600.469 / 600.669 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: Steiner Tree and TSP **Date:** 01/29/15
Scribe: Katie Henry

2.1 Steiner Tree

Definition 2.1.1 *In the Steiner Tree problem the input is a graph $G = (V, E)$ with edge costs $C : E \to \mathbb{R}^+$ and terminals $T \subseteq V$. A subgraph of G is a feasible solution if it spans all of T. The objective is to minimize the cost.*

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 T).

Trivial cases:

- If T is everything, solution is MST.
- If T is two nodes, solution is shortest path.

Definition 2.1.2 *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.*

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.

Lemma 2.1.3 *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 $C(H) \leq C'(H')$.*

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 $C(H) \leq C(\hat{H}) \leq C'(H')$. ■

Lemma 2.1.4 *Let H is a solution to the steiner tree problem on (G, C, T). Then there is a solution H' to the metric steiner tree problem on (V, C', T) with $C'(H') \leq C(H)$.*

Proof: H itself is a solution to the metric steiner tree problem on (V, C', T) with $C'(H') \leq C(H)$, and clearly $C'(H) \leq C(H')$. Thus we can simply let $H' = H$. ■

Theorem 2.1.5 *If there is an α-approximation for metric Steiner tree, then there is an α-approximation for Steiner tree.*

Proof of Theorem 2.1.5: Let C' be the metric completion of (G, C), and 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 then get $\hat{H}$ by replacing each edge $e = \{u, v\} \in H'$ by a $u - v$ path of length $C'(u, v)$. Since this is not necessarily a tree any longer, we then let H be an arbitrary spanning tree of $\hat{H}$.

Claim 2.1.6 *H is feasible.*

Proof: H spans a superset of the nodes spanned by H', and thus spans T. ■

Claim 2.1.7 $C(H) \leq \alpha \cdot OPT$

Proof: We know that $C(H) \leq C(\hat{H})$ because H is a subtree of $\hat{H}$. We know that $C(\hat{H}) \leq C'(H')$ by Lemma 2.1.3. Since $\mathcal{A}$ is assumed to be an α-approximation for the metric Steiner tree problem, we know that $C'(H') \leq \alpha \cdot OPT(V, C', T)$ (i.e. α times the optimum of the metric instance). Finally, we know that $OPT(V, C', T) \leq OPT(G, C, T)$ by Lemma 2.1.4. Putting these all together, we get that $C(H) \leq \alpha \cdot OPT(G, C, T)$ as claimed. ■

Claims 2.1.6 and 2.1.7 together give the theorem. ■

2.1.1 Metric Steiner tree

So now we want to design an approximation algorithm for the Metric Steiner Tree problem – by Theorem 2.1.5 this will give the same approximation for the non-metric case.

Algorithm 1 Metric Steiner Tree

Input: Set of points V metric edge costs C, terminals $T \subseteq V$.
Let F be minimum spanning tree on T.
Return F.

F is the solution returned by the metric Steiner tree algorithm.

Claim 2.1.8 *F is feasible.*

Proof: By definition it spans T. ■

Theorem 2.1.9 $C(F) \leq 2(1 - \frac{1}{|T|})OPT$.

Proof: Let F^* be the optimal solution. Our goal is to find *some* spanning tree $\bar{F}$ of T with $C(\bar{F}) \leq 2(1 - \frac{1}{|T|})C(F^*)$: if we can prove this, then since F is the *minimum* spanning tree of T we will have proved that $C(F) \leq C(\bar{F}) \leq 2(1 - \frac{1}{|T|})C(F^*)$. In order to prove this, we first need a definition.

Definition 2.1.10 *A graph G is* Eulerian *if there is a closed tour of G which traverses every edge exactly once.*

Eulerian graphs are a standard topic in graph theory, in part because it turns out that they have a very simple characterization:

Theorem 2.1.11 *G is Eulerian iff all degrees are even and G is connected.*

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 Eulierian, each edge is in at most one of these paths and hence no edge was double counted. Thus $C(H) \leq C(2F^*) = 2C(F^*) = 2OPT$.

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

A natural question is whether our analysis is tight: is this algorithm actually better than a $2(1-\frac{1}{|T|})$-approximation?

Theorem 2.1.12 *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:

Theorem 2.1.13 (Byrka, Grandoni, Rothvoß, Sanita) *There is a* $(\ln(4)+\epsilon) \approx 1.39$*-approximation to Steiner Tree.*

2.2 TSP: traveling salesman problem

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.

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

Algorithm 2 TSP

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

Theorem 2.2.1 *Algorithm is a* $2(1 - \frac{1}{n})$ *approx.*

Proof of Theorem 2.2.1: 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 $C(H) \leq C(C) = C(2T) = 2C(T) \leq 2C(F) \leq (1 - \frac{1}{|T|}) \cdot OPT$. ■

While this is a simple algorithm, there is a better one due to Christofides [1976]. 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?

Definition 2.2.2 *A **perfect matching** of nodes S is a matching of size $\frac{|S|}{2}$ (i.e. everyone is matched).*

There exists a polytime algorithm to find a min-cost perfect matching.

Theorem 2.2.3 (Christofides 1976) *There is a 3/2-approximation for TSP.*

In order to prove this, we will first note something simple about degrees in graphs: the sum of the degrees is always even.

Lemma 2.2.4 *Let G be an arbitrary graph, and let $d(v)$ denote the degree of v in G. Then $\sum_{v \in V} d(v)$ is even.*

Proof: 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. ■

We can now define Christofides's algorithm.

Algorithm 3 Christofides TSP

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 = shortcutted 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 2.2.5 *Everything is well-defined.*

Proof: We know from Lemma 2.2.4 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 2.2.6 *Christofides is a 3/2-approximation.*

Proof: 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 $C(M) \leq \frac{1}{2}OPT$. To see this, 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. ■