# Lecture 9

## 1 Overview

Last time we looked at packet scheduling in full generality and described a centralized method for designing a schedule using random delays.

Today we will look in path selection in general networks. We will use this problem to introduce linear programs and certain fast methods for solving linear programs (these methods are used in many areas ranging from machine learning to finance to optimization.)

## 2 Path Selection in general networks.

The generic problem is given a graph $G = (V, E)$ and a set of pairs $(s_1, t_1), \ldots, (s_k, t_k)$ find paths for the pairs that minimize the maximum over edges $e$ of the number of paths that use $e$. That is, we wish to minimize the maximum congestion of the solution.

This problem can be posed as the following integer linear program. We let $\mathcal{P}_i$ denote the set of paths between $s_i$ and $t_i$. We let $\mathcal{P}$ denote the set of all paths in $G$.

$$\min C$$
$$\forall e \sum_{e \in p \in \mathcal{P}} x_p \leq C$$
$$\forall i \sum_{p \in \mathcal{P}_i} x_p \geq 1.$$

If $x_p$ is forced to be in $\{0, 1\}$, this ILP exactly expresses a solution to our problem. There are two problems with this formulation. The first is that the constraint that the variables be integer is NP-complete to implement.

The second is that there are an exponential number of variables, even simple networks have an exponential number of paths. For example, the $2 \times n$ grid has an exponential number of paths from the top left node to the bottom right node. There are two ways around this. One is to use the ellipsoid method to solve the problem. We will perhaps discuss this approach later in the semester.

Another approach is to formulate a polynomial sized LP. The following is such a formulation.

$$\min C$$
$$\forall e \sum_{i} f_i(e) \leq C$$

$$\forall i, \sum_{e=(s_i,v)} f_i(e) = 1.$$

$$\forall v \notin \{s_i, t_i\} : \sum_{e=(u,v)} f_i(e) - \sum_{e=(v,u)} f_i(e) = 0.$$

This is a "flow" formulation, where $f_i(e)$ is the amount of flow for pair $i$ that traveres edge $e$. The first constraint ensures the congestion is not too high. The second set of constraints enforces the condition that at least one unit of flow is routed out of $s_i$. The third set of constraints enforces the condition that the flow into a node of a certain type equals the flow out of the node.

This particular problem is an instance of the multicommodity flow problem. From a solution to this formulation, one can obtain a solution to the first form. First note that this problem too may not solve to integers. For example, for the problem of routing on a "square" graph, routing one unit of flow from the top right to the lower left and routing one unit of flow from the lower right to the top left can be done with congestion 1 by routing a half unit of flow on each of the two possible simple paths for each pair. Routing with single paths always creates a congestion of 2.

### 2.1 Flow decomposition.

Thus, we assume for now that we have a "fractional" solution to the second linear program. We would like to have a set of fractional paths that we can use to route on. This can be accomplished using a simple idea sometimes called flow decomposition.

For each particular pair $(s_i, t_i)$, we do the following. We find a path from $s_i$ to $t_i$ where every edge has a positive flow value $f_i(e)$. We can do this since there is as long as some edge adjacent to source with positive flow, and the conservation constraints essentially allow us to leave any node we enter until we reach the sink.

We output $(p, x_p)$ where $p$ is the path and $x_p$ is the minimum flow value on any edge in the path. We subtract $x_p$ from the $f_i(e)$ for every edge on the path. This preserves the conservation constraints and it reduces the flow between $s_i$ and $t_i$ by $x_p$.

We continue until no flow remains between $s_i$ and $t_i$. We have obtained a set $(p_1, x_{p_1}) \ldots$ where the total $x_{p_i}$ values sum to one.

*Question 1:* Describe an implementation of this algorithm. How long does it take? (Use $m$ as the number of edges in the graph and $n$ as the number of nodes in the graph.)

## 3 Randomized Rounding.

Now, we have a set of fractional paths for all our packet sources and destinations with small congestion. We wish to choose one path. How should we do it? Well, for each pair $(s_i, t_i)$ we have a set of paths and values of $x_p$'s that sum to 1. How about we view the $x_p$'s as probabilities and choose a path according to this probability distribution?

What is the maximum congestion? Well, what is the expected congestion? Well, for an edge $e$, we consider all the paths that use it. We define a $0-1$ variable $X_p$ for each path $p$ that uses it, and note that the expectation for the $X_p$ is $x_p$. Thus, the expected congestion

for $e$ is

$$\sum_p x_p \leq C.$$

Cool.

With high probability, what is the maximum congestion. We can use Chernoff bounds to prove that the maximum congestion is at most

$$C + O(\sqrt{C \log N}) + O(\log N).$$

**Question 2:** Show this. In class, I described how to deal with dependence. Try to remember and explain the dependence, and explain how to get around it.