$l_e = \frac{\sum_{i|e \in Q_i} p_i}{C(e)} \leq 1$$

Simplifying Assumption I: Let us assume $b_i = p_i$ (that is, we would like to maximize the throughput)

Simplifying Assumption II: Capacity of all edges is 1. $\forall e$, $C(e) = 1$

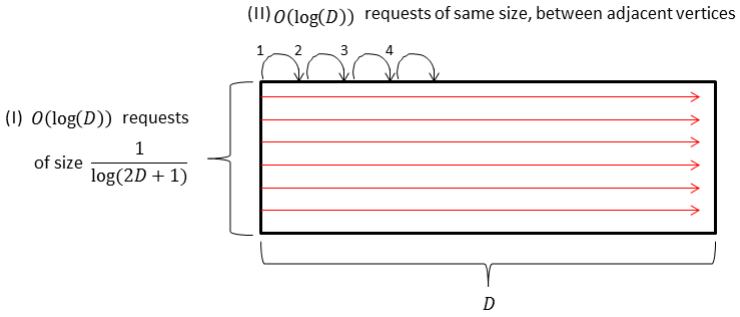
Necessary Assumption: Size of each request is small enough. $\forall i, e$ - $p_i \leq \frac{C(e)}{\log(2 \cdot D+2)}$
This means that there we cannot get a request that will jam an edge by itself and that we can get $\sim \log D$ request on each edge.

An example for the necessity of the assumption - if $p_i$=1:
Let G be a simple path with n edges. First request will be from 1st to last node with $p_1$=1. ON must accept it, otherwise OPT will, and we get OPT=1 and ON=0.
Now we provide a sequence of n requests with $p_i$=1, each on a separate edge. ON cannot serve them, as the load reached the capacity. OPT will serve them all. We get ON=1 and OPT=n.

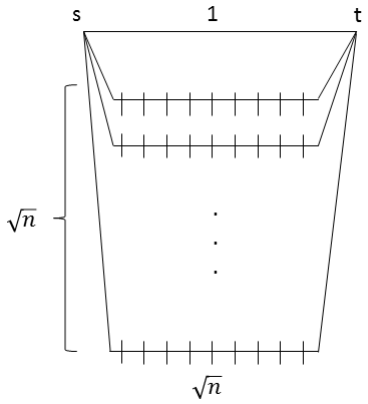
## 3.1 Simple solutions

We show that simple strategies yield a competitive ratio of at least $D$ (or $\sqrt{D}$).

**Algorithm 1** - Serve every request.
First, we'll provide a sequence of $O(\log D)$ requests from 1st to last node, each of size $\frac{1}{\log(2D+1)}$.
ON will serve them all, the load of every edge will reach its capacity of 1 and ON's benefit will be 1.
Then we'll provide a sequence of requests of size 1, each between different adjacent nodes.
OPT will serve only this sequence of requests, gaining total benefit of D.
ON is D-competitive.




**Algorithm 2** - Use shortest path only (or approximation to shortest path).
We have many request from $s$ to $t$.
OPT can get value $\Theta(\sqrt{n})$.
To be better than $\sqrt{n}$ competitive, ON must use path of length $\sqrt{n}$ although the shortest path is of length 1.




## 3.2 Exponent Algorithm

We'll describe a feasible, $O(\log D)$-competitive algorithm. Let $X_e$ be a weight on an edge such that $X_e = (2 \cdot D + 2)^{l_e} - 1$, where $l_e$ is the total load on the edge.

**Exponent Algorithm:** Serve request i only if exists a path from $s_i$ to $t_i$ with total weight $\leq D$.
Total weight = sum of all $X_e$ of all edges in the path (not including the added load of request i).
Note that we don't need to choose the minimal path.

The usage of exponent weights resembles the solution of the "One Way Trading" problem - the higher the level, the tougher the conditions (higher minimum share sell price vs total accumulative weight) competitive.

**Theorem:** Exponent Algorithm is feasible and $O(\log D)$

**Proof:** (ONLY Feasibility)
Overflow may occur on an edge only if $l_e \geq 1 - \frac{1}{\log(2 \cdot D+2)}$ (since we assumed $p \leq \frac{1}{\log(2 \cdot D+2)}$). In that case, $X_e > (2 \cdot D + 2)^{1-\frac{1}{\log(2 \cdot D+2)}} - 1 = \frac{(2 \cdot D+2)}{2} - 1 = D$ (where the first equality derived from: $a^{1/\log a} = 2$). This implies that our algorithm will not use that edge, since the total weight of the path used should be $\leq D$.

Proof of competitiveness - see next lecture...