Network Design Foundation
Fall 2011
Lecture 7

Instructor: Mohammad T. Hajiaghayi
Scribe: Anu Bandi

October 12, 2011

1 Overview

In this lecture we explore the problem of OBLIVIOUS ROUTING. Oblivious routing is used to solve the network flow problem by preselecting the paths to use for any source-destination pair. Therefore this algorithm is oblivious as to what the input source-destination pair might be, as well as to other source-destination pairs.

2 Definitions

OBLIVIOUS ROUTING

INPUT : A graph $G(V, E)$, where each edge (u, v) has capacity $C(u, v)$. Source target pairs (s_i, t_i) and a demand d_i for each pair.

GOAL: Find the fixed routing rule to route 1 unit of flow between all possible source-destination pairs, while keeping congestion close to optimum congestion for any set of source-destination pairs and demands.

$$\textbf{Minimize: } max_{demand-matrixD}\left[\frac{Congestion_{OBL}(D)}{Congestion_{OPT}(D)}\right]$$

The demand-matrix D contains the congestion between any two vertices. The congestion on an edge e is defined as follows:

$$Congestion_e = \frac{flow_e}{capacity_e}$$

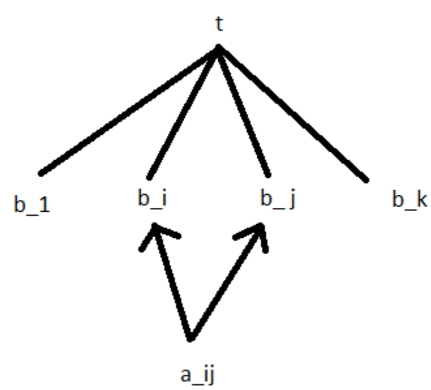
(*Note*: For the purposes of this algorithm, don't worry about violating the capacity. Assume that the flows will be much smaller than the capacity or that these are soft capacities.)

3 Results

1. For undirected graphs there are oblivious routing algorithms with these competitive ratios. (*Note*: COMP(OBL) compares with OPT offline.)
   - $O(\lg^3 n)$ by [Racke Focs '02]
   - $O(\lg^2 n \lg \lg n)$ by [HHR '03]
   - $O(\lg n)$ by [Racke Stoc '08]
2. For directed or node weighted graphs, there are directed graphs in which every oblivious routing algorithm OBL has these COMP(OBL):
   - $\Omega(\sqrt{n})$ by [ACFKR STOC '03]
   - $\Omega(\sqrt{n})$ by [HKRL SODA '05]
3. If you know the probability of each demand i.e. the demand distribution, then the competitive ratio for *undirected graphs* is as follows:
   - $O(\lg n)$
   - $\Omega(\frac{\lg n}{\lg \lg n})$
4. If the demand distribution for a *directed graph* is know then the competitive ratio is as follows:
   - $O(\lg^2 n)$
   - $\Omega(\lg n)$

4 COMP(OBL)=$\Omega(\sqrt{n})$

Proof (by construction):



Consider this weighted directed graph, where each edge has capacity one. The commodity pairs are (a_{ij}, t). Any oblivious routing scheme defines one unit of flow from each node a_{ij}. Then it follows by the averaging argument, that atleast one node b_x receives $\geq \binom{k}{2} / k$ units of flow.

$$\mathsf{Flow}(b_x) \geq \binom{k}{2} / k$$

$$\mathsf{Congestion}(b_x) \geq \frac{\binom{k}{2} / k}{1} = \binom{k}{2} / k$$

$$\mathsf{Congestion}(\mathsf{OBL}) = \binom{k}{2} / k = \frac{k(k-1)/2}{k} = \frac{k-1}{2}$$

However OPT will avoid b_x completely. It will route demands from nodes a_{ix} using the paths $a_{ix} \to b_i \to t$. Similarly it will route demands from nodes a_{xj} using the paths $a_{xj} \to b_j \to t$.

$$\mathsf{Congestion}(\mathsf{OPT}) = 1$$

$$\mathsf{COMP}(\mathsf{OBL}) = \frac{k-1}{2} = \Omega(k) = \Omega(\sqrt{n})$$

5 Tree Decomposition

Racke (Focs '02) introduced a tree decomposition that aims at constructing a tree that does not approximate point-to-point distances in the input graph (like Bartal or FRT's technique) but instead approximates the *cut structure* of the graph.

The Model

We are given a graph $G(V, E)$ where $|V| = n$. We have a capacity function C on the edges. $C(u, v) = C(v, u)$ since the graph is undirected. Assume $C(u, v) > 0$ if there is an edge (u, v) and $C(u, v) = 0$ iff $(u, v) \notin E$.

Decomposition Trees

Like Bartal or FRT, a decomposition tree for the graph G is a rooted tree $T = (V_t, E_t)$ whose leaf nodes correspond to nodes in G. Whenever we use the concept of a decomposition tree for a graph G, we implicitly assume that we are also given an embedding of T into G using these functions:

- $m_V : V_t \to V$ that maps tree nodes to nodes in the original graph.
- $m_E : E_t \to E^*$ that maps an edge $e_t = (u_t, v_t)$ of T to a path P_{uv} between the corresponding end points $u = m_v(u_t)$ and $v = m_v(v_t)$ in G.

In addition we also introduce the following functions:

- $m'_V : V \to V_t$ that maps nodes in the graph to leaf nodes in T.
- $m'_E : E \to E^*_t$ that maps an edge $e = (u, v) \in E$ of G to the unique shortest path in T between $m'_v(u)$ and $m'_v(v)$.

For a multicommodity flow f_T on a decomposition tree we use $m(f_T)$ to denote the multicommodity flow obtained by mapping f_T to G via the edge mapping function m_E. For a flow f in G we define $m'(f)$ as the flow in T.

Given a decomposition tree T for G we define the capacity $C(u_t, v_t)$ of a tree edge $e_t = (u_t, v_t)$ as $c(u_t, v_t) = \sum_{u \in V_{ut}, v \in V_{vt}} c(u, v)$, where V_{ut} and V_{vt} denote the two partitions of V induced by the cut corresponding to edge e_T.

Theorem 1 *Suppose you are given a multicommodity flow f in G with congestion C_G. Then the flow $m'(f)$ obtained by mapping f to some decomposition tree T results in a flow in T that has congestion $C_T \leq C_G$.*

Proof: Suppose an edge $e_t = (u_t, v_t)$ in the tree has congestion C_T. All traffic that traverse the cut in G between V_{ut},V_{vt} contributes to this edge. The total capacity of all edges over this cut is exactly $C(e_t)$. Hence by a simple averaging argument, one of these edges must have relative load at least C_T. Thus $C_T \leq C_G$. ■

6 $O(\lg n)$ Bound

Given a decomposition tree with an embedding of this tree into graph G we can ask for the load that is induced on a graph edge e by this embedding. Let

- $load_T(e) = \sum_{e_t \in E_T : e \in m_E(e_t)} c(e_t)$
- $rload_T(e) = \frac{load_T(e)}{c(e)}$

Like Bartal or FRT, we are looking for a convex combination of decomposition trees such that for every edge the expected relative load is small, i.e.

$$\textbf{minimize } B = max_{e \in E} \sum_i [\lambda_i rload_{T_i}(e)] = max_{e \in E} \sum_i \frac{\lambda_i load_{T_i}(e)}{c_e}$$

Where tree i has probability $\lambda_i \geq 0$. And $\sum \lambda_i = 1$.

Theorem 2 *Suppose we are given a convex combination of decomposition trees with maximum expected relative load B and suppose that we are given for each tree T_i a multicommodity flow f_i that has congestion C in T_i. Then the multicommodity flow $\sum \lambda_i m_{T_i}(f_i)$ has congestion at most BC when mapped to G.*

(*Note*: This is like FT or Bartal.)

Proof: Fix a tree T_i. Routing the flow f_i in the tree generates congestion at most C, which means that the amount of traffic that is sent along an edge

$e_t = (u_t, v_t)$ is at most $C(e_t)$. Hence the total traffic that is induced on a graph edge e when mapping $\lambda_i f_i$ to G is at most $C\lambda_i load_{T_i}(e)$. Therefore the relative load induced on e when mapping all flows $\lambda_i f_i$ is at most $C \sum_i \frac{load_{T_i}(e)}{C(e)} = C \sum_i \lambda_i rload_{T_i}(e) \leq CB$. ■

[Racke Stoc '08] shows that we can indeed obtain a convex combination of decomposition trees for which $B = O(lgn)$.

New Oblivious Routing Algorithm: The convex combination of decomposition trees defines a unit flow for every source-target pair, by combining for a pair (u, v) the paths between u and v in trees T_i where the path from T_i is weighted with λ_i.

Proof: Given a demand vector that can be routed with congestion C in G, routing it in a decomposition tree creates congestion less than C in any tree by Thm 1. Now mapping the flows from all decomposition trees back (and thus scaling it by a factor λ_i) the thing that indeed we have done in our oblivious routing gives a solution in G with congestion at most $Bmax_i C_{OPT}(T_i) \leq BC_{OPT}(G)$ due to Thm 2. Hence the oblivious routing scheme has competitive ratio $O(lg\ n)$ as desired. ■

This convex combination of trees has several other applications like the Bartal/FRT result. Ex. min bisection, sparset cut, multicost routing, online multicut, etc. It gives $O(lg\ n)$ approximation.