COMPSCI 532: Design and Analysis of Algorithms	October 23, 2015
Lecture 14	
Lecturer: Debmalya Panigrahi	*Scribe: Allen Xiao*

1 Overview

In this lecture, we introduce several network design problems: metric TSP, Steiner tree, and Steiner forest. We give approximation algorithms for the first two based on minimum spanning trees and the metric completion of the graph.

2 Metric Traveling Salesman

The traveling salesman problem is a classical NP-hard problem, and also hard to approximate at all under the assumption $\mathsf{P} \neq \mathsf{NP}$.

Definition 1. *A vertex **tour** of a graph G is a path which visits all vertices and returns to its starting vertex. This is another name for a Hamiltonian cycle on G.*

Traveling salesman problems involve finding a *tour* of undirected graph G with edge weights $d(\cdot)$, which we can think of as a sequence of vertices $\pi = (v_1, v_2, \ldots)$. The cost of a tour is the sum of costs of edges it traverses.

$$d(\pi) = \sum_{i=1}^{|\pi|-1} d(v_i, v_{i+1})$$

Definition 2. *Given a graph $G = (V, E)$ with nonnegative edge weights $d(\cdot)$, the **traveling salesman problem** (TSP) is to find a minimum weight tour π visiting each vertex exactly once.*

A relaxed version of the problem does admit approximation, however.

Definition 3. *Given a graph $G = (V, E)$ with nonnegative edge weights $d(\cdot)$, the **metric traveling salesman problem** (metric TSP) is to find a minimum weight tour π visiting each vertex at least once.*

This version is called *metric* because it is equivalent to solving the regular TSP problem on the *metric completion* of G.

Definition 4. *The **metric completion** of $G = (V, E)$ with nonnegative edge weights $d(\cdot)$ is the complete graph on $G^* = (V, V \times V)$, with edge weights $d^*(\cdot)$. For any pair of vertices $v, w \in V$, we set $d^*(v, w)$ to be the length of the minimum-weight path between them in G under $d(\cdot)$.*

The metric completion has the quality of being a *metric space* of the vertices under the new distance function.

Definition 5. *A **metric space** is a tuple (V, d), where V is a set of points and $d(\cdot)$ is a distance function between pairs of points satsifying:*

Figure 1: An example of the metric completion of a graph. The dashed edges are new edges added to make the graph complete.

1. *Nonnegativity:*
$$\forall v, w \in V, d(v, w) \geq 0$$
2. *Symmetry:*
$$\forall v, w \in V, d(v, w) = d(w, v)$$
3. *Triangle Inequality:*
$$\forall v, w, z \in V, d(v, w) \leq d(v, z) + d(z, w)$$

When $d(\cdot)$ satisfies these properties, we call $d(\cdot)$ a ***metric****.*

Examples of metric spaces include:

- Any Euclidean space using Euclidean distance.
- Vertices of a graph (with nonnegative edge weights) using shortest path distances.

Metric completion uses the second case to gain metric properties on the edge weights. The key property for our approximation algorithms will be the triangle inequality. To summarize, we will list the key differences in metric TSP and TSP.

1. The metric completion G^* is a *complete* graph.
2. The weights $d^*(\cdot)$ in the metric completion are *metric* and the triangle inequality holds.
3. This reduction requires the tour in G^* to visit each vertex *exactly once*.
4. The reduction maps the "exactly once" tour in G^* to one in G which may revisit vertices with equal cost (each metric completion edge maps to a shortest path in G). With this in mind, we will focus on the TSP instance on G^*.

2.1 Approximation using MST

For the rest of this note, we will refer to metric TSP as simply "TSP" for convenience. First, we will use the MST of the metric completion to construct an approximate tour.

Obviously, the optimal value for a tour is at least the value of the minimum spanning tree, since the tour itself spans the graph and is acyclic.

$$\text{OPT(TSP)} \geq \text{OPT(MST)}$$

Figure 2: An example of shortcutting the in-order traversal of an MST starting from a. The red edges are the shortcut edges to avoid revisiting vertex b.

Then, by giving a bound against the value of the MST, we can have a bound against the optimal tour. In this section, we will construct a tour π where:

$$d^*(\pi) \leq 2\text{OPT(MST)}$$

We will construct a tour which has cost no more than twice that of the minimum spanning tree.

Let T be the minimum spanning tree of the metric completion G^*. If we ignore the constraint of visiting every node exactly once, an in-order traversal of T does indeed visit every node, and has cost $\leq$ OPT(MST).

To fix this, we will apply *shortcutting* to construct a tour which does not revisit vertices. When the in-order traversal repeats a vertex, we will instead travel directly to the next unvisited vertex. Since G^* is complete, we know we can choose a direct edge to the next unvisited vertex. Since $d^*(\cdot)$ is metric, the triangle inequality tells us that the cost of taking this edge is at least as short as the original in-order path.

This is a tour π which visits each vertex exactly once, and has cost at most that of the cost of the in-order traversal (which we know is bounded above by 2OPT(MST). Thus, this tour satisfies:

$$d^*(\pi) \leq 2\text{OPT(MST)} \leq 2\text{OPT(TSP)}$$

And this algorithm is a 2-approximation.

2.2 Christofides' algorithm

The primary reason for the 2-approximation was the fact that the in-order traversal used every MST edge twice. We will try improve on this by constructing our tour to traverse MST edges only once, at least before shortcutting. The tool which allows this is the Euler walk.

Theorem 1 (Euler Walk)**.** *If every vertex of a connected graph has even degree, there is a tour visiting every edge exactly once (called the* ***Euler walk****). A graph which has this property is called an* ***Eulerian graph****.*

We omit the proof of this theorem (which is actually if and only if). Intuitively, every vertex has even degree, entering and exiting a vertex during a tour will "use up" its edges in batches of two.

If the MST was Eulerian (by definition, it cannot be), this theorem implies that a tour exists with ratio 1 ($d^*(\pi) \leq$ OPT(MST)). In particular, the Euler walk will cross each edge exactly

once. The in-order traversal from the previous section was more or less constructing an Eulerian graph from the MST, but effectively doubled the number of edges. This time, we will handle the odd-degree vertices of the MST explicitly.

Lemma 2 (Handshaking Lemma)**.** *Every finite graph has an even number of odd-degree vertices.*

Proof. An edge (v, w) shows up in the degree count for exactly two vertices, v and w. Taking the sum of degrees over all vertices:

$$\sum_{v \in V} \deg(v) = 2|E|$$

The total sum is even. Let us separate V into V_e and V_o, where the vertices in V_e have even degree and the vertices in V_o have odd degree.

$$\sum_{v \in V} \deg(v) = \sum_{v \in V_e} \deg(v) + \sum_{v \in V_o} \deg(v) = 2|E|$$

$\deg(v)$ for $v \in V_e$ is by definition even, thus the first sum is even. Subtracting from $2|E|$ (also even) gives an even quantity, therefore:

$$\sum_{v \in V_o} \deg(v) \quad \text{is even.}$$

Each component in the sum is odd, so the total number of components must be even (the parity bit must be multiplied by some factor of 2 before it is 0). □

With this in mind, we will augment the initial MST by perfect matching the odd-degree vertices. This matching adds 1 to the degree of every odd-degree vertex, and therefore the resulting graph has only even-degree vertices. The combined graph is Eulerian and we can use the Euler walk to construct the tour. The final algorithm is a result of Christofides [Chr76].

Algorithm 1 (Christofides 1976)

1: $T \leftarrow$ minimum spanning tree of G^*
2: $M \leftarrow$ min-cost matching of odd-degree vertices of T
3: $W \leftarrow$ an Euler walk on $T \cup M$
4: $\pi \leftarrow$ a shortcutting tour on the order of vertices in W
5: **return** π

The cost of π, since it shortcuts an Euler walk, is bounded above by the cost of the edges in the MST T plus the cost of edges in the matching M.

$$d^*(\pi) \leq d^*(W) = d^*(T) + d^*(M)$$

To analyze the approximation ratio, we analyze separately the cost of T and M separately.

1. By definition, $d^*(T) = \text{OPT(MST)}$. As we argued in the previous section, this gives us:

$$d^*(T) = \text{OPT(MST)} \leq \text{OPT(TSP)}$$

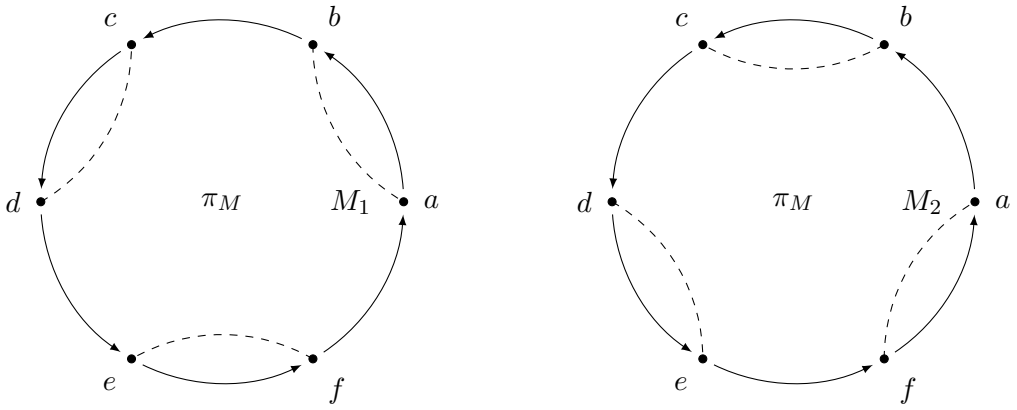

Figure 3: The way we bound the cost of the matching on odd-degree MST vertices. The solid cycle is the optimal tour π_M on the odd-degree MST vertices. We can split π_M into two perfect matchings M_1 and M_2 (dashed lines above). The cheaper between M_1 and M_2 costs at most half $d^*(\pi_M)$ and (by minimality of M) must cost more than M.

2. Let π_M be the optimal TSP tour on just the odd-degree MST vertices (i.e. the vertices of M). As we showed before, M has an even number of vertices and G^* is complete, so we can find two perfect matchings M_1 and M_2 going in either direction of π_M. We can write π_M as a sequence of vertices:

$$\pi_M = (v_1, \ldots, v_{|M|})$$

M_1 and M_2 are defined as matchings of alternating edges of π_M:

$$\begin{aligned} M_1 &= \{(v_1, v_2), (v_3, v_4), \ldots, (v_{|M|-1}, v_{|M|})\} \\ M_2 &= \{(v_1, v_{|M|}), (v_{|M|-1}, v_{|M|-2}) \ldots, (v_3, v_2)\} \end{aligned}$$

By the triangle inequality, π_M requires no edges outside the direct edges $(v_1, v_2), (v_2, v_3)$, and so on. Writing π_M as a set of edges:

$$\begin{aligned} \pi_M &= \{(v_1, v_2), (v_2, v_3), \ldots, (v_{|M|-1}, v_{|M|}), (v_{|M|}, v_1)\} \\ &= M_1 \cup M_2 \end{aligned}$$

We can take the cheaper matching between M_1 and M_2 to say that:

$$d^*(\pi_M) = d^*(M_1 \cup M_2) = d^*(M_1) + d^*(M_2) \geq 2 \min\{d^*(M_1) d^*(M_2)\}$$

Since M is an the optimal perfect matching on the vertices, it must cost no more than either M_1 or M_2.

$$d^*(M) \leq \min\{d^*(M_1), d^*(M_2)\} \leq \frac{1}{2} d^*(\pi_M)$$

Finally, since π_M is tour on only a subset of the vertices of G^*, it must cost no more than the optimal tour on G^*.

$$d^*(M) \leq \frac{1}{2} d^*(\pi_M) \leq \frac{1}{2} \text{OPT(TSP)}$$

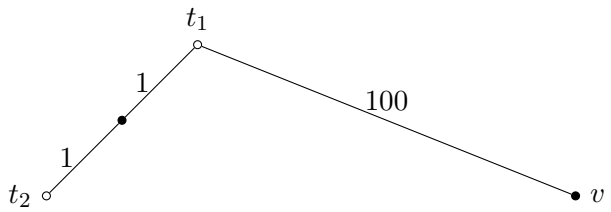

Figure 4: An example where the MST of G will provide a poor approximation for the Steiner tree, because it connects the "unnecessary" vertex v. Additionally, the induced subgraph of terminals is disconnected. The ideal subgraph would leave out v, but keep the rest of the graph connecting t_1 and t_2.

It follows that:

$$\begin{aligned} \text{ALGO} &= d^*(\pi) \leq d^*(W) \\ &= d^*(T) + d^*(M) \\ &\leq \left(1 + \frac{1}{2}\right) \text{OPT(TSP)} \\ &= \frac{3}{2}\text{OPT(TSP)} \end{aligned}$$

And the algorithm is (3/2)-approximate.

3 Steiner Tree

Definition 6. *Given a graph $G = (V, E)$ with edge costs $c(\cdot)$, and a set of terminals $T = (t_1, \ldots, t_k) \subseteq V$, the **Steiner tree** (ST) problem asks to find the minimum cost subgraph H of G connecting T.*

Steiner tree generalizes both s-t shortest path (let s and t be the only terminals) and minimum spanning tree ($T = V$). Steiner trees do not in general span G, however, and using the MST of G can lead to bad approximations. The MST on the induced subgraph of terminals is also insufficient, since it may be disconnected (even empty). However, there is a subgraph whose MST provides a good approximation.

We will use the metric completion of G to construct the correct subgraph of terminals. Let G^* be the metric completion of G, and $G' \subseteq G^*$ be the induced of terminals in G^*. This subgraph satisfies:

- G' is a complete graph on terminals.
- All terminal-to-terminal paths in G' map to paths in G of the same cost.
- Edge costs are metric, and satisfy the triangle inequality.

Using the same argument as our MST 2-approximation for metric TSP, shortcutting the MST on G' (and mapping back to paths in G) provides a 2-approximation for the optimal Steiner tree.

Figure 5: An example graph and its Steiner forest. The resulting subgraph need not be connected.

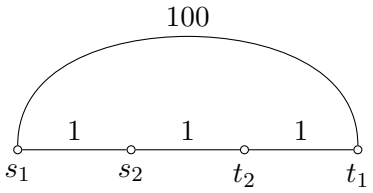

Figure 6: The Steiner forest may also connect certain pairs through other pairs.

Consider an in-order traversal of the optimal Steiner tree on G as Γ. This has cost exactly twice OPT. Now, the shortcutting tour of the MST on G' costs no more than Γ, so using this gives a Steiner tree of cost no more than twice OPT.

The current state of the art for approximating Steiner tree is slightly less than 1.4 [BGRS10], but it is believed that the "correct" approximation lower bound for Steiner tree is 4/3.

4 Steiner Forest

Steiner forest is a generalization of Steiner tree. We will not provide algorithms for Steiner forest this lecture, but instead highlight the new difficulties which distinguish it from Steiner tree.

First, to motivate them, notice that we can restate the Steiner tree problem by selecting a single terminal $r \in T$ as root, then requiring the subgraph to connect all pairs (r, t_i) for $t_i \in T, t_i \neq r$. Instead of a fixed r, we will identify connectivity requirements for arbitrary pairs in T. The resulting subgraph H is a *forest* (a collection of trees).

Definition 7. *Given a graph $G = (V, E)$ with edge costs $c(\cdot)$, and a set of terminal pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$, the **Steiner forest** (SF) problem asks to find the minimum cost subgraph H of G connecting all terminal pairs (s_i, t_i).*

Constant approximation for Steiner forest was open for a long time, until it was solved using primal-dual techniques. We will examine this algorithm and the primal-dual framework in the next lecture.

References

[BGRS10] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved LP-based approximation for Steiner tree. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 583–592. ACM, 2010.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document, 1976.