Previously, we have considered on-line algorithms for routing and congestion control which required that we route (or reject) pairs as they were given. Now we will begin to examine off-line algorithms. These algorithms are given the entire set of inputs before they must make routing decisions, and so may achieve better preformance. Equivalently, we can think of off-line algorithms as having the freedom to reroute paths whenever they wish.

The following definitions carry over from the on-line to the off-line case. Let $f(e)$ be the number of paths routed on a single edge e. Let $\lambda := \max_e f(e)$ be the congestion of a routing. The minimum congestion over all routings will be denoted λ^*. Again $c(e)$ will be the cost associated with each edge. For this section we will assume that $c(e) \geq 0$. Finally, $\mathrm{dist}_c(s,t)$ will denote the total cost of the path from s to t using cost function c.

The next lemma (presented last time) will be crucial in developing preformance guarantees for our algorithm.

Lemma 1

$$\frac{\sum_{s,t} dist_c(s,t)}{\sum_e c(e)} \leq \lambda^* \leq \lambda$$

Note that this lemma works for any cost function such that $c(e) \geq 0$ for all edges e. In our off-line algorithm, $c(e)$ will change as we reroute various paths, but c(e) will always be non-negative. The fact that λ^* is "sandwiched" between two expressions gives an immediate corollary to the lemma.

Corollary 1 *If $f, \lambda,$ and c are as above, $\epsilon \geq 0$, and*

$$\lambda \leq (1+\epsilon)\frac{\sum_{s,t} dist_c(s,t)}{\sum_e c(e)}$$

then $\lambda \leq (1+\epsilon)\lambda^*$.

Recall that if we let $P_{s,t}$ denote the $s-t$ path which is being using in the current solution then

$$\begin{aligned}
\sum_{s,t} \mathrm{dist}_c(s,t) &\leq \sum_{s,t}\left(\sum_{e\in P_{s,t}} c(e)\right) \\
&= \sum_e c(e)f(e) \\
&\leq \lambda \sum_e c(e)
\end{aligned}$$

■

Our main goal is to make the $\sum_{s,t} \mathrm{dist}_c(s,t)$ and $\lambda\sum_e c(e)$ as close to one another as possible. From the proof of the lemma, this entails making two inequalities close to equalities:

Subgoal 1: Make $\sum_e c(e)f(e) \leq \lambda \sum_e (c(e))$ close to an equation.

Subgoal 2: Make $\sum_{s,t} \text{dist}_c(s,t) \leq \sum_{s,t} \left(\sum_{e \in P_{s,t}} c(e)\right)$ close to an equation.

Note that from a linear programming point of view, our lemma is just weak duality applied to a linear program version of our problem. Subgoals 1 and 2 mimic the complementary slackness conditions.

So far we have not yet stated mathematically what we mean by "close to an equation". Loosely speaking, the closeness of the equation will be related to the closeness of our approximate solution to the true solution.

The remainder of this lecture will be devoted to accomplishing subgoal 1. We will say that a cost function $c(e)$ satisfies condition 1 if for every edge either

$$f(e) \geq \frac{\lambda}{1+\epsilon} \text{ or } c(e) \leq \frac{\epsilon}{m} \max_e c(e).$$

In other words, either an edge is being used a number of times which is close to the congestion, or the cost of the edge is small compared with the maximum cost edge. It turns out that condition 1 is exactly what we need to meet subgoal 1.

Lemma 2 *If a cost function $c(e)$ satisfies condition 1, then*

$$\lambda \sum_e c(e) \leq \frac{1+\epsilon}{1-\epsilon} \sum_e c(e)f(e).$$

Proof. Set $\tilde{E} = \{\text{edges } e \text{ such that} f(e) \geq \lambda/(1+\epsilon)\}$. Then

$$\begin{aligned}
\lambda \sum_e c(e) &= \lambda \sum_{e \in \tilde{E}} c(e) + \lambda \sum_{e \notin \tilde{E}} c(e) \\
&\leq \lambda \sum_{e \in \tilde{E}} c(e) + \lambda \cdot m \left(\frac{\epsilon}{m} \cdot \max_{\bar{e}} c(\bar{e})\right) \\
&\leq (1+\epsilon) \sum_{e \in \tilde{E}} c(e)f(e) + \lambda\epsilon \max_{\bar{e}} c(\bar{e}) \\
&\leq (1+\epsilon) \sum_e c(e)f(e) + \lambda\epsilon \max_{\bar{e}} c(\bar{e})
\end{aligned}$$

Rearranging the terms gives

$$\begin{aligned}
(1+\epsilon) \sum_e f(e)c(e) &\geq \lambda \sum_e c(e) - \lambda\epsilon \max_{\bar{e}} c(\bar{e}) \\
&\geq \lambda \sum_e c(e) - \epsilon\lambda \sum_e c(e) \\
&= \lambda(1-\epsilon) \sum_e c(e).
\end{aligned}$$

Dividing both sides by $1-\epsilon$ gives the desired result. ■

As in the on-line case, our cost function will be a base raised to the power of the number of paths using that edge. In this case, we set the base so that we are ensuring that condition 1 is satisfied.

Lemma 3 *If $c(e) = a^{f(e)}$ and $\log a \geq \frac{1+\epsilon}{\lambda\epsilon} \log \frac{m}{\epsilon}$ then condition 1 is satisfied.*

Proof. Suppose that edge e satisfies $f(e) < \frac{\lambda}{1+\epsilon}$. (Then we must have $c(e) \leq \epsilon/m \max_{\bar{e}} c(\bar{e})$ in order to satisfy condition 1.) Note $c(e) = a^{f(e)} \leq a^{\frac{\lambda}{1+\epsilon}}$.

By our assumption on a,

$$\begin{aligned} \log a \geq \frac{1+\epsilon}{\lambda\epsilon} \log \frac{m}{\epsilon} \quad & \Rightarrow \quad \log \frac{m}{\epsilon} \leq \frac{\lambda\epsilon}{1+\epsilon} \log a \\ & \Rightarrow \quad \frac{m}{\epsilon} \leq a^{\lambda\epsilon/(1+\epsilon)} \\ & \Rightarrow \quad \frac{\epsilon}{m} \geq a^{\frac{\lambda}{1+\epsilon} - \lambda} \\ & \Rightarrow \quad a^{\frac{\lambda}{1+\epsilon}} \leq \frac{\epsilon}{m} a^{\lambda} \end{aligned}$$

Combining, we have that $c(e) \leq \frac{\epsilon}{m} a^{\lambda}$. Now we just note that in our case $\max_{\bar{e}} c(\bar{e}) = a^{\lambda}$, so $c(e) \leq \frac{\epsilon}{m} \max_{\bar{e}} c(\bar{e})$, and so condition 1 is satisfied. ■

The only difficulty in applying this theorem directly is that a depends on the value of λ. To satisfy subgoal 1 with λ^* would require that we know that value of λ^* before beginning the algorithm. We can work around this by setting up phases where we are sure that a is large enough to satisfy the lemma.

> **Outline of off-line algorithm**
> Start with any routing, which gives f and λ
> The current phase will end if λ decreases by a factor of 2.
> During the phase choose a so that $\log a = \frac{2(1+\epsilon)}{\lambda\epsilon} \log \frac{m}{\epsilon}$.

Our lemma guarantees that during any phase condition 1 is always true. Hence subgoal 1 is satisfied, with "close" meaning that

$$\lambda \sum_{e} c(e) \leq \frac{1+\epsilon}{1-\epsilon} \sum_{e} c(e) f(e).$$

Now that we have met subgoal 1, we may turn our attention towards achieving subgoal 2. This will be accomplished by the work done inside of a phase. First we must change our outline of the off-line algorithm to read "phase ends if λ decreases by a factor of 2 or subgoal 2 is met". Within a phase, the outline looks like:

> **Outline of one phase**
> While phase continues (subgoal 2 unmet, λ hasn't decreased by a factor of 2)
> For each s, t pair:
> Compare $dist_c(s,t)$ and $\sum_{e \in P_{s,t}} c(e)$ and reroute if necessary.