College of Computer & Information Science
Northeastern University
CS G399: Algorithmic Power Tools I

Fall 2007
12 October 2007
Scribe: Laura Poplawski

Lecture Outline:

- Network Design
  - Generalized Steiner Network (iterated rounding algorithm)

In this lecture, we first wrap up Steiner Forest (already included in the notes for the previous lecture) and introduce the Generalized Steiner Network problem.

1 Generalized Steiner Network

Problem 1. Given: undirected graph $G = (V, E)$, edge cost function $C : E \to \mathbb{Z}^+$, r_{uv} for all pairs of nodes u and v specifying a required number of edge-disjoint paths between u and v, u_e for each edge specifying how many copies of edge e we can create.
Goal: Determine subgraph H of G with least cost in which there are most u_e copies of each edge e, and for all pairs u,v, there are at least r_{uv} edge-disjoint paths connecting u and v.

2 LP for Generalized Steiner Network

As with Steiner Forest, we could write a flow-based LP, but we will work with a cut-based LP instead. For all cuts S, define $f(S) = max_{(u \in S, v \notin S)} r_{uv}$. In other words, $f(S)$ = the number of edges that we need to create that cross cut $(S, \bar{S})$. Let $\delta(S)$ = the set of all edges that cross cut $(S, \bar{S})$.

$$\text{minimize} \sum_{e \in E} x_e c_e$$

$$\forall S, \sum_{e \in \delta(S)} x_e \geq f(S)$$

$$\forall e, u_e \geq x_e \geq 0$$

Notice, there are an exponential number of constraints in this LP. However, the ellipsoid method can solve an LP with exponential constraints if we have a *polynomial-time separation oracle*. This is a method by which, given a potential solution x to the LP, we can in polynomial time either verify that x is a feasible solution or give a separating hyperplane H. A separating hyperplane is a hyperplane such that all points in the LP's solution are on one side of the hyperplane, x is on the other side. A constaint of the LP which is violated by x is always a separating hyperplane, since all points in the solution must satisfy all constraints.

3 Agenda for describing and proving a 2-approximation algorithm [1]

1. Show the above LP can be solved in polynomial time. In fact, show that an optimal *Basic Feasible Solution* (BFS) can be found in polynomial time. A BFS (also called a *vertex* or an *extreme point*) is a solution which lies at one of the vertices of the polyhedron described by the LP.

2. Prove that in any BFS of the above LP, there exists an edge e such that $x_e \geq \frac{1}{2}$.

3. Iterative rounding algorithm:
   1: $F \leftarrow \emptyset$
   2: define f (the remaining problem instance)
   3: **while** $f \neq 0$ ($\exists$ a set that is not satisfied) **do**
   4: solve the LP for f to obtain an optimal BFS x.
   5: add e to F if $x_e \geq \frac{1}{2}$.
   6: recompute f
   7: return F

4. Show that the algorithm gives a 2-approximation.

4 Solvable in polynomial time

To do step 1 in the agenda, we show that we can give a polynomial time separation oracle. Using an algorithm of Grotschel, Lovasz, and Schrijver, one can then obtain an optimal BFS of the LP in polynomial time.

To verify whether a given point x is a feasible solution: given x, we want to find out whether $\forall$ cuts S, $\forall u \in S, v \notin S \sum_{e \in \delta(S)} x_e \geq r_{uv}$. To do this, run a max flow algorithm for all u, v pairs (n^2 max flow iterations) with capacity of edge $e = x_e$ (for all e). If max flow between u and $v \geq r_{uv}$, for all u, v, then we have a feasible solution. If not, the max flow algorithm will give us the min cut, which will give us a cut that that is a violated constraint.

It is important to note that we have shown that the above LP is poly-time solvable for the class of functions f defined by the generalized Steiner Network constraints, not for any function f.

References

[1] K. Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica*: 21(1): 39-60, 2001.