CS 684: Approximation and Network Algorithms	**11/14/01**
Connectivity Network Design I	**scribe: Andrew Hutchings**

1 Connectivity Network Design

Today we revisit the previously discussed the Connectivity Network Design problem. Given a graph $G = (V, E)$, and a cost c_e for each edge e, and a connectivity requirement $r_{ij} = r_{ji}$ for each pair of nodes (i, j), we wish to find the minimum cost set of edges such that the minimum $i - j$ cut is at least r_{ij} for all nodes.

Earlier we had a $2H(R)$ approximation for this problem, with R being the the maximum connection requirement and $H(t)$ being the harmonic sum out to t terms. We wish to have a better, constant factor approximation using a stronger technique.

2 Explicit Use of Linear Programs for Approximation

As it turns out, we can use properties of Linear Programs to approximate Connectivity Network Design to within 2. For our purposes (and for simplicity of analysis) we will only get a factor of 3 approximation.

We proceed by relaxing our desired problem as a Linear Program, with decision variables x_e representing the choice to include edge e in our solution. If we let $f(S) = \max_{i \in S, j \notin S} r_{ij}$, then our LP is:

$$\begin{aligned} \min \ & \sum_e c_e x_e & \\ \text{s.t. } & 0 \leq x_e \leq 1 & \text{for all } e \in E \\ & \sum_{e \in \delta(S)} x_e \geq f(S) & \text{for all } S \subseteq V \end{aligned}$$

Claim 2.1 0-1 *solutions of this LP are in* 1 *to* 1 *correspondence with solutions to the network design problem.*

So if we could solve this LP as an IP, we would have solved this network design optimally. Our scheme will be to solve the LP, and "hope" for some edge e to carry a large value, in this case we want $x_e \geq 1/3$. Using such "heavy" edges and recursing will give us the desired approximation. To this end we will establish that, in fact, it will always be the case that at least one edge will carry a sufficiently high value.

3 Vertices of the LP Solutions

Consider min cost bipartite matching, expressed as an LP. This will serve as an example of how the structure of LP solutions can force some variables to be large.

Definition 3.1 *A feasible LP solution is a vertex if and only if there does not exist* $\mathbf{y}$ *such that* $\mathbf{x}+\mathbf{y}$ *and* $\mathbf{x}-\mathbf{y}$ *are both feasible solutions.*

In an LP with constraints $0 \leq x_i \leq 1$ for all i and $A\mathbf{x} \geq \mathbf{b}$, let A_0 be the submatrix of A using only the rows i for which $A_i\mathbf{x} = b_i$ and columns j for which $0 < x_j < 1$.

Claim 3.2 *A feasible point* $\mathbf{x}$ *is a vertex iff* A_0 *has full column rank.*

Note that if $\mathbf{x}$ is not a vertex then any $\mathbf{y}$-direction must have $y_e = 0$ if $x_e = 0$ or $x_e = 1$ (otherwise it can make $\mathbf{x}$ infeasible in that coordinate). For the rows with non-tight inequalities, any coordinates will work, since we can always choose an ϵ small enough that those inequalities aren't violated by $a_e \cdot (\mathbf{x} \pm \epsilon\mathbf{y})$.

Let $\mathbf{y}'$ and $\mathbf{x}'$ be coordinates of $\mathbf{y}$ and $\mathbf{b}$ restricted to the columns of A_0. Let $\mathbf{b}'$ be $\mathbf{b}$ restricted to the rows of A_0. Since $A_0\mathbf{x}' = \mathbf{b}'$, we must have $A_0\mathbf{y} = 0$ in order for $A_0(\mathbf{x}' \pm \mathbf{y}')$ to be feasible. But in this case $\mathbf{y}'$ can only be nonzero if A_0 does not have full column rank, since $A_0\mathbf{y}' = 0$ implies either $\mathbf{y}' = 0$ or the linear dependence of the columns of A_0.

We also note that optimal LP vertices are obtainable in the process of solving the LP. Thus we can take advantage of the special structure of these solutions.

4 Special Cases of LP

Considering now bipartite matching expressed (and solved) as an LP.

$$
\begin{aligned}
\min\ & \sum_e c_e x_e & \\
\text{s.t. } & 0 \leq x_e \leq 1 & \text{for all } e \in E \\
& \sum_{e \in \delta(v)} x_e \geq 1 & \text{for all } v \in V
\end{aligned}
$$

Assume we have a vertex solution to the LP formulation. Delete all edges (and columns) e with $x_e = 0$, and assume, for contradiction's sake, that all edges have $x_e < 1$, which of course implies that every node has degree ≥ 2 in the resulting graph.

Note that we have $|V|$ rows of A_0 corresponding to the nodes of G. Also note that the structure of the bipartite graph ensures that these rows are not independent, since for any set of active edges the sums of the rows for the left and right sides are equal. This means that A_0 can have at most $|V| - 1$ columns and have full column rank.

Every node having degree at least 2 implies that there are at least $|V|$ edges, and therefore at least $|V|$ independent columns. This contradiction ensures that at least one edge e will be assigned a value of 1.

A similar argument can be made for the nonbipartite matching case, with weaker results. Using the same LP, we note that in this case the $|V|$ rows could all be independent, and if we once again assume that no $x_e = 1$, the degree argument above implies that each node (after deleting inactive edges) has degree at least two, and in fact, with $|V|$ active edges, every node must have degree exactly two, and our active graph is a collection of disjoint cycles. Thus the constraints of the LP ensure that some edge e (all edges, in fact) will have $x_e \geq 1/2$.

5 Back to Connectivity Network Design

We will establish a similar result stating that we can always find a "heavy" edge in a vertex solution to the Connectivity Network Design LP. We will show that and edge e with $x_e \geq 1/3$ always exists, which can be used for a 3-approximation. In actuality, edges of weight 1/2 always exist, implying a 2-approximation, but this is beyond our scope.

Our scheme will proceed as follows:

Algorithm 5.1 *1. Obtain a vertex solution to the Connectivity Network Design LP.*

2. Identify "heavy" edges with value $\geq 1/3$. Call these E_0. Let $\delta_{E_0}(S)$ be the number of edges of E_0 in the cut induced by S.

3. Update f by $f(S) \leftarrow f(S) - \delta_{E_0}(S)$ for all S. Include the heavy edges in our solution set, and delete them from the graph.

4. Repeat until $f(S) \leq 0$ for all S

Intuitively, we never increase the cost of an edge by more than a factor of 3 when we include it, and thus cannot overpay by more than this factor. Next time we will prove results about the existence of heavy edges in our network design problem and demonstrate that 3 is indeed the correct factor, given this analysis.