Advanced Approximation Algorithms	**(CMU 15-854B, Spring 2008)**

Lecture 12: Steiner Tree Problems

February 21, 2008

Lecturer: Anupam Gupta *Scribe: Michael Dinitz*

1 Finishing up Group Steiner Tree Hardness

Last time we saw the construction of Halperin and Krauthgamer [5] that reduces Label-Cover to Group Steiner Tree, proving $\Omega(\log^{2-\epsilon} k)$-hardness. In particular, suppose that we are given an instance $\mathcal{G}$ of Label-Cover, and the HK reduction results in an instance $\mathcal{T}$ of Group Steiner Tree. They show that if $\mathsf{OPT}(\mathcal{G}) = 1$ then $\mathsf{OPT}(\mathcal{T}) = H$, while if $\mathsf{OPT}(\mathcal{G}) < \eta$ then $\mathsf{OPT}(\mathcal{T}) = \Omega(H^2 \log k)$. Note that there is obviously a relation between η and the other parameters, but for clarity we will ignore it. By letting $H = \Theta(\log^{1+\epsilon} k)$, this gives the desired $\Omega(\log^{2-\epsilon} k)$ gap.

A few comments about how to read the the hardness paper [5] and the integrality gap paper [4]. First, at a high level, how does the set cover hardness argument go? It basically says that with high probability, if a set cover of low cost exists then there are a large number of constraints (i.e. edges) (u, v) in the original label cover instance such that $Sugg(u)$ and $Sugg(v)$ are consistent, where we say that they are consistent if there is some $\alpha \in Sugg(u)$ and some $\beta \in Sugg(V)$ such that $\pi_{uv}(\alpha) = \pi_{vu}(\beta)$.

Recall that the reduction of [5] worked by taking a large number of elements (groups) and flipping coins at each level of the tree to see how they filtered down. The integrality gap proof of [4] works by proving that if all of these coin flips are independent (which they are not in [5]) then with high probability no cheap group steiner tree exists, while clearly a good fractional solution exists. The hardness result of [5] works by extending this in two ways. The first and easier of these is proving that if the coin flips are positively correlated then with high probability no cheap solution exists. The second extension, which is proved in Lemma 4.3 of the paper, is that if there are both positive *and* negative correlations then if a cheap solution exists, then for a lot of the "stars" (individual internal nodes and their immediate children in the tree) there are a lot of constraints (u, v) in the original Label Cover instance that have consistent suggested sets.

2 Steiner Tree

In the Steiner Tree problem we are given a metric (V, d) and a set $R \subseteq V$ of *required* vertices, also called *terminals*. We will typically think of this metric as the complete graph G on V where the edge between u and v has length d_{uv}. A Steiner tree is a set $E \subseteq \binom{V}{2}$ of edges such that all of the terminals fall into the same connected component of $G[E']$ (the graph restricted to edges in E'). The goal is to minimize $\sum_{e \in E'} d_e$. Note that if $R = V$ then this is just the problem of computing the Minimum Spanning Tree, which can be solved in polynomial time. But for general R the problem is NP-hard.

Consider the following MST heuristic: ignore all vertices in $V \setminus R$ (the *steiner* points), and compute a MST on R. Let $n' = |R|$ be the number of terminals.

Theorem 2.1. *The MST heuristic is a* $2(1 - \frac{1}{n'})$*-approximation to the minimum Steiner tree.*

Proof. Let T be the optimal Steiner tree. Consider an Euler tour of T. Since this tour uses each edge twice, its length is twice the length of T. Now shortcut this tour into a cycle on the terminals. We know from the triangle inequality that shortcutting does not increase the length, so the total length of this cycle is at most twice the length of T. Note that this is a cycle on n' vertices, so the longest edge in it has length at least $\frac{1}{n'}$ of the total length. So by removing this edge, we get a path on the terminals of length at most $2(1 - \frac{1}{n'})length(T)$. Since this is a path, by definition it is a tree, and so the MST has length at most the length of the path. Thus the MST has length at most $2(1 - \frac{1}{n'})length(T) = 2(1 - \frac{1}{n'})\mathsf{OPT}$. □

As an aside, note that the cycle we found in this analysis is also a 2-approximation to the optimal traveling salesman tour. In 1976 a modification of this was used by Christofides to give a $3/2$-approximation for TSP.

The best known hardness for Steiner tree is a constant:

Theorem 2.2 (Bern and Plassmann [1])**.** *There exists some* $\epsilon > 0$ *such that the Steiner Tree problem is* $(1 + \epsilon)$*-hard.*

While this theorem rules out the existence of a PTAS, the current best ϵ is very small, so this hardness is rather weak. The current best upper bound does not come close to matching it:

Theorem 2.3 (Robins and Zelikovsky [7])**.** *There is a* $1 + \frac{\ln 3}{2} \approx 1.55$ *approximation for the Steiner Tree problem*

The intuition behind their result is the following. Define a k-restricted steiner tree to be a steiner tree with the property that if all of the terminals are removed, then each connected component has at most k vertices (obviously all of which are steiner nodes). Robins and Zelikovsky's algorithm finds close to optimal k-restricted steiner trees, and then argues that the best k-restricted steiner tree is a good approximation to the best steiner tree.

2.1 LP Relaxation

We say that a set $S \subseteq V$ *crosses* the set R if $S \cap R \neq \emptyset$ and $R \setminus S \neq \emptyset$, i.e. if there is at least one terminal in S and at least one terminal not in S. Let $\mathcal{S}$ be the collection of all sets that cross R. Also, for any set $S \subseteq V$ we let ∂S be the set of edges with one endpoint in S and one endpoint in $V \setminus S$. We use the following LP relaxation of the Steiner Tree problem, which has a variable for every edge and a constraint for every crossing set. It is easy to see that the integer version of this is indeed a formulation of the Steiner Tree problem, and thus the LP relaxation will have cost at most the cost of the optimal steiner tree.

$$
\begin{aligned}
\min \sum_{e \in \binom{V}{2}} & d_e x_e \\
\text{s.t.} \quad \sum_{e \in \partial S} & x_e \geq 1 \qquad \forall S \in \mathcal{S} \\
& x_e \geq 0 \qquad \forall e \in \binom{V}{2}
\end{aligned}
$$

How is this relaxation for the special case of $R = V$, i.e. the MST problem? Consider the n-cycle. The MST has cost $n-1$, but we could set $x_e = 1/2$ for all edges on the cycle to get a fractional solution of value $n/2$. Thus the integrality gap is at least $2(1 - \frac{1}{n'})$. In the MST case, though, we can add the extra constraints that $\sum_{u,v \in X} x_{uv} \leq |X| - 1$ for all $X \subseteq V$ and that $\sum_{u,v \in \binom{V}{2}} x_{uv} = n-1$. This results in an integral polytope, giving yet another proof that computing the MST is in P.

Back in the Steiner tree case, we claim that this LP relaxation still has an integrality gap of at most 2. To prove this we will give a primal-dual algorithm that is a 2-approximation. The dual to the Steiner Tree LP is:

$$
\begin{aligned}
\max \sum_{S \in \mathcal{S}} & y_S \\
\text{s.t.} \quad \sum_{S : \partial S \ni e} & y_S \leq d_e \qquad \forall e \in \binom{V}{2} \\
& y_S \geq 0 \qquad \forall S \in \mathcal{S}
\end{aligned}
$$

Consider the following simple primal-dual algorithm. Initially set all of the dual variables to 0, and set the n' singleton sets consisting of just one element of R to be "active". All other crossing sets are "inactive". Uniformly raise the dual values of all active sets, until some dual constraint becomes tight. This tight constraint corresponds to an edge e, so we include e in our primal solution. We also set to inactive all of the sets that contributed to this constraint, and make their union active. The algorithm ends when there is only one active set. This obviously results in a tree, since the active sets are just the connected components of the graph that the algorithm builds and all edges are added between active sets.

We claim that this is a $2(1 - \frac{1}{n'})$-approximation. To see this, we associate a bank account with each crossing set. Initially all accounts are empty. When we start raising the dual values of the singleton sets, their bank account increase at the same rate, so at time t each account has t in it. When the first constraint (say corresponding to edge e) becomes tight, it is because the two singleton sets containing the endpoints of the edge have a combined value of d_e in their bank accounts, i.e. each one has value $d_e/2$. So doubling one of their accounts pays for the edge that we buy, and the other account is transferred to the new set created by their union. So after the first edge is bought, it is still true that every set with a bank account has value equal to the current time.

Assume that at some point in the algorithm every active set has an account equal to the time. Notice that for every edge e that does not have both endpoints contained by a single active set, there are exactly two active sets that contribute to its constraint (the sets containing each of its endpoints). This is easy to prove by induction. So when the next constraint becomes tight (say corresponding to edge e) there are two active sets that contributed to it. Thus we can again pay for half of the edge with one of the accounts, and give the other one to the new active set consisting of their union. So at all times in the algorithm every active set has an account equal to the time, and we have paid for half of every edge that was bought. If the algorithm ends at time t, then we also have t money left over.

Note that the total amount of money raised equals $\sum_{S \in \mathcal{S}} y_S$, which equals the value of the dual solution OPT_{dual}. And this money, minus the t left over at the end, was enough to pay for half of every edge that we bought. Thus the cost of the solution is at most $2(\mathsf{OPT}_{dual} - t)$. Since there are at most $n' = |R|$ active sets at any point, we know that the total money raised is at most $n't$, and thus $t \geq \mathsf{OPT}_{dual}/n'$. So the total cost of the solution is at most $2(\mathsf{OPT}_{dual} - \mathsf{OPT}_{dual}/n') = 2(1 - \frac{1}{n'})\mathsf{OPT}_{dual} \leq 2(1 - \frac{1}{n'})\mathsf{OPT}_{LP} \leq 2(1 - \frac{1}{n'})\mathsf{OPT}$, as claimed.

2.2 Bidirected Cut Relaxation

Another formulation and relaxation of the Steiner tree problem is the *bidirected cut* relaxation. We view every edge as two oppositely directed arcs, each with the same length as the edge. We arbitrarily choose some $r \in R$ to be the *root*, and require a Steiner tree to be directed towards the root (i.e. every terminal must have a directed path to the root). Obviously a Steiner tree in the original undirected version is still a valid solution in the directed setting, since we can just choose the appropriately directed arc corresponding to each edge. For any set S, let $\partial^+ S$ be the set of arcs that leave S. The bidirected cut relaxation is just the directed version of the basic relaxation, so there are $2\binom{n}{2}$ variables (one for each arc) and a constraint for every set that contains at least one vertex from R but not the root. Let A be the set of arcs (all ordered pairs of nodes except loops), and let $\mathcal{C} = \{S \subseteq V : S \cap R \neq \emptyset \wedge r \notin S\}$.

$$
\begin{aligned}
\min \sum_{e \in A} d_e x_e & \\
\text{s.t.} \quad \sum_{e \in \partial^+ S} x_e \geq 1 & \qquad \forall S \in \mathcal{C} \\
x_e \geq 0 & \qquad \forall e \in A
\end{aligned}
$$

Interestingly, the integrality gap of this formulation is unknown. The best lower bound on it is $8/7$ (due to Goemans), but there are currently no algorithms using this relaxation that do better than $2(1 - \frac{1}{n'})$. On the other hand, in the special case of *quasi-bipartite* graphs (in which there are no edges between steiner nodes) this relaxation can be used to obtain a $3/2$-approximation [6].

3 Extensions of Steiner Tree

3.1 Online Steiner Tree

In the online setting the vertices arrive one at a time, and we always have to maintain a Steiner tree. We also have to keep what we have already bought, so the final solution will be the union of the Steiner trees from every time point. We define the *competitive ratio* or an algorithm in the usual way, as the maximum over input sequences of the cost incurred by the algorithm divided by the cost of the optimal offline tree. The obvious algorithm in this setting is the greedy algorithm: when a new node arrives, connect it to the existing tree in the cheapest possible way.

Theorem 3.1. *Greedy is $O(\log n)$-competitive.*

Proof. For ease of exposition, we will assume that n' is a power of 2, although this assumption can be removed with some extra technical work. Let T be the optimal Steiner tree, and consider the Euler tour of the tree. Shortcut this tour to get a cycle on just the required vertices. Obviously the cost of this cycle is at most $2\mathsf{OPT}$, since shortcutting does not increase the cost and the Euler tour uses every edge in the tree twice. Since n' is even, this cycle has even length, so we can break it into two alternating matchings. Thus at least one of these matchings has cost at most OPT. Let M be this matching.

Consider some $(u, v) \in M$. Without loss of generality, assume that u arrived before v. Thus when v arrived, it could have chosen this edge. Since this argument holds for one of the vertices in every edge of the matching, we know that the cost the greedy algorithm incurs due to these $n/2$ vertices is at most the cost of the matching, which is at most OPT. But now we can repeat this analysis on the remaining $n/2$ nodes: an Euler tour of the optimal tree costs at most $2\mathsf{OPT}$, shortcutting it to just the remaining required nodes still costs at most $2\mathsf{OPT}$, so there is a matching of the remaining required nodes that costs at most OPT, so there are $n/4$ nodes that together cost at most OPT. We can do this again on the remaining $n/4$ nodes, and keep doing it until there are no more remaining nodes, which will happen after $O(\log n)$ rounds. Since each round costs at most OPT, this implies that the greedy algorithm is $O(\log n)$-competitive. □

3.2 Priority Steiner Tree

In this variation the set of required vertices R is the disjoint union of some subsets R_i, so $R = R_1 \uplus R_2 \uplus \cdots \uplus R_k$. We think of the R_i's as priority levels, so the vertices in R_1 have a higher priority than vertices in R_2, etc. The edge set is similarly divided, so $E = E_1 \uplus \cdots \uplus E_k$. Here we think of E_1 as the highest quality edges, E_2 as the second highest quality, etc. The goal is to output a set of edges $E' = E'_1 \uplus E'_2 \uplus \cdots \uplus E'_k$ of minimum cost such that all of the vertices in R_i can connect to the root r using only the edges of $E_1 \uplus \cdots \uplus E_i$. More formally, every vertex in $R_i \cup \{r\}$ must be in the same connected component of the graph $(V, E'_1 \uplus \cdots \uplus E'_i)$ for all $i \in [k]$. This model was first studied by Charikar, Naor, and Schieber [2]

Theorem 3.2. *There is an $O(k)$-approximation algorithm for Priority Steiner Tree*

Proof. Let $G_i = (V, E_1 \uplus \cdots \uplus E_i)$, and let $\mathcal{I}_i$ be the instance of Steiner tree with graph G_i and required vertices $R_i \cup \{r\}$. Note that the optimal steiner tree for $\mathcal{I}_i$ has cost at most OPT, since the optimal priority Steiner tree must contain as a subtree a valid Steiner tree for $\mathcal{I}_i$. So by computing an $O(1)$ approximation (e.g. form Theorem 2.1 or Theorem 2.3) to each of the subproblems $\mathcal{I}_i$ for all $i \in [k]$, and then taking the union of the solutions, we get a priority Steiner tree of cost at most $O(k)\mathsf{OPT}$. While taking the union may cause some cycles, it is easy to see that any cycles can be broken while maintaining feasibility and without increasing the cost. □

Theorem 3.3 (Charikar et al. [2]). *There is an $O(\log n)$-approximation algorithm for Priority Steiner Tree*

Proof. We give only a sketch of the proof. Order the vertices in R by their priority, so every node in R_1 is before every node in R_2, which is before every node in R_3, etc. Then give this sequence as input to the online greedy algorithm, but when it is in the part of the sequence corresponding to R_i only let it use edges in $E_1 \uplus \cdots \uplus E_i$. The same argument as in Theorem 3.1 basically works, although some extra care has to be taken since the graph itself is changing during this process due to the allowable priorities. □

A weaker hardness result is also known:

Theorem 3.4 (Chuzhoy et al. [3]). *Priority Steiner Tree cannot be approximated better than $\Omega(\log \log n)$ unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{O(\log \log \log n)})$*

Using a similar construction, it is known that the natural LP relaxation has an integrality gap of $\Omega(\frac{\log n}{\log \log n})$.

References

[1] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2,. *Inf. Process. Lett.*, 32(4):171–176, 1989.

[2] M. Charikar, J. S. Naor, and B. Schieber. Resource optimization in qos multicast routing of real-time multimedia. *IEEE/ACM Trans. Netw.*, 12(2):340–348, 2004.

[3] J. Chuzhoy, A. Gupta, J. S. Naor, and A. Sinha. On the approximability of some network design problems. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 943–951, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[4] E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan, and N. Wang. Integrality ratio for group steiner trees and directed steiner trees. *SIAM J. Comput.*, 36(5):1494–1511, 2006.

[5] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *STOC '03: Proceedings of the thirty-fifth annual ACM Symposium on Theory of Computing*, pages 585–594, New York, NY, USA, 2003. ACM.

[6] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. In *SODA '99: Proceedings of the tenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[7] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.