

Disclaimer: *These are rough notes, with some exercises.*

15.1 Path selection without linear programming.

Question: Do you recall, the path selection problem?

Sure, we wish to choose a path for each of k pairs s_i, t_i and minimize the maximum congestion.

Question: How did we solve it previously?

We solved the following linear program.

$$\sum_{p \in \mathcal{P}_i} f_p = 1.$$

$$\sum_{e \in p} f_p < C.$$

$$\min C.$$

And used randomized rounding to get a solution with congestion $C + O(\sqrt{C \log n} + \log n)$, with n being the number of nodes in the network.

Question: A bit of heavy machinery? What is a heuristic method?

Route along shortest paths, do well “on average”. But, we might congest an edge, for example, consider a network with k disjoint long paths and one short path. Make congested edges long and route along shortest path.

Question: Two person game?

Sure, this is a two person game. The dual player is weighting the edges to make shortest paths as expensive as possible.

$$\max \sum_i d_i.$$

$$\forall p \in \mathcal{P}_i \sum_{e \in p} d(e) \geq d_i.$$

$$\sum_e d(e) = 1.$$

The d_i are the length of the shortest s_i, t_i paths. These problems are linear programming duals. Complementary slackness says to route only along shortest paths, and to have distance only on congested edges.

Question: Never mind. How should you penalize congested edges?

Why not exponentially. An edge with flow $f(e)$ should get $d(e) = (1 + \epsilon)^{f(e)}$.

Question: Now what?

Ok, the router routes along shortest paths, the toll collector charges exponentially as above, the router routes along shortest paths. Might be a loop!

Question: Ok, how to fix the loop?

Only (re)route a path if it is much shorter. Only if the new path is shorter by a factor of $(1 - 2\epsilon)$.

Question: Why does this terminate?

Let's consider a potential function, the sum of the distances $\sum_e d(e) = \sum_e (1 + \epsilon)^{f(e)}$. Does this decrease? Well, consider an update.

Question: What does an update do?

Reducing the flow on a path, p , and increases it on p' . Reduce the potential on p by

$$d(p) - d(p)/(1 + \epsilon) \approx d(p)(-\epsilon + \epsilon^2),$$

from $1/(1 - x) \approx (1 + x - x^2)$ using Taylor's formula for $1/(1 - x)$. Furthermore, we increase the potential on p' by

$$d(p')(1 + \epsilon) - d(p') = \epsilon d(p').$$

Now, $d(p') \leq (1 - 2\epsilon)d(p)$, thus the increase is at most

$$\epsilon d(p') \leq (\epsilon - 2\epsilon^2)d(p').$$

Thus, the total potential decreases by $\epsilon^2/2$ times the length of the path.

Exercise 1: Figure out how to set things up, so that the number of iterations is polynomial. What is your polynomial?

Question: At some point, we terminate. What is the value of the congestion? What is a lower bound on optimal?

From linear programming duality, the optimal congestion is at least $\sum_i d_i / \sum_e d(e)$. This is weak duality and can be argued directly very easily as well using the notion that it is, the average congestion relative to the weight function. This must be a lower bound on the maximum congestion.

Question: How close does our flow get to this average for this weight function?

Let p_i be the path that routes flow i , we have the following.

$$\sum_i d(p_i) = \sum_e f(e)d(e) \leq \sum_i (1 + 2\epsilon)d_i \leq (1 + 2\epsilon)C_{opt} / \sum_e d(e)$$

Or, we have

$$\frac{\sum_e f(e)d(e)}{\sum_e d(e)} \leq (1 + 2\epsilon)C_{opt} / \sum_e d(e).$$

We want the left hand side to somehow represent the maximum $f(e)$.

Question: Why should it?

Well, the larger $f(e)$ are multiplied by bigger $d(e)$. Exponentially bigger!

Question: How to proceed?

Plug in $d(e)$.

$$\frac{\sum_e f(e)(1 + \epsilon)^{f(e)}}{\sum_e (1 + \epsilon)^{f(e)}}.$$

Let's only consider the big edges in the numerator. That is,

$$\sum_{e; f(e) \geq f_{max} - 2 \ln |E| / \epsilon} (1 + \epsilon)^{f(e)}$$

The rest has a multiplier $d(e)$ that is smaller by at least $1/|E|^2$.

$$f_{lb} \frac{\sum_{e > f_{lb}} (1 + \epsilon)^{f(e)}}{\sum_e (1 + \epsilon)^{f(e)}} \geq f_{lb}(1 - 1/|E|),$$

where $f_{lb} = f_{max} - 2 \ln |E| / \epsilon$. That is, we get that

$$(f_{max} - 2 \ln |E| / \epsilon)(1 - 1/|E|) \leq (1 + \epsilon)C_{opt}.$$

Or.

$$f_{max} \leq (1 + 1/|E|)(1 + \epsilon)C_{opt} + 2 \ln |E| / \epsilon.$$

(With the liberal notion that $1/1 - x \approx 1 + x$.)

15.2 Multiplicative updates.

Each of n stock advisor gives advice on a stock, whether it is to go up or down, every day. Which one should you trust?

Question: Can you eventually do as well as the best?

In the beginning, let's trust them all equally, and make a prediction based on uniformly weighting them, i.e. computing $\sum_i w_i p_i$ for each expert i and for weight initially $1/n$ on each expert.

Let's say up or down based on the sign of the result.

If an expert was wrong we decrease his weight a bit, say by a factor of $(1 - \epsilon)$ if he or she was wrong.

Question: How well do you do?

Lemma 15.1 *If m_i^t is the number of mistakes expert i , and m^t be the number of mistakes made by the algorithm, then*

$$m^t \leq \frac{2 \ln n}{\epsilon} + 2(1 + \epsilon)m_i^t.$$

Question: Why? Well, what is w_i^t ?

Well,

$$w_i^t = (1 - \epsilon)^t.$$

Question: Well, then?

Let the potential be the sum of the weights. Every time one makes a mistake the potential decreases, half the weights decrease by a factor of $(1 - \epsilon)$ thus the potential decreases by $(1 - \epsilon/2)$. Furthermore, $\Phi \geq w_i^t$, for all i