IOE 691: Approximation Algorithms | Date: 1/9/2017

Lecture Notes: Greedy Algorithms: TSP and k-center

Instructor: Viswanath Nagarajan | Scribe: Karmel Shehadeh

1 Traveling Salesman Problem (TSP)

The input is a set of cities $V = \{1, 2, ..., n\}$, and a distance function $d : V \times V \to R_+$ that satisfies:

- Symmetry: $d(u, v) = d(v, u),\ \forall u, v \in V$
- Triangular inequality: $d(u, w) \leq d(u, v) + d(v, w),\ \forall u, v, w \in V$

Such a distance function is called a *metric*. We also view (V, d) as a complete graph on vertices V with edge-costs given by d. The goal is to find a tour of minimum distance that visits each city exactly once and return to its starting point. We will present two approximation algorithms for TSP, both relying on the minimum spanning tree algorithm.

Lemma 1.1 *For any instance I to the traveling salesman problem, the cost of optimal tour it at least the cost of the minimum spanning tree on I, i.e., $MST(I) \leq TSP(I)$*

Proof: We assume instance I has $n \geq 2$ cities. Start with the optimal TSP tour of cost $TSP(I)$. If your remove one edge from the tour (break the cycle), the result is a spanning tree *ST(I)* with a cost at most $TSP(I)$. Since the minimum spanning tree (MST) is the one with the minimum cost over all spanning trees, it follows that $MST(I) \leq TSP(I)$ ■

Algorithm 1.1. *Double-tree algorithm*

1. Compute the minimum spanning tree M on (V, d).
2. Double all edges of M and call the resulting graph D.
3. Find a walk W that uses each edge of D exactly once (see Theorem 1.1 below).
4. Shortcut W by skipping vertices that are re-visited to get a valid TSP tour T.

Analysis of algorithm 1.1

We first need some preliminaries about *Eulerian* graphs.

Definition 1.1 *An Eulerian graph is a graph with the following properties:*

1. *Connected: there is a path between every pair of vertices.*
2. *All vertices have even degree. Recall that the degree of a vertex is the number of edges incident to it, with self-loops counted twice.*

Theorem 1.1 ([1]) *There is a walk in every Eulerian graph that visits each edge exactly once and this can be computed in polynomial time.*

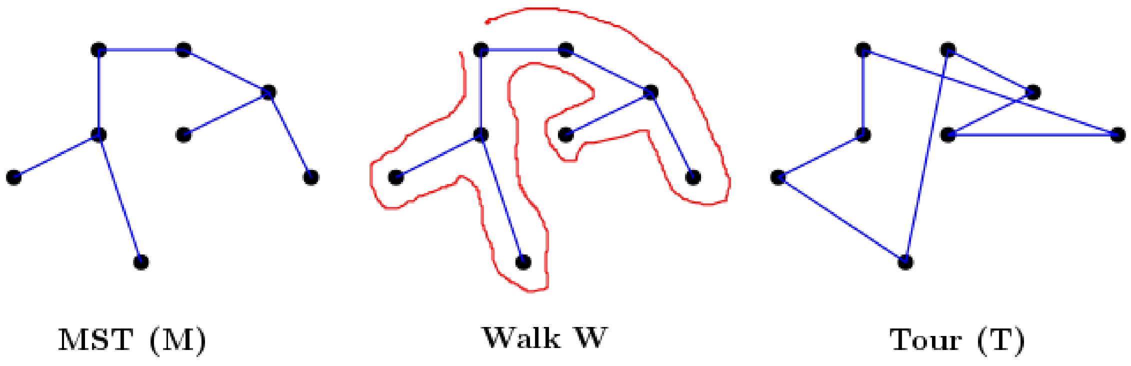

Figure 1: Illustrative instance of the resulting TSP tour from the double tree algorithm

Theorem 1.2 *The double-tree algorithm for TSP is a 2-approximation algorithm.*

Proof: Let OPT be the cost of the optimal TSP tour. By Lemma 1.1, the cost of the minimum spanning tree M is at most OPT. We then double each edge (replace it with two copies) of M and the cost of the resulting graph D is at most $2OPT$. Also D is *Eulerian* by construction. By Theorem 1.1, a walk W of cost at most $2OPT$ can be constructed via the *Eulerian* traversal of the edges in D. Let W be the sequence $i_0, i_1, ..., i_k$ of cities where there may be repetitions. To get a tour T, we removing all but the first occurrence of each city in this sequence. This tour T contains each city exactly once (starts at i_0 and returns to i_0). We now show that the cost of T is at most that of W. Consider two consecutive cities in T: i_ℓ and i_m (we omitted $i_{\ell+1}, ..., i_{m-1}$ since these cities were already visited earlier in T). It then follows from the triangle inequality (and induction) that the distance d_{i_ℓ, i_m} is upper bounded by the total distance of the edges $(i_\ell, i_{\ell+1}), ..., (i_{m-1}, i_m)$. Adding up over all edges in T, the cost of T is at most the cost of W which is at most $2OPT$. ■

We now discuss a better approximation algorithm by building on the above ideas. First we need the following problem definition.

Matching: The input is a graph $G = (U, E)$ with even number of vertices U and distance function $d : U \times U \to R_+$. The goal is to find edges $K \subseteq E$ such that each vertex has exactly one end-point in K with minimum cost $\sum_{e \in K} d(e)$.

Theorem 1.3 ([2]) *A minimum cost matching can be found in polynomial time.*

Algorithm 1.2 Christofides' algorithm for TSP.

1. Compute the minimum spanning tree M on (V, d).
2. Compute the minimum cost matching K on odd degree vertices of M.
3. Add the edges of K to M to obtain an Eulerian graph D'.
4. Find a walk W' that uses each edge of D' exactly once.
5. Shortcut W' by skipping vertices that are re-visited to get a valid TSP tour T'.

Observation 1.1 *The number of odd degree vertices in M is even.*

Proof: Let $V_{even} \subset V$ and $V_{odd} \subset V$ be the subsets of even and odd degree vertcies in M

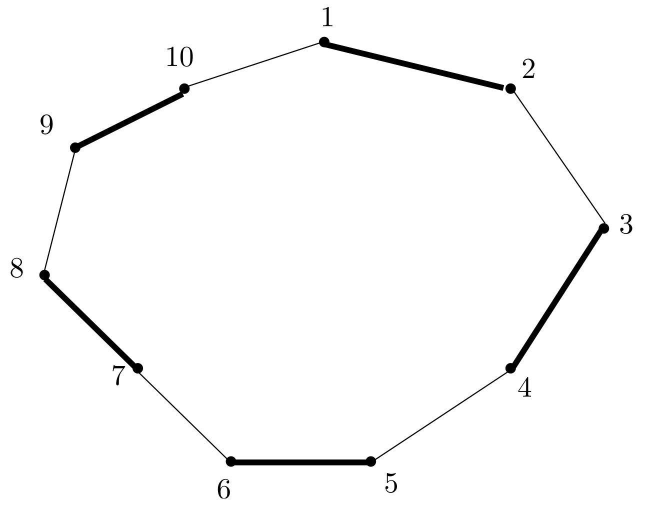

Figure 2: Illustrative instance the two matchings M_1 (thick edges), and M_2 (thin edges)

respectively.

$$2|E| = \sum_{\vartheta \in V} deg(\vartheta) = \sum_{\vartheta \in V_{odd}} deg(\vartheta) + \underbrace{\sum_{\vartheta \in V_{even}} deg(\vartheta)}_{\text{even}} = even$$

Hence $|V_{odd}|$ is even. ■

Analysis of algorithm 1.2. The only additional fact we need is:

Observation 1.2 *The minimum cost matching on any set U (even number of vertices) is at most $\frac{1}{2}OPT$, where OPT is the cost of the optimal TSP tour.*

Proof: Consider the optimal TSP tour O and shortcut over all vertices *not* in U to obtain cycle O' containing vertices U. By triangle inequality, the cost of O' is at most that of O which is OPT. We define two candidate matchings on U using O'. By renumbering vertices let O' be the sequence $1, 2, \cdots, |U|, 1$ of vertices. Let M_1 be the matching that pairs vertices as $(1,2), (3,4) \cdots (|U|-1, |U|)$ and M_2 be $(|U|, 1), (2,3) \cdots (|U|-2, |U|-1)$. See Figure 2 for an example. Then $\text{Cost}(M_1) + \text{Cost}(M_2) = \text{Cost}(O') \leq OPT$. So $\min(\text{Cost}(M_1), \text{Cost}(M_2)) \leq OPT/2$. ■

Theorem 1.4 *Christofides' algorithm for TSP is a 3/2-approximation algorithm.*

Proof: We know that the $Cost(MST) \leq OPT$, and that the min-cost matching has $\text{Cost}(K) \leq OPT/2$. So the cost of D' (and hence T') is at most $\frac{3}{2}OPT$. ■

2 The k-center problem

The input is a set **V** of vertices, distance function $d{:}V \times V \to R_+$ (symmetric and satisfies triangle inequality) and a bound k. The distances model similarity between these vertices (i.e., closer vertices are similar to each other). One wants to find k-clusters which group together vertices that are most similar into the same clusters. Formally, The goal is to choose a set $S \subseteq V$ of k "cluster

Algorithm 1 A greedy 2-approximation algorithm for the k-center problem

```
1: set $s_1 \in V$ as the first center (arbitrary)
2: $S \leftarrow \{s_1\}$
3: for $i = 1, 2, ..., k$
.      $s_i \leftarrow \arg\max_{u \in V} \left( \min_{s \in \{s_1, ..., s_{i-1}\}} d(s, u) \right)$
.      $S \leftarrow S \cup \{s_i\}$
```

centers" so as to minimize the maximum distance of any vertex to its nearest center, i.e.

$$\min_{S \subseteq V : |S| = k} \quad \max_{\vartheta \in V} \left(\min_{s \in S} d(\vartheta, s) \right).$$

A natural greedy algorithm is to repeatedly pick as a new center the vertex that is as far as possible from the existing centers.

Theorem 2.1 *Algorithm 2.1 is a 2-approximation algorithm for the k-center problem.*

Proof: Let $O^* = \{o_1, ..., o_k\}$ be the optimal set of k-centers with optimal cost R^*. Let $X_1, X_2, ..., X_k$ be clusters, where, X_i are all vertices that are closet to $o_i \in O^*$. Note that $d(u, o_i) \leq R^*$ for all $u \in X_i$ and $i \in [k]$. Observe that if $|S \cap X_i| \neq \phi, \forall i = 1, ..., k$, then S is a 2-approximation. This is because each $u \in X_i$ can connect to the center $s' \in S \cap X_i$ at distance $d(u, s') \leq d(u, o_i) + d(o_i, s') \leq 2R^*$ by triangle inequality and definition of cluster X_i; so $\min_{s \in S} d(u, s) \leq 2R^*$ for all $u \in V$.

Now suppose in some iteration j, we pick center s_j in some cluster X in which an earlier center s_r was already picked ($r < j$). Since $s_j, s_r \in X$, the distance between s_j and s_r is $d(s_j, s_r) \leq 2R^*$ (by triangular inequality) and thus $min_{j'<j} d(s_j, s_{j'}) \leq 2R^*$. Now, by the greedy choice of s_j, for all $u \in V$:

$$\min_{j'<j} d(u, s_{j'}) \leq \min_{j'<j} d(s_j, s_{j'}) \leq 2R^*.$$

This clearly implies $\min\limits_{s \in S} d(u, s) \leq \min\limits_{j'<j} d(u, s_{j'}) \leq 2R^*$ for all $u \in V$. ∎

References

[1] J. A. Bondy and U. S. R. Murty, *Graph theory with applications*, vol. 290. 1976.

[2] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices," *Journal of Research of the National Bureau of Standards Section B*, p. 125–130, 1965.