601.435/635 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: Steiner Tree, TSP **Date:** 1/31/19
Scribe: Michael Dinitz

Today we're going to talk about two problems: Steiner Tree and Traveling Salesperson (TSP). For both of these we'll design relatively simply approximation algorithms based on ad hoc, combinatorial techniques. TSP is probably the more famous of the two, but they're both fundamental and extraordinarily important. I am particularly partial to Steiner Tree, as it is the primary example of a broad class of problems known as *network design problems*, where we are given a graph and are asked to find the cheapest subgraph that satisfies some type of connectivity constraint. These types of problems actually form a large fraction of my research, so I really like them.

1.1 Steiner Tree

Recall the Minimum Spanning Tree problem from undergraduate algorithms:

Definition 1.1.1 *The* Minimum Spanning Tree *problem is defined as follows:*

- *Input:*
  - *Graph* $G = (V, E)$
  - *Costs* $c : E \to \mathbb{R}^+$
- *Feasible solutions:* $F \subseteq E$ *such that* F *is connected and spans all vertices*
- *Objective function: minimize* $\sum_{e \in F} c(e)$

Note that since we are minimizing the cost, without loss of generality the solution is a tree (if some solution has a cycle then we can remove any edge from the cycle and have a subgraph of lesser cost which still spans V). So we do not need to include " no cycles" in the definition of feasible.

As you should know, there are many polynomial-time algorithms which solve MST (Kruskal's, Prim's, Boruvka's, etc.). So MST is not NP-hard, and there's not much reason to think about approximation algorithms for it. But if we generalize the problem slightly, so instead of spanning *all* vertices we only need to span a given set of vertices, then we get the following problem which *is* NP-hard.

Definition 1.1.2 *The* Steiner Tree *problem is defined as follows:*

- *Input:*
  - *Graph* $G = (V, E)$
  - *Costs* $c : E \to \mathbb{R}^+$
  - *Terminals* $T \subseteq V$

- *Feasible solutions: $F \subseteq E$ so that F is connected and spans all of T.*
- *Objective function: minimize $\sum_{e \in F} c(e)$*

Clearly when $T = V$ this is exactly the MST problem. Interestingly, if $T = \{s, t\}$ then this is the shortest $s - t$ path problem! So we are in some sense "interpolating" between two easy problems, which gives us a hard problem. And note that again, since we are minimizing cost the solution is WLOG a tree.

In order to approximate Steiner Tree, we will first consider a variant/special case.

Definition 1.1.3 *The* metric completion c' *of (G, c) is the metric on V in which for all $u, v \in V$, the distance $c'(u, v)$ is the cost of the shortest $u - v$ path in G under edge costs c.*

In other words, in the metric completion there is an edge between every two nodes, and the cost of this edge is the cost of the shortest-path in the original graph under the original costs. Note that this gives a metric space, i.e., $c'(u, v) \leq c'(u, w) + c'(w, v)$ for all $u, v, w \in V$.

The Metric Steiner Tree problem is the variant of Steiner tree in which we operate in a metric space rather than a graph. In other words, the input is a metric space (V, c) and a set of terminals $T \subseteq V$, and the goal is to compute the minimum cost tree spanning T. We will show that it is possible to reduce the general case to the metric case, so we will be free to just assume that we're in the metric case.

1.1.0.1 Reduction to Metric

In this reduction, let $(G = (V, E), c, T)$ be an instance of Steiner Tree, and let c' be the metric completion of (G, c). We're going to prove the following theorem.

Theorem 1.1.4 *If there is an α-approximation for Metric Steiner tree, then there is an α-approximation for Steiner tree.*

To prove this, we're going to show that approximating Metric Steiner Tree of (V, c', T) is just as good as approximating Steiner tree on (G, c, T). Let's start with the trivial direction.

Lemma 1.1.5 *Let H be a solution to the Steiner tree problem on (G, c, T). Then H is a solution to the Metric Steiner tree problem on (V, c', T) with $c'(H) \leq c(H)$.*

Proof: Clearly H is a solution to the metric version. By triangle inequality, $c'(u, v) \leq c(u, v)$ for all $\{u, v\} \in E$, and hence $c'(H') \leq c(H)$. ■

Now we show the other direction.

Lemma 1.1.6 *Let H' be a solution to Metric Steiner Tree on (V, c', T). Then there is a solution H to the Steiner tree problem on (G, c, T) with $c(H) \leq c'(H')$, and given H' we can find such an H in polynomial time.*

Proof: We first get a subgraph $\hat{H}$ of G by replacing each edge $e = \{u, v\}$ of H' with a $u - v$ path in G of length $C'(u, v)$ (some such path exists by the definition of C'). Note that this does not increase the cost. We then let H be an arbitrary spanning tree of $\hat{H}$. Since $\hat{H}$ spans every node that H' spans (and possibly more), this implies that H is a feasible solution to the original Steiner

tree problem. And since replacing an edge by the appropriate path does not increase the cost, we have that $c(H) \leq c(\hat{H}) \leq c'(H')$. Clearly all of this can be done in polynomial time. ■

Proof of Theorem 1.1.4: Let $\mathcal{A}$ be an α-approximation algorithm for the metric Steiner tree problem. We first use $\mathcal{A}$ on the instance (V, c', T) of metric Steiner tree to get some H' which spans T. We can use the polyomial-time algorithm of Lemma 1.1.6 to get a solution H to Steiner tree on (G, c, T). We know from Lemma 1.1.6 that H is feasible.

Let OPT_{metric} denote the optimal solution to the metric Steiner tree problem on (V, c', T), and let $OPT_{steiner}$ be the optimal solution to Steiner tree on (G, c, T). Then we know that $c(H) \leq c'(H')$ by Lemma 1.1.6, and by definition we know that $c'(H') \leq \alpha \cdot c'(OPT_{metric})$. Now the definition of OPT_{metric} implies that that $c'(OPT_{metric}) \leq c'(OPT_{steiner})$, and Lemma 1.1.5 implies that $c'(OPT_{steiner}) \leq c(OPT_{steiner})$. Putting this all together, we get that

$$\begin{aligned} c(H) &\leq c'(H') && \text{(Lemma 1.1.6)} \\ &\leq \alpha \cdot c'(OPT_{metric}) && \text{(definition of } \mathcal{A}) \\ &\leq \alpha \cdot c'(OPT_{steiner}) && \text{(definition of } OPT_{metric}) \\ &\leq \alpha \cdot c(OPT_{steiner}) && \text{(Lemma 1.1.5)} \end{aligned}$$

Since H is feasible and was compute in polynomial time, this implies that we have an α-approximation to Steiner tree. ■

1.1.1 Approximating Metric Steiner Tree

So from now on we will assume that we are solving the metric Steiner tree problem, i.e., the input is actually a metric space and not just a graph. How can we use this to design a simple approximation algorithm? Here's a simple proposal:

Algorithm 1 Metric Steiner Tree

Input: Set of points V, metric $c : V \times V \to \mathbb{R}^+$, terminals $T \subseteq V$.
Let F be minimum spanning tree on T (computed using your favorite polytime MST algorithm).
Return F.

Theorem 1.1.7 *This is a $2(1 - \frac{1}{|T|})$-approximation algorithm for Metric Steiner Tree.*

Proof: Let F be the solution returned by the algorithm. Clearly F is feasible, since it is connected and spans T. Let F^* be the optimal solution, so we want to prove that $c(F) \leq (2 - \frac{1}{|T|})c(F^*)$. Since F is the minimum spanning tree for T, it is sufficient for us to prove that there is *some* spanning tree $\hat{F}$ of T with $c(\hat{F}) \leq 2(1 - \frac{1}{|T|})c(F^*)$, since then we would also have that $c(F) \leq c(\hat{F})$ by definition and so would be done.

In order to do this, let's first make a standard definition from graph theory: a graph $G = (V, E)$ is *Eulerian* if there is a closed tour of G which traverses every edge exactly once. A well-known theorem in graph theory is that a graph is Eulerian if and only if every vertex has even degree.

Let $2F^*$ be F^* with two copies of each edge. Now all degrees are even, and thus $2F^*$ is Eulerian, so there is an Eulerian tour C of $2F^*$. This tour hits all of the nodes (multiple times), but we only care about hitting T. So we can obtain a cycle H on T by "shortcutting" C to T: informally, we walk the Eulerian path to get an ordering of all nodes (where we only consider the first time we see a vertex), and then build the cycle on T defined by this ordering (there are more details about this in Section 2.4 of the book).

Because c is a metric space, the cost of each edge in the cycle H is at most the cost of the path (in C) that it replaces. And because $2F^*$ is Eulerian, each edge is in at most one of these paths and hence no edge was double counted. Thus $c(H) \leq c(C) = c(2F^*) = 2C(F^*) = 2OPT$.

Since H is a cycle, we can remove the most costly edge to get a spanning tree $\hat{F}$ of T (in fact, a spanning path). By a simple averaging argument the most costly edge has cost at least $\frac{1}{|T|} \cdot c(H)$, and thus $c(\hat{F}) \leq (1 - \frac{1}{|T|}) \cdot c(H) \leq 2(1 - \frac{1}{|T|})OPT$. ■

Theorem 1.1.8 *There is an instance of metric Steiner tree in which the algorithm returns a solution of cost* $2(1 - \frac{1}{|T|}) \cdot OPT$.

Proof: Consider the metric defined by a star (one center and $n-1$ leaves) in which edges of the star have cost 1 (so the cost of leaf-leaf pairs is 2). The terminals are the leaves. Then the optimal solution is the star itself, which has cost $n-1$. But the algorithm finds a minimum spanning tree on the leaves, which clearly has cost $2(n-2) = 2(1 - \frac{1}{n-1})OPT$. ■

It turns out that better algorithms do exist. The above algorithm has been known for a long time, since (I think) 1968 [GP68]. So there has been a long series of papers which improved on this algorithm and gave better than 2-approxiamtions. The current best known is the following, which is based on a technique called "iterative randomized rounding" which we'll talk about much later in the semester:

Theorem 1.1.9 (Byrka, Grandoni, Rothvoß, Sanita [BGRS13]) *There is a* $(\ln(4) + \epsilon) \approx 1.39$*-approximation to Steiner Tree.*

The best known hardness of approximation is still pretty far from this:

Theorem 1.1.10 (Chlebík and Chlebíková [CC02]) *Assuming that* $\mathbf{P} \neq \mathbf{NP}$*, there is no polynomial-time approximation algorithm with approximation ratio better than* $96/95$.

1.2 TSP

Recall that in (metric) TSP the input is a set of points V with metric edge costs $c : V \times V \to \mathbb{R}^+$. A solution is feasible if it is a Hamiltonian cycle of V (a spanning cycle visiting every vertex exactly once). The objective is to minimize the cost of this cycle. We're only going to talk about the metric case here.

The first algorithm we will consider is similar to the algorithm we used for Steiner Tree.

Theorem 1.2.1 *This is a* $2(1 - \frac{1}{n})$ *approx.*

Proof: Clearly the algorithm runs in polynomial time and returns a valid tour. The only question is how good the tour is.

Algorithm 2 TSP: Attempt 1

Input: Set of points V with metric costs $c : V \times V \to \mathbb{R}^+$

Compute MST T.
Double all edges to get $2T$ (Eulerian), so let C be Eulerian tour.
Let H be shortcutted C (treat all nodes as terminals).
return H

Let H^* be the optimal solution, and let F be the path obtained by removing the heaviest edge of H^*. Then F is a spanning tree (actually spanning path) of V of cost $c(F) \leq (1-\frac{1}{n})c(H^*) = (1-\frac{1}{n})\cdot OPT$. But since T is the minimum spanning tree, we know that $c(T) \leq c(F)$. Thus we have that $c(H) \leq c(C) = c(2T) = 2c(T) \leq 2c(F) \leq 2(1 - \frac{1}{|T|}) \cdot OPT$. ■

While this is a simple algorithm, there is a better one due to Christofides [Chr76]. In fact, this was one of the first nontrivial approximation algorithms developed. The key idea is to note that the only reason we had to double all edges of T was to get an Eulerian graph (which we could then shortcut). What if there were a way to get an Eulerian graph of smaller cost?

To start, let's make a simple observation about degrees in graphs:

Lemma 1.2.2 *Let G be an arbitrary graph. Then there are an even number of nodes with odd degree.*

Proof: Let $d(v)$ denote the degree of v in G. Clearly $\sum_{v \in V} d(v) = \sum_{e \in E} 2 = 2|E|$, since adding the degrees counts every edge twice (once at each endpoint). And obviously $2|E|$ is even. Since the sum of the degrees are even, there must be an even number of nodes of odd degree. ■

So if we consider the MST T, there are an even number of nodes that are "blocking" T from being Eulerian. Let's just fix those nodes!

Definition 1.2.3 *A **perfect matching** of nodes S is a matching of size $\frac{|S|}{2}$ (i.e., everyone is matched).*

There is a very famous and classical algorithm to find maximum weight matchings in general graphs in polynomial time due to Edmonds [Edm65], so we will use the ability to find maximum weight matchings in general graphs as a black box.

We can now define Christofides's algorithm.

Algorithm 3 Christofides's algorithms

Compute MST T.
Let D be odd-degree nodes in T.
Let M be min-cost perfect matching of D.
Let C be Eulerian tour of $T + M$.
return H be the Hamiltonian tour obtained by shortcutting C.

We used the notation $T + M$ to denote the multigraph obtained by taking the union of T and M with multiplicities: if an edge appears in both T and in M then it appears twice in $T + M$.

Theorem 1.2.4 *Everything is well-defined.*

Proof: We know from Lemma 1.2.2 that $|D|$ is odd, and since we are in a metric there is an edge between all pairs, so the min-cost perfect matching M is well-defined (we can break ties arbitrarily). Consider some node v. If it has even degree in T then it has even degree in $T + M$, while if it had odd degree in T then it has even degree in $T + M$. Thus $T + M$ is Eulerian, so C exists. ■

Theorem 1.2.5 *Christofides's algorithm is a 3/2-approximation.*

Proof: Theorem 1.2.4 implies that the algorithm returns a feasible tour, and clearly it runs in polynomial time if we use polynomial-time algorithms to compute the MST and perfect matching.

Let H^* be the optimal solution. Then H^* spans V and T is the least expensive way of spanning T. So $c(T) \leq OPT$ (this is the same as in the previous algorithm). So we simply need to show that $c(M) \leq \frac{1}{2}OPT$, since then we would have

$$c(H) \leq c(C) = c(T) + c(M) \leq OPT + \frac{1}{2}OPT = \frac{3}{2}OPT.$$

To upper bound the cost of M, define H_D to be H^* shortcutted to D (so H_D is a cycle on D). Without loss of generality we can rename nodes so that H_D is the cycle $(v_1, v_2, \ldots, v_{|D|})$. Since $|D|$ is even, we can partition H_D into two perfect matchings:

- $M_1 = \{\{v_1, v_2\}, (v_3, v_4\}, \ldots, \{v_{|D|-1}, v_{|D|}\}\}$, and
- $M_2 = \{\{v_{|D|}, v_1\}, (v_2, v_3\}, \ldots, \{v_{|D|-2}, v_{|D|-1}\}\}$

Clearly M_1 and M_2 form a partition of the edges of H_D into two perfect matchings of D. Thus $c(M_1) + c(M_2) = C(H_D)$, so $\min\{c(M_1), c(M_2)\} \leq \frac{1}{2}c(H_D) \leq \frac{1}{2}c(H^*) = \frac{1}{2} \cdot OPT$. Finally, since M is the minimum cost perfect patching on D we know that $c(M) \leq \min\{c(M_1), c(M_2)\}$, proving the theorem. ■

Unlike Steiner Tree, this simple 3/2-approximation is still the best-known approximation for general metric spaces! There has been a lot of progress recently on both extensions of TSP (e.g., to asymmetric settings were $c(u, v) \neq c(v, u)$) and to special cases (e.g., metrics which arise from unweighted graphs). But for the basic problem we just considered, 3/2 is still the best bound we know of.

References

[BGRS13] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013.

[CC02] Miroslav Chlebík and Janka Chlebíková. Approximation hardness of the steiner tree problem on graphs. In *Algorithm Theory — SWAT 2002*, pages 170–179, 2002.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Carnegie Mellon University, Management Sciences Research Group, February 1976.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[GP68] E. Gilbert and H. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.