COMPSCI 632: Approximation Algorithms	October 29, 2017
Lecture 17	
Lecturer: Debmalya Panigrahi	*Scribe: Keerti Anand*

1 Overview

In this lecture, we study dependent rounding and try to apply it on a classical problem, which is **Group Steiner Tree**.

2 Group Steiner Tree

The **Group Steiner Tree** problem, asks us that given a graph and a set of vertex class, construct a tree(forest) of minimum sum total of edge weights, that would maintain connectivity between any two representative vertices of the same class.

2.1 GST on a tree

Instead of solving the problem on a general graph, we try to solve it for a tree. We can argue that any graph has a distance preserving embedding in the form of a tree such that the distances stretch by a factor of atmost $log(n)$. Also, wlog we can assume that the vertices in a vertex class (henceforth referred to as terminals) are present as leaves in the tree. This is because if that is not a case, we can always add a zero cost edge from the non-leaf terminal to a newly created vertex and declare this vertex as the terminal of the vertex class.

Lemma 1. *GST is NP-Hard.*

Proof. GST is a more general version of the Steiner Tree problem. □

Lemma 2. *GST is hard to approximate. That is, there does not exist a constant factor approximation.*

Proof. We can reduce an instance of set cover to GST. Consider the following reduction, the sets from the Set Cover instance are represented as terminals, connected to the root by the cost equal to choosing that set. The elements of the set cover are characterised by the vertex class, that is, if an element belongs to a certain set of sets. Then the terminals corresponding to that set belong to the group corresponding to the element. Though, a terminal can belong to many groups, we can always construct an instance, where the groups they belong to our disjoint using zero weight edges such that the solution does not change. Now, it is easy to see that the min set cover corresponds to finding the GST on this tree. □

Theorem 3. *GST is at least as hard as Set Cover.*

Now we try to give an approximate algorithm for GST. Note that, k is the number of groups. To run this algorithm, we must make a few assumptions.

1. Every Group is Singleton. **Run the algorithm repeatedly for each vertex in a certain group**
2. Choose one of the vertex as the root of the tree. **This at most doubles the height of the tree**

Note Connecting the groups with each other is now equivalent to connecting them with the root.
Integer Case Every terminal from the group should be connected to the root.
Fractional Relaxation Fractionally choose the edges such that for each terminal, the total length of edges on at least one path to the root is at least 1.
Let us now write the LP of the fractional relaxation (note that $p(e)$ represents parent of e, in the path to the root, and $i(e)$ indicates the set of edges incident on terminal of group i) The variable $f_{e,i}$ denotes the flow on edge e for group i.

$$\begin{array}{lll} \text{minimize} & \sum_{e=(u,v)\in E} c_e X_e & \\ \text{subject to} & X_e \geq 0 & \forall e \in E \\ & f_{e,i} \leq X_e & \forall e \in E, \forall i \in 1,2..k \\ & \sum_{e:p(e)=e'} f_{e,i} = f_{e',i} & \forall i \in 1,2..k \forall e' \in E \\ & \sum_{i(e)} f_{e,i} \geq 1 \forall i \in 1,2..k & \end{array}$$

Now, we can solve this fractional solution using the LP solving algorithms and the question that remains to be answered is how to round this to an integral solution. Let us try a few options:

1. **Independent Rounding** Round each edge independently with probability X_e. Clearly, the expected cost is equal to the fractional solution. The problem, lies in the case, when there are two terminals, with very long paths from the root. The LP assigns $\frac{1}{2}$ to each edge. But, it is very easy to see that with high probability, one of the terminals is not connected to the root. Hence, this approach fails.
2. **Choosing paths for each group independently** We choose each path independently with the probability proportional to the sum of X_e of the edges on it. Suppose there are k branches from the root, and each branch has each one of the k terminals(attached to it with zero cost edges). The LP would assign $X_e = \frac{1}{k}$ for each edge in the branches. The probability that any path does not get chosen is given by $(1-\frac{1}{k})^k \leq e^{-1}$ Therefore, with constant probability, we do get a path. However, the cost of this scheme is k times the fractional cost. Hence, the competitive ratio becomes $\Omega(k)$.

Remark 1. *Both Independent rounding of edges and choosing paths independently do not yield the desired solution.*

3 Dependent Rounding

We require a form of dependent rounding such that if on a certain path, an edge is not chosen, then choosing the other edges on the path does not yield any gain. Also, in order to keep our expected cost same as the fractional cost, we need P[an edge e is chosen] = X_e. The following is the dependent rounding scheme:

1. For the top-most edges emanating from the root, sample with probability X_e independently.
2. For any subsequent edge, the probability of choosing it is conditional on whether its parent, $p(e)$ is chosen or not. If $p(e)$ is not chosen, then there is no point choosing it and hence P(choosing e| p(e) is not chosen) = 0. Also, P(choosing e|p(e) is chosen) = $\frac{X_e}{X_{p(e)}}$ Since

$$\sum_{e:p(e)=e'} f_{e,i} = f_{e',i}$$

, therefore we can claim, that $X_{p(e)} \geq X_e$ (try doing this yourself), and therefore, the probabilities make sense. Also, note that the unconditional probability of choosing edge e is equal to X_e.

Return the set of chosen edges.

4 Analysis

Let F be the output of the algorithm. S_i denotes the i_{th} group of terminals. $T_i = S_i \cap F$. We need to find $P(|T_i| \geq 1)$. Note that $E(|T_i|) = \sum_{v \in S_i} P(v \in F) = \sum_{v \in T_i} X_{e_v} \geq \sum_{v \in S_i} f_{e,i} = 1$ We need to bound the probability when $|T_i|$ becomes huge. For this, we define random variable $T_i^{'}$ such that $T_i^{'} = T_i$ if $|T_i| \leq 2H$ and $T_i^{'} = 0$ otherwise. (H is the height of the tree)
$E[|T_i| \mid v \in T_i] \leq H$
$E[|T_i|] = \sum_{v \in S_i} P(v \in T_i^{'}) = \sum P(v \in T_i).P(v \in T_i^{'} \mid v \in T_i)$
$= \sum_{v \in S_i} f_{e_v,i} P(|T_i| \leq 2H \mid v \in T_i) \geq \frac{1}{2} \sum_{v \in S_i} f_{e_v,i} = \frac{1}{2}$
$\Rightarrow E(|T_i^{'}|) \geq \frac{1}{2}$
$\Rightarrow P(|T_i^{'}| > 0) \geq \frac{1}{4H}$
Repeating our algorithm $4Hlogk$ times yields:
$\Rightarrow P(S_i \text{ is not connected}) \geq (1 - \frac{1}{4H})^{\frac{1}{4HlogH}} \geq \frac{1}{poly(k)}$ By taking union bound, we connect all S_i with high probability. Cost of algorithm is $4Hlogk \times OPT$.
Also, note that any graph can be converted into a tree with height $h = log(n)$. Therefore, the competitive ratio is $O(log(n)logk)$.