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

Lecture 11: Group Steiner Tree Gaps and Hardness

February 19, 2008

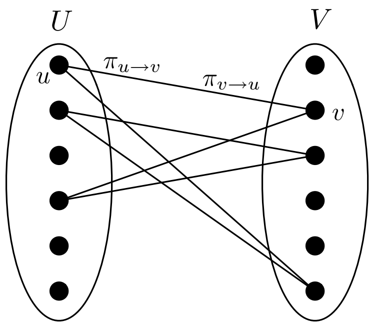
Lecturer: Anupam Gupta *Scribe: Kanat Tangwongsan*

In this lecture, we review hardness results derived from the consistent-labeling family and start to prove the $\Omega(\log^{2-\varepsilon} k)$ hardness for group steiner tree. First, we revisit the hardness proof of Max-k-Coverage and outline a proof of Min-Set-Cover. Then, we consider a direct translation between height-1 Group Steiner Tree and Set Cover, and use it to set up the reduction for proving the $\Omega(\log^{2-\varepsilon} k)$ hardness. Finally, we discuss the integrality gap.

1 Recap: Consistent-Labeling and Hypercubes

So far, we have seen a number of hardness proofs using reductions from problems in the consistent-labeling family. In this lecture, we will focus on reductions from a member of this family, namely (K, L) binary consistent labeling.

Problem Definition. An instance of the (K, L) *binary consistent labeling* problem consists of a d-regular bipartite graph $\mathcal{G} = (U, V, E)$ together with functions $\pi_{u\to v}\colon L \to K$ and $\pi_{v\to u}\colon L \to K$ for all edges $uv \in E$. We assume $|U| = |V| = n$.



Given such an instance, we want to find a labeling function $f : U \cup V \to L$ that maximizes the fraction of edges satisfied; we say that an edge $e \in E$ is satisfied if and only if $\pi_{u\to v}(f(u)) = \pi_{v\to u}(f(v))$. The problem of (K, L) binary consistent labeling is a special case of the Consistent-Labeling problem we saw on Homework #1 and is essentially the same as Label-Cover.

How did we use Consistent-Labeling? In this class thus far, most reductions from Consistent-Labeling make use of Boolean hypercubes $\{0, 1\}^{|K|}$. Our universe B consists of smaller sub-universes, each of which is a Boolean hypercube $\{0, 1\}^{|K|}$. Typically, given $\mathcal{G} = (U, V, E)$, the universe B is constructed by taking the disjoint union of B_e for all edges $e \in E$, where B_e is a copy of $\{0, 1\}^{|K|}$.

In Max-k-Coverage and Set-Cover, the key idea is to produce a set of "cuts" for each subuniverse such that unless the two pieces of the same cut are picked, we still leave that subuniverse uncovered. We accomplish this by defining the following sets of dictators on each subuniverse: for each $a \in K$,

$$D_a^{uv} = \{x \in B_{uv} : x_a = 1\}.$$

We call the sets D_a^{uv} and $\overline{D}_a^{uv}$ (the set complement of D_a^{uv} w.r.t. B_{uv}) a *complement pair.* Clearly, every complement pair partitions the hypercube $\{0,1\}^{|K|}$ exactly in half. Furthermore, by construction, we know that any $t \leq |K|$ sets containing no complement pair can cover at most $1 - 2^{-t}$ fraction of the hypercube!

Abstract Universes. In many cases, it is probably more convenient or even crucial to work with an abstract universe, rather than a collection of Boolean hypercubes. Suppose we are given an abstract universe $A = \biguplus_e A_e$, where the subuniverses all have the same size. Like before, we want the property that unless both sets in a complement pair are picked, picking a large number of sets still won't cover the whole subuniverse. For this, we will generate $|K|$ random cuts on A_{uv}. For each $a \in K$, we construct the set $D_a^{uv} \subseteq A_{uv}$ such that $\mathbf{Pr}[x \in D_a^{uv}] = 1/2$ for all $x \in A_{uv}$; therefore, each dictator set divides the subuniverse roughly in half. It can be shown that the dictator family $\{D_a^{uv} : a \in K\}$ has the desired property with high probability.

1.1 Hardness of Max-k-Coverage

We revisit the hardness proof of Max-k-Coverage presented in Lecture 3. The familiar Max-k-Coverage problem is the following. Given a family of sets $S_1, \ldots, S_m$ and a number $k \leq m$, we want to pick k sets to maximize coverage. We already saw a proof of 1 vs. $3/4 + \varepsilon$ hardness via a reduction from consistent labeling, and improved it to 1 vs. $1 - (1 - 1/k)^k + \varepsilon$ in Homework #1. The basic idea is as follows.

Starting with a Consistent-Labeling instance $\mathcal{G} = (U, V, E)$, we construct a Max-k-Coverage instance $\mathcal{H}$ based on the dictator-set families outlined above. Our universe A consists of nd subuniverses, one for each edge $uv \in E$. Each subuniverse A_{uv} is a block of binary strings $\{0,1\}^{|K|}$. Hence, we have a total of $nd2^{|K|}$ elements in the universe.

Every vertex-label pair has a corresponding set. We construct these sets by examining the π function and combining together appropriate dictator sets. For each $u \in U$ and $\alpha \in L$, let $S_{u,\alpha} = \bigcup_{v:uv\in E} D^{uv}_{\pi_{u\to v}(\alpha)}$. For $v \in V$ and $\alpha \in L$, $S_{v,\alpha} = \bigcup_{u:uv\in E} \overline{D}^{uv}_{\pi_{v\to u}(\alpha)}$.

Finally, set $k = 2n$. Now we can show the following.

- **Completeness.** If $\mathsf{Opt}(\mathcal{G}) = 1$, then $\mathsf{Opt}(\mathcal{H}) = 1$. That is, we can cover all elements using $2n$ sets. It is easy to see that the collection $\{S_{v,f(v)} : v \in U \cup V\}$ covers all elements.
- **Soundness.** If $\mathsf{Opt}(\mathcal{G}) < \eta$, then $\mathsf{Opt}(\mathcal{H}) < 3/4 - \varepsilon(\eta)$. The idea is to show the contrapositive version. Suppose $\mathsf{Opt}(\mathcal{H}) \geq 3/4 - \varepsilon(\eta)$. Then we show that on a good fraction of edges (say, $\geq \varepsilon/20$), the number of matching suggestions on each node (say, $\leq 10/\varepsilon$) is small. Therefore, suppose we "decode" the solution by assigning a label from the suggestion uniformly at random (or arbitrarily at random if the suggestion is empty). We will satisfy $\eta := \varepsilon^3/2000$ fraction of the edges in expectation, which by an averaging argument, implies that the original Consistent-Labeling instance has a labeling which satisfies at least η fraction of the edges. This then indicates that $\mathsf{Opt}(G) \geq \eta$.

What if we have an abstract universe? If we replace our universe with an abstract one, we can still have the same completeness lemma; however, the soundness lemma will instead be, *with high probability*, $\mathsf{Opt}(\mathcal{G}) < \eta \implies \mathsf{Opt}(\mathcal{H}) < 3/4 - \varepsilon(\eta)$.

2 Hardness of Set Cover

In this section, we look at the outline of a proof that there is no factor $\Omega(\log n)$ algorithm for Set-Cover unless $\mathrm{NP} \in \mathrm{DTIME}(n^{O(\log \log n)})$. We will flesh out the details in Homework #3.

The proof is via a reduction from Consistent-Labeling. Given a Consistent-Labeling instance $\mathcal{G}$, we produce a Set-Cover instance $\mathcal{H}$ using our Max-k-Coverage construction. We will have to prove two lemmas: completeness and soundness.

- **Completeness Lemma.** If $\mathsf{Opt}(\mathcal{G}) = 1$, then $\mathsf{Opt}(\mathcal{G}) \leq 2n$. This is the easy part: we pick the same sets as in Max-k-Coverage.
- **Soundness Lemma.** If $\mathsf{Opt}((G)) < \eta$ (roughly, $\eta = 1/\lambda^2$), then $\mathsf{Opt}(\mathcal{H}) \geq 2n \cdot \lambda$ (for some large λ). Like before, we will show the contrapositive version: $\mathsf{Opt}(\mathcal{H}) < 2n \cdot \lambda \implies \mathsf{Opt}(\mathcal{H}) > \eta$. Suppose $\mathsf{Opt}(\mathcal{H}) < 2n \cdot \lambda$. The basic idea of the proof is as follows. Since there are $2n$ vertices, the average number of labels per vertex is λ. Consider an edge uv: if a collection of λ sets can cover A_{uv} entirely, then there must be a pair that matches up. Therefore, we satisfy uv with probability $p \approx 1/\lambda^2$.

In the homework, we will translate this informal idea into a proof. A common mistake is to miscalculate the universe size; to avoid this pitfall, we might want to define and use $N = n^{O(\log \log n)}$ in our proof.

3 Hardness of Group Steiner Tree

We will now start a hardness proof of group steiner tree. The proof is due to Halperin and Krauthgamer [2] from STOC'03. First, let us recall the group steiner tree problem.

Given an weighted undirected tree $T = (V, E, w)$, a collection of subsets (called *groups* or *colors*) $g_1, \ldots, g_k \subseteq V$, and a *root* note r, the group steiner tree problem is to find a subtree of T of minimum weight such that at least one vertex from each group is connected to the root.

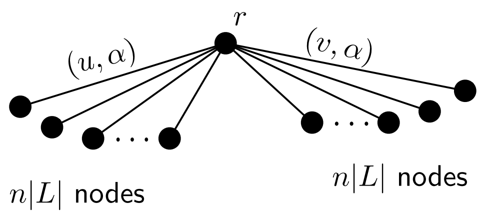
A special case of group steiner tree is when the tree has height 1. As seen in last lecture, this problem has a direct correspondence with the Min-Set-Cover problem: a set-cover instance can be represented as this special case by having edges of the tree correspond to sets and colors correspond to the ground elements. Therefore, the vertex incident to the edge for S_i has colors corresponding to the elements in S_i.

3.1 Reduction to Tree of Height $= 1$

This relationship between group steiner tree and set cover motivates the following reduction. Like in the set-cover proof, the reduction is from (K, L) binary consistent labeling.

The reduction takes as input $G = (U, V, E)$ and an abstract universe $A = \biguplus A_{uv}$, and produces a tree $\mathcal{T}_1$ of height of 1. We present two views of the same construction. The first view applies the direct translation from the familiar set-cover reduction, whereas the second view is a random process performed directly on the tree.

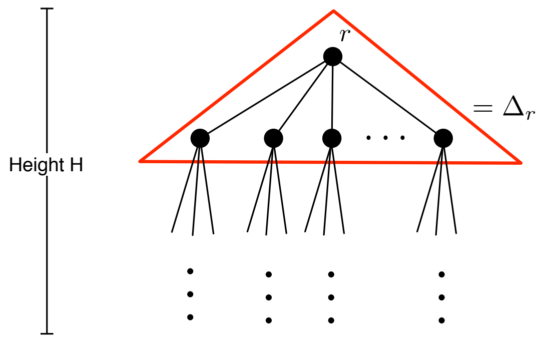
View 1. The first step in the process is to construct a set-cover instance like in Section 1.1. Note that the set system has $2n|L|$ sets, and the ground elements are exactly the universe A. In the group-steiner world (shown in a figure below), this set-cover instance corresponds to a tree of height 1 containing a special node r designated as the root and $2n|L|$ leaves corresponding to the vertex-label pairs.



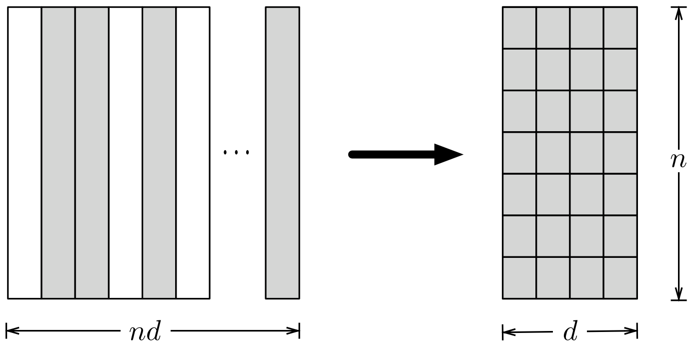
View 2. We can also do the same construction directly on trees. Start off with a tree like above, with all elements of the universe initially at the root r. For each element $x \in A_{uv}$, we duplicate x for a total of $|L|$ copies, one for each label in L; we refer to them as $\{x^{(\ell)} : \ell \in L\}$. Then flip a fair coin for each $x^{(\ell)}$: if it comes up heads, we push it down the edge $(u, \pi_{u \to v}(\ell))$; otherwise, we push it down $(v, \pi_{v \to u}(\ell))$.

3.2 Building an H-level Tree

Now we will construct an H-level tree $\mathcal{T}_H$, in which each non-leaf node has $2n|L|$ children. We will still refer to the edges by the ordered-pair (u, α), where $u \in U \cup V$ and $\alpha \in L$. The process can be thought of as a recursive composition of multiple instances of height-1 tree. First, let us consider the universe (and subuniverses) we will be using at each level.



Universes. In level 1, our universe A consists nd subuniverses, one for each edge: $A = \biguplus_{uv} A_{uv}$. In other words, we partition the universe A into nd subuniverses of equal size. Let denote by u_i the size of the universe at level i. Thus, we have $u_1 = |A|/(nd)$. For each subuniverse in level 1, we further partition it into n subuniverses in level 2, so $u_2 = u_1/n = |A|/(n^2 d)$. In general, each subuniverse in level h is further partitioned into n subuniverses in level $h+1$; thus, we have $u_h = |A|/(n^h d)$. For technical reasons (which will become apparent in subsequent lectures), we make $u_H = n^H \cdot d$ and therefore $|A| = u_H^2$.



For all non-leaf nodes of $\mathcal{T}_H$, we denote by Δ_v the height-1 tree rooted at v. Our goal is to make each Δ_v look essentially like a $\mathcal{T}_1$ instance of the previous section. For this, we need nd "relevant" subuniverses at each node to form a universe that our construction for $\mathcal{T}_1$ applies. Clearly, the root node has nd level-1 subuniverses as we want. Suppose we name the nd subuniverses at level 1 using the elements of E, i.e. $\{A_{uv} : uv \in E\}$ are level-1 subuniverses. Now consider a node w at level 1. It must be incident to some edge (u, α). We know that in the $\mathcal{T}_1$ construction, elements from exactly d subuniverses, namely A_{uv} for $uv \in E$, can be pushed down this edge. Taking these d level-1 subuniverses and dividing each of them equally into n pieces, we have nd sub-subuniverses ready for level 2. In general, this argument can be inductively applied to show that at any non-leaf node, we have nd "relevant" subuniverses to form a useful universe.

Edge Costs. Every edge in level h has cost w^h, where $w = \frac{1}{2n}$.

Assigning Colors. Given the work we have done, we know that each Δ_v looks like a $\mathcal{T}_1$ instance; therefore, we apply the $\mathcal{T}_1$ construction on each Δ_v, all *independently*. This process will give us a set of colors at every node (except the root), but in group steiner tree, only the leaves are given colors. To finish the construction, we assign colors to the leaf nodes as follows: a leaf u has color g if and only if every node along the path from root to u, including u itself, contains color g (viewed from the perspective of height-1 trees).

An Alternative View. In effect, we could have applied the process in View 2 recursively and arrive at the same construction. That is, we start with the whole universe at the root r (level 0). Using the process in View 2, we replicate and push elements of the universe down to level 1. Then we repeat the same process for each node in level 1 but with only the elements present at that node.

3.3 Completeness and Soundness

Lemma 3.1 (Completeness). *If* $\mathsf{Opt}(\mathcal{G}) = 1$, *then* $\mathsf{Opt}(\mathcal{T}_H) = H$.

Proof. Suppose that $\mathsf{Opt}(\mathcal{G}) = 1$, so there is a labeling function f that satisfies all of the edges. We will choose the following edges in the $\mathcal{T}_H$ instance: in level i, we choose $(u, f(u))$ for all $u \in U \cup V$. To see that this is a valid solution, consider any element $x \in A_{uv}$. At each level, one copy of x went down either (u, α) or (v, α), where $\alpha = \pi_{u \to v}(f(u)) = \pi_{v \to u}(f(v))$, depending on the outcome of the coin-flip. Therefore, we can inductively argue that a copy of x has a path to the root in our solution. The cost of our solution can be analyzed level by level. Clearly, in this process, exactly $(2n)^i$ edges are picked in level i, incurring a cost of $w_i \cdot (2n)^i = 1$. Summing up all H levels, we have that the total cost is H. $\square$

The Decoding Algorithm. The soundness proof is more involved and will be discussed next time. We describe here an intuition behind the decoding algorithm for group steiner tree.

Let Δ_v denote the height-1 tree rooted at v. We say that Δ_v is *occupied* if and only if at least of the edges is chosen. Note that each Δ_v is (can be translated into) a set-cover instance. The decoding algorithm is roughly as follows: first pick a height uniformly at random. Then, within that height, pick uniformly at random an occupied Δ_v, and decode it just like a set-cover instance.

4 Integrality Gap for Group Steiner Tree

The integrality gap of $\Omega(\log^2 k)$ was proved by Halperin et al. [1] (Integrality Ratio for Group Steiner Trees and Directed Steiner Trees). Here's the idea.

1. Create a balanced δ-ary tree with height $H + 1$ levels, where the edge costs decrease exponentially: edges at level h cost $1/2^h$. (Hence edges incident to the root cost $1/2$, and edges incident to the leaves cost $1/2^H$.)

2. The arity of the tree is $\delta = c \log n$ for some constant c. So the height is $H = \log n / \log \delta$. The number of leaves is n, the total number of nodes is $n(1 + o(1))$. The number of groups/colors/elements is $k = 2^{2H}$.

3. Do the following independently for each of the k elements x. The element x starts off at the root. For each of the δ child edges, it flips a fair coin independently: if the coin comes up heads, a copy of x slides down that edge, where it begins such a random process anew. Note that the expected number of copies of x at level i is $(d/2)^i$. The group for x is the set of all the leaves that contain a copy of x.

Small fractional solution. It is not that difficult to show that this instance has a fractional solution of value $O(H)$ with high probability (where the probability is over the random choice of groups above):

- Consider some node at level i in this tree. Conditioned on this node getting a copy of x, the expected number of leaves containing x below it is $\mu_h = (d/2)^{H-h}$. With high probability, the actual number lies in $[\mu_h/3, 3\mu_h]$.
- For an edge e going from level $i-1$ to level i, set $x_e = 9/\mu_e$ — this is a feasible solution (whp) with cost $9H$.

No small integer solution. Next, we show that "with high probability, all integer solutions have cost $\Omega(H^2 \log k) = \Omega(H \log^2 k)$." This proves the $\Omega(\log^2 k)$ integrality gap for the problem.

- Since each leaf-edge costs $1/2^H$, no tree of cost C can use more than $C2^H$ of these. There are d^H leaf-edges, so the number of possible trees of cost C is $\binom{d^H}{C2^H} \leq \exp(CH2^H \log d)$. Call this value $M(C)$.
- Suppose $\mathsf{Fail}(C)$ is the probability that for a fixed tree of cost C, it misses the group for x created above. By independence of groups, the probability that this fixed tree hits all the groups is at most $(1 - \mathsf{Fail}(C))^k \leq \exp(-k \times \mathsf{Fail}(C)) = \mathsf{Hitall}(C)$.
- Hence, taking a trivial union bound over all trees, the chance that *there exists some tree* that hits all the groups is at most $M(C) \times \mathsf{Hitall}(C)$.
- If this product is $o(1)$ for some C, no tree of cost C is feasible with probability $1 - o(1)$. We need to show this holds true for some constant $\delta > 0$ and $C = \delta H^2 \log k$.

Now for the most technical part. One needs to prove that for some fixed tree of cost $\delta H^2 \log k$, and some x, none of the leafs in this tree are hit by the random process above. The proof strategy seems natural in hindsight, but also somewhat delicate: details in Section 2.3.1 of the paper above. (Anupam's question: can one get a slightly worse bound but without the case analysis at the end?)

This analysis (and extensions of it) is actually used for the soundness part of the hardness proof!

References

[1] 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, 2007.

[2] 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.