Lecture notes for Apr 3, 2023
Metric TSP and edge-cuts

Chun-Hung Liu

April 3, 2023

1 Metric TSP

Due to the hardness for approximating TSP, we consider a special case of TSP.

==================================

Metric TSP
Input: A positive integer n and a function $w : E(K_n) \to \mathbb{R}_{\geq 0}$ that satisfies the triangle inequality (i.e. for any $x, y, z \in V(K_n)$, $w(x, y) + w(y, z) \geq w(x, z)$).
Output: A cycle C of K_n containing every vertex exactly once such that $\sum_{e \in E(C)} w(e)$ is minimum.
==================================

We will give constant factor approximation for Metric TSP.

Lemma 1 *Let (K_n, w) be a weighted complete graph such that w satisfies the triangle inequality. Let H be a loopless graph with $V(H) = V(K_n)$. Let w_H be the weight function such that for every $uv \in E(H)$, $w_H(uv) = w(uv)$. If R is an Eulerian circuit of H, then we can construct a Hamiltonian cycle C of K_n with $w(C) \leq w_H(R)$ in time $O(|E(H)|)$.*

Proof. Since K_n is simple, we can describe R by listing the order vertices that it visits. Let $R = v_1 v_2 v_3 ....$ Note that R is a walk that visits all vertices of K_n since H is connected and $V(H) = V(K_n)$. Let $R_1 = R$. For $i \geq 1$, if $v_{i+1} \in \{v_j : j \in [i]\}$, then define R_{i+1} to be the sequence obtained from R_i

by removing v_{i+1}; otherwise, define $R_{i+1} = R_i$. If $R_{i+1} = R_i$, then R_{i+1} is a walk visiting all vertices with $w_H(R_{i+1}) = w_H(R_i)$; otherwise, since K_n is a complete graph, R_{i+1} describes the walk obtained from R_i by replacing the path $v_i v_{i+1} v_{i+2}$ by the edge $v_i v_{i+2}$, so R_{i+1} is also a walk visiting all vertices of K_n (as v_{i+1} was visited before v_i) and $w_H(R_{i+1}) = w_H(R_i) - w(v_i v_{i+1}) - w(v_{i+1} v_{i+2}) + w(v_i v_{i+2}) \leq w_H(R_i)$ (by the triangle inequality). Hence $R_{|R|+1}$ is a closed walk that visits all vertices of K_n with no repeated vertices (so it is a Hamiltonian cycle) with weight at most $w(R_1) = w(R)$. Note that $R_{|R|+1}$ can be constructed in time $O(|R|) = O(|E(H)|)$. ■

==============================

A 2-approximation algorithm for Metric TSP
Input: A positive integer n and a function $w : E(K_n) \to \mathbb{R}_{\geq 0}$ that satisfies the triangle inequality (i.e. for any $x, y, z \in V(K_n)$, $w(x, y) + w(y, z) \geq w(x, z)$).
Output: A Hamiltonian cycle C of K_n with $w(C) \leq 2\text{OPT}$, where $\text{OPT} = \min_Z w(Z)$ over all Hamiltonian cycles Z of (K_n, w).
Procedure:

Step 1: Find a minimum weighted spanning tree T of (K_n, w).

Step 2: Double each edge of T to obtain the graph T' and find an Eulerian tour R of T'.

Step 3: Since R is an Eulerian circuit of a connected graph T' with $V(T') = V(K_n)$, we can replace R by a Hamiltonian cycle C of K_n by Lemma 1. Output C.

==============================

Theorem 2 *The above algorithm outputs a Hamiltonian cycle C of K_n with $w(C) \leq 2\text{OPT}$ in time $O(n^2)$, where $\text{OPT} = \min_Z w(Z)$ over all Hamiltonian cycle Z of (K_n, w).*

Proof. Let C^* be a Hamiltonian cycle of K_n with $w(C^*) = \text{OPT}$. Since C^* contains a Hamiltonian path of K_n, which is a spanning tree of K_n, we know $w(C^*) \geq w(T)$. Note that $w(R) = w(T') = 2w(T)$. By Lemma 1, $w(C) \leq w(R) \leq 2w(T) \leq 2w(C^*) = 2\text{OPT}$. This shows the correctness.

Step 1 takes time $O(n^2)$ (by using Prim's algorithm). Step 2 takes time $O(n)$. Step 3 takes time $O(|E(T')|) = O(n)$. So the algorithm takes time $O(n^2)$. ■

We can improve the approximation factor by using a slower polynomial time algorithm.

================================

Christofide's algorithm for Metric TSP
Input: A positive integer n and a function $w : E(K_n) \to \mathbb{R}_{\geq 0}$ that satisfies the triangle inequality (i.e. for any $x, y, z \in V(K_n)$, $w(x, y) + w(y, z) \geq w(x, z)$).
Output: A Hamiltonian cycle C of K_n with $w(C) \leq \frac{3}{2}\text{OPT}$, where $\text{OPT} = \min_Z w(Z)$ over all Hamiltonian cycles Z of (K_n, w).
Procedure:

Step 1: Find a minimum weighted spanning tree T of (K_n, w).

Step 2: Let X be the set of odd degree vertices in T. Find a minimum weighted X-join J in (K_n, w).

Step 3: Note that the graph $T + J$ is Eulerian. Find an Eulerian circuit R of $T + J$.

Step 4: Replace R by a Hamiltonian cycle C of K_n by Lemma 1. Output C.

================================

Theorem 3 *Christofide's algorithm outputs a Hamiltonian cycle C of K_n with $w(C) \leq \frac{3}{2}\text{OPT}$ in time $O(n^3)$.*

Proof. Let C^* be a Hamiltonian cycle of K_n with $w(C^*) = \text{OPT}$. Since C^* contains a Hamiltonian path of K_n, which is a spanning tree of K_n, we know $w(C^*) \geq w(T)$. Since C^* is a Hamiltonian cycle, it passes through all vertices in X. Let $x_1, x_2, ..., x_{|X|}$ be the vertices in X, ordered by the order passed though by C^*. For every $i \in [|X|]$, let P_i be the subpath of C^* between x_i and x_{i+1} internally disjoint from X, where $x_{|X|+1} = x_1$. Note that X is the set of odd degree vertices, so $|X|$ is even. Hence $\bigcup_{i=1}^{|X|/2} E(P_{2i-1})$ and $\bigcup_{i=1}^{|X|/2} E(P_{2i})$ are two disjoint X-joins. That is, $E(C^*)$ is a union of two disjoint X-joins, so $w(C^*) \geq 2w(J)$. Hence $w(C) \leq w(R) = w(T+J) \leq \frac{3}{2}w(C^*)$. This shows the correctness.

Step 1 takes time $O(n^2)$. Step 2 takes time $O(n^3)$. Step 3 takes time $O(n)$. Step 4 takes time $O(n)$. So it takes $O(n^3)$ in total. ■

This $\frac{3}{2}$ approximation ratio was established in the 1970s. Only until recently (2021), Karlin, Klein and Gharan proved that there is a randomized algorithm that outputs a Hamiltonian cycle with expected weight at most $(\frac{3}{2} - \epsilon)$OPT for some constant $\epsilon > 10^{-36}$.

2 Preparation for Gomory-Hu tree

Recall that we have a polynomial time algorithm to find a maximum flow and a minimum cut of a network. In particular, given a digraph D and two vertices x, y, we can find an edge-cut of D of minimum size that separates x and y by considering the network $(D, x, y, 1)$. Given a graph G, by replacing each edge of G by a pair of directed edges with different directions, the above algorithm finds, given two vertices x, y, an edge-cut of G of minimum size that separates x and y. So we can determine the edge-connectivity of G by checking the minimum size of an edge-cut for $\binom{|V(G)|}{2}$ pairs of distinct vertices. That is, we can determine the edge-connectivity of a n-vertex graph by applying the algorithm for finding a maximum flow and a minimum cut $\binom{|V(G)|}{2}$ times.

Gomory-Hu tree provides a more efficient way to find the edge-connectivity and more structural information for the graph. We will consider a more general setting for weighted graphs.

2.1 Edge-cuts

Before defining Gomory-Hu trees, we define some terminologies that will be convenient later.

An *edge-cut* of a positive weighted graph (G, w) is an ordered partition $[A, B]$ of $V(G)$ into (possibly empty) sets. And the *weight* of $[A, B]$ is defined to be $\sum_{e \in \delta(A)} w(e)$, denoted by $w(A, B)$. For two vertices u and v of G,

- we say that an edge-cut of (G, w) *separating* u and v if u and v are in different parts of the edge-cut, and

- a *minimum weighted* (u, v)*-edge-cut* is an edge-cut $[A, B]$ of G with $u \in A$ and $v \in B$ such that the weight of $[A, B]$ is the minimum among all edge-cuts of (G, w) separating u and v.

Theorem 4 *Let (G, w) be a weighted graph with positive w. Let u, v be distinct vertices of G. Then a minimum weighted (u, v)-edge-cut can be found in time $O(|V(G)|^2\sqrt{|E(G)|})$.*

Proof. Create a digraph D obtained from G by replacing each edge of G by two directed edges with different directions. For every $(x, y) \in E(D)$, define $c(x, y) = w(xy)$. Then use Edmonds-Karp algorithm to find a minimum cut S for the network (D, u, v, c) in time $O(|V(G)||E(G)|^2)$. (In fact, it can be done in time $O(|V(G)|^2\sqrt{|E(G)|})$ by a more complicated algorithm.) Let $A = S$ and $B = V(G) - S$. Then $[A, B]$ is a minimum weighted (u, v)-edge-cut. ■

2.2 Tree-cut decomposition

A *tree-cut decomposition* of (G, w) is a pair $(T, \mathcal{X})$, where T is a tree and $\mathcal{X}$ is a partition $\{X_t : t \in V(T)\}$ of $V(G)$ into (possibly empty) sets indexed by $V(T)$. For every edge xy of T,

- we know that $T - xy$ contains two components T_x and T_y of T, where T_x contains x and T_y contains y, so $[\bigcup_{t\in V(T_x)} X_t, \bigcup_{t\in V(T_y)} X_t]$ is an edge-cut of (G, w), and we call this edge-cut the *edge-cut given by* (x, y) *(with respect to* $(T, \mathcal{X})$*)*,

- the *adhesion of* xy *(with respect to* $(T, \mathcal{X})$*)* is defined to be the weight of the edge-cut given by (x, y),

2.3 Gomory-Hu tree

A *Gomory-Hu tree* of a positive weighted graph (G, w) is a tree-cut decomposition $(T, \mathcal{X})$ of G such that

- $|X_t| = 1$ for every $t \in V(T)$, and

- for any distinct $x, y \in V(G)$, the minimum weight of an edge-cut of (G, w) separating x and y equals the minimum of the adhesion of e over all edges e in the unique path in T between x and y.

(Note that it implies that if the edge e gives the minimum adhesion in the path between x and y, then the edge-cut given by e is an edge-cut of (G, w) separating x and y.)

Therefore, if a Gomory-Hu tree $(T, \mathcal{X})$ of a positive weighted graph (G, w) is given, we can find the edge-connectivity in linear time by simply finding the minimum adhesion of the edges of T, which only takes time $O(|V(G)|)$; and for any distinct vertices x, y of G, we can find an edge-cut separating x and y with minimum weight by simply finding the edge in T in the path between x and y giving the minimum adhesion, which only takes time $O(|V(G)|)$.