MS&E 319/CS 369X: Topics in Network Algorithms. Winter 2005-06
Course URL: `http://www.stanford.edu/~ashishg/network-algorithms`.
Instructor: Ashish Goel, Stanford University.

Lecture 9, 2/15/2006. Scribed by Dominic DiPalantino.

9.1 Throughput Competitive On-line Routing

In the problem under consideration (taken from [1]), request i arrives in an on-line fashion, in the form $< s_i, t_i, r_i >$ representing the source node, destination node, and traffic rate requirement respectively. Upon receipt of request i, the algorithm must either accept it or reject it, with no knowledge of future requests. In the algorithm considered in the previous lecture, we proceeded by modifying $\frac{\min_{e \in R} c_e}{\tilde{s}_t}$, increasing the numerator while scaling the denominator to keep it feasible.

In primal-dual algorithms, it is imporant to increase slowly. Equivalently here, we assume that $r_i \leq \frac{1}{\log \mu}$, with $\mu = 2(N+1)$ and N the number of nodes in the network.

Herein we assume that all capacities are 1, and that all accepted connections remain until time infinity.

Define the dual variables $\lambda_i(e) =$ load on edge e before i^{th} request, and $l_i(e) = \frac{1}{N}(\mu^{\lambda_i(e)} - 1)$. Note that l_i has values near 0, and then increases sharply as λ approaches 1.

The algorithm is as follows: for the i^{th} request, compute the cost θ_i of the shortest-cost route (using dual costs). If $(\theta_i > 1)$, reject the request; else reserve bandwidth r_i along every edge on a shortest-cost route.

We need to check a few properties of this algorithm.

Proposition 9.1 *The algorithm maintains feasibility.*

Proof: Suppose, for the sake of contradiction, that edge e exceeds capacity after the i^{th} request, and is the first to do so. Then this request was accepted and used edge e, so $l_i(e) \leq 1$, i.e. $\frac{1}{N}(\mu^{\lambda_i(e)} - 1) \leq 1$. Exceeding capacity means $\lambda_i(e) + r_i > 1$. So $\lambda_i(e) + \frac{1}{\log \mu} > 1$, implying $\mu^{\lambda_i(e)} > \mu^{1 - \frac{1}{\log \mu}} = \frac{\mu}{2}$. So $\mu^{\lambda_i(e)} > N + 1$, contradicting $\frac{1}{N}(\mu^{\lambda_i(e)} - 1) \leq 1$. Hence, the algorithm has feasible output. ■

Next, we relate profit to costs. Toward this end, let π_i be the total amount of accepted requirement before i^{th} request, i.e. the sum of all accepted r_k prior to i.

Proposition 9.2 *For all k, we have $2\pi_k \log \mu \geq \sum_e l_k(e)$.*

Proof: We proceed by induction, noting that it suffices to show that as each packet is accepted, the change in the LHS is greater or equal than the change in the RHS. (Note that the statement is trivially true at $k = 0$, and that packets that are rejected do not affect the equation at all.)

The change in LHS is $2r_k \log \mu$, that in RHS is $\sum_{e \in R_k}(l_{k+1}(e) - l_k(e))$, where R_k is the route for accepted packet k.

Then the change in RHS is $\frac{1}{N}\sum_{e \in R_k}(\mu^{\lambda_{k+1}(e)} - \mu^{\lambda_k(e)}) = \frac{1}{N}\sum_{e \in R_k}(\mu^{\lambda_{k+1}(e)}(\mu^{r_k} - 1))$.

Next, we use a small trick from analysis. We have $\mu^{r_k} - 1 = 2^{r_k \log \mu} - 1$. This is of the form $f(z) = 2^z - 1$; note that $f(0) = 0$, $f(1) = 1$, and f' is increasing; it then follows that $f(z) \leq z$ for $z \in [0, 1]$. In particular, $\mu^{r_k} - 1 \leq r_k \log \mu$, and the change in RHS is less than or equal to $\frac{1}{N} \sum_{e \in R_k} (\mu^{\lambda_{k+1}(e)} r_k \log \mu) = \frac{r_k \log \mu}{N} \sum_{e \in R_k} (\mu^{\lambda_{k+1}(e)} - 1 + 1) \leq \frac{r_k \log \mu}{N} \sum_{e \in R_k} 1 + \frac{r_k \log \mu}{N} (\sum_{e \in R_k} (\mu^{\lambda_k(e)} - 1))$. Noting that $\sum_{e \in R_k} (\mu^{\lambda_k(e)} - 1) \leq 1$, we have that the above is $\leq 2 r_k \log \mu$, our desired result. This proves the inductive step, and we're done.

■

Next, we wish to relate costs to the optimal profit. Let π^* be the optimal profit, given full knowledge of the incoming requests. We decompose this as $\pi^* = \pi^{(I)} + \pi^{(X)}$, where the first term is the profit from requests routed from both our algorithm and the optimal decision process, and the second is the proft from requests rejected by our algorithm but accepted in the optimal case. Implicitly, we have a notion here of a final request; let t be the termination time in the above definition, i.e. we only face t requests, and the profit of our algorithm is π_t. We obtain the following bound:

Proposition 9.3 $\pi^* < \pi_t (2 \log \mu + 1)$

Proof: The proof consists of bounding $\pi^{(I)}$ and $\pi^{(X)}$ individually. First, note we have $\pi^{(I)} \leq \pi_t$.

Finally, we bound $\pi^{(X)}$. Toward this end, let σ_X be the set of requests that contribute to $\pi^{(X)}$. Then for $i \in \sigma_X$, we have $\sum_{e \in R_i^*} l_i(e) > 1$, where R_i^* is the route used by the optimal solution. It follows that $\sum_{e \in R_i^*} l_t(e) > 1$, since this terms only increase as $i \to t$. Then we have $r_i \sum_{e \in R_i^*} l_t(e) > r_i$; summing over $i \in \sigma_X$, we have $\sum_{i \in \sigma_X} r_i \sum_{e \in R_i^*} l_t(e) > \sum_{i \in \sigma_X} r_i = \pi^{(X)}$. Exchanging the order of summation, we have $\sum_e l_t(e) \sum_{i \in \sigma_X} r_i > \pi^{(X)}$. The optimal solution must be feasible, so $\sum_{i \in \sigma_X} r_i \leq 1$, whence $\sum_e l_t(e) > \pi^{(X)}$. This combined with the previous proposition yields $\pi^{(X)} < 2 \pi_t \log \mu$. Combining this with the above bound on $\pi^{(I)}$, we have the desired result.

■

This useful result was obtained while presuming very little about the adversary.

9.2 Open Shortest Path First

In this scheme, all links on the network have associated costs. Open Shortest Path First, or OSPF, computes the shortest cost for each source, destination pair, and sends each packet along these paths.

If routers set these costs, they should be of the form $\alpha \beta^\lambda$, where α and β are parameters and λ is the load. This scheme, when stabilized, sends most data across. This assumes, however, that all routers are using the scheme.

Consider a packet proceeding along a fixed route from s_i to t_i. Each router along the way could add its dual cost - t_i could then send the total back to s_i, which could then decide whether or not to increase flow. In this way, routers could set costs.

References

[1] B. Awerbuch, Y. Azar, and S. Plotkin. *Throughput-competitive online routing.* Proc. of FOCS, 32-40, 1993.

[2] F. Kelly, A. Maulloo, and D. Tan. *Rate control in communication networks: shadow prices, proportional fairness and stability.* Journal of the Operational Research Society, 49:237–252, 1998.