11 Metric TSP

11.1 Inapproximability of TSP • 11.2 Double-Tree Algorithm • 11.3 Christofides' Algorithm

Travelling Salesman Problem (TSP): Given an undirected complete graph G, with weights $w : E(G) \to \mathbb{R}_{\geq 0}$, find a Hamilton cycle of minimum weight.

11.1 Inapproximability of TSP

Theorem 11.1. *There is no k-approximation algorithm for TSP for any $k \in \mathbb{R}$, unless $\mathcal{P} = \mathcal{NP}$. The same is true if we replace k by any function $f(n)$ of $n = |V(G)|$ that is computable in polynomial time.*

Proof. We show that, given a k-approximation algorithm for TSP, we get a polynomial algorithm for the decision problem Hamilton Cycle. Since we proved that problem to be $\mathcal{NP}$-complete, this would mean $\mathcal{P} = \mathcal{NP}$.
Let G be an instance of Hamilton Cycle, i.e. a graph for which we want to determine if it contains a Hamilton cycle. We define a new complete graph G' with $V(G') = V(G)$, and weights w that are 1 for edges that are also in G, and kn otherwise.
We claim that G has a Hamilton cycle if and only if the k-approximation algorithm applied to G' finds a Hamilton cycle of weight $\leq kn$. If G has no Hamilton cycle, then every Hamilton cycle in G' has weight $> kn$, since it must use one of the edges of weight kn. If G does have a Hamilton cycle, then it gives a Hamilton cycle T^* of weight n in G', which must be minimum. Then the k-approximation algorithm will return a Hamilton cycle of weight $\leq k \cdot w(T^*) = kn$.
The second statement of the theorem follows by just replacing k by $f(n)$ in the proof above. The only complication is that the function should be computable in polynomial time, since otherwise the resulting algorithm for Hamilton Cycle would not be polynomial. □

Note that a function that is computable in polynomial time need not be a polynomial; for instance, $f(n) = 2^n$ can be computed in n multiplication steps. We won't go into what functions have this property or not, the point here is that an approximation factor like the $\log(n)$ that we saw for Set Cover is not possible for TSP (unless, sigh, $\mathcal{P} = \mathcal{NP}$). This follows since $\log(n) \leq n$ and n is computible in polynomial time (by the ingenious algorithm that, given n, returns n).

11.2 Double-Tree Algorithm

Since approximating the general TSP is hopeless, it makes sense to consider restricted versions, like the metric one that we saw in Problem Set 9. Most applications satisfy this condition.
Note that this is not the same as the *Euclidean* TSP, where the vertices are points in $\mathbb{R}^2$ and the weights their Euclidean distances.

METRIC TSP: Let G be a complete undirected graph G with a weights $d : E(G) \to \mathbb{R}_{\geq 0}$ that satisfy the triangle inequality

$$d(uw) \leq d(uv) + d(vw)$$

for all $u, v, w \in V(G)$.
Find a minimum-weight Hamilton cycle in G.

Note that the inapproximability proof above does not apply here, because the graph G' constructed there does not satisfy the triangle inequality: For most H, there will be a triangle in G' with one edge of weight kn and two edges of weight 1, and $kn > 1 + 1$.

The following lemma will be used in the proofs for our next two algorithms. Recall that a *tour* is a closed walk, i.e. a sequence of consecutive edges that starts and ends at the same vertex, and is allowed to repeat vertices and edges.

Lemma 11.2 (Shortcutting)**.** *Let G be a complete graph with weights d satisfying the triangle inequality.*
Given a tour E that visits all vertices in G, one can construct a Hamilton cycle C with $d(C) \leq d(E)$.

Proof. Write $V(G) = \{v_1, v_2, \ldots, v_n\}$, with the vertices in the order in which they appear in in E, starting from an arbitrary vertex and in an arbitrary direction. Also write $v_{n+1} = v_1$. Then we take the Hamilton cycle $C = v_1 v_2 \cdots v_n v_{n+1}$.
For each i, let P_i be the walk along E from v_i to v_{i+1}. Then by repeated applications of the triangle inequality and completeness of the graph we have

$$d(v_i v_{i+1}) \leq d(P_i).$$

Therefore

$$d(C) = \sum_{i=1}^{n} d(v_i v_{i+1}) \leq \sum_{i=1}^{n} d(P_i) = d(E).$$

□

Double-Tree Algorithm

1. Find an MST T in G with respect to d;
2. Double the edges of T to obtain a graph D;
3. Find an Euler tour E in D;
4. Shortcut E to get a Hamilton cycle C and return C.

Theorem 11.3. *The Double-Tree algorithm is a 2-approximation algorithm for Metric TSP.*

Proof. If C^* is a minimum Hamilton cycle we have

$$d(C) \leq d(E) = 2d(T) \leq 2d(C^*).$$

The first inequality comes from Lemma 11.2, and the equality is clear from the construction. The last inequality follows from the fact that a Hamilton cycle minus any edge e is a spanning tree, so $d(T) \leq d(C^*) - d(e) \leq d(C^*)$. □

11.3 Christofides' Algorithm

Christofides' Algorithm for Metric TSP

1. Find an MST T in G with respect to d;
2. Let H be the complete graph on the vertices that have odd degree in T;
3. Find a minimum weight perfect matching M in H, with weights d;
4. Find an Euler tour E in the graph $(V(G), E(T) \cup M)$;
5. Shortcut E to get a Hamilton cycle C and return C.

To see that step 3 is well-defined, recall from basic graph theory the fact that the number of odd-degree vertices is always even (proof: the sum of the degrees is twice the number of edges). So $|H|$ is even and has a perfect matching.
Step 4 is well-defined because adding the matching edges exactly turns the odd-degree vertices into even-degree vertices, so that an Euler tour exists.

Theorem 11.4. *Christofides' algorithm is a 3/2-approximation algorithm for* Metric TSP.

Proof. If C^* is a minimum Hamilton cycle we have

$$d(C) \leq d(E) = d(T) + d(M) \leq d(C^*) + \frac{1}{2}d(C^*) = \frac{3}{2}d(C^*).$$

Everything is similar to in the previous proof, except for the inequality $d(M) \leq d(C^*)/2$.
We'll prove that $d(C^*) \geq 2d(M)$. Write $H = \{h_1, \ldots, h_m\}$ in the order of appearance in C^*, and define the Hamilton cycle on H

$$C_H = h_1 h_2 \cdots h_m h_1.$$

Then $d(C^*) \geq d(C_H)$ by the triangle inequality.
Since $|H|$ is even, $|E(C_H)|$ is even, so we have $E(C_H) = M_1 \cup M_2$, a disjoint union of two perfect matchings of H (simply put edges of C_H alternately in M_1 and M_2). Since M is a minimum weight perfect matching on H, we have $d(M) \leq d(M_i)$. Therefore

$$d(C^*) \geq d(C_H) = d(M_1) + d(M_2) \geq 2d(M).$$

□

A few research results

Christofides' algorithm has the best-known worst-case approximation factor for METRIC TSP. The best lower bound is $220/219 = 1.004\cdots$, i.e. no k-approximation is possible with $k \leq 220/219$ (yes, unless $\mathcal{P} = \mathcal{NP}$). On the other hand, for the EUCLIDEAN TSP (which is also $\mathcal{NP}$-hard), there is an $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ (by Arora in 1998).

Instead of looking at worst-case bounds, one can also empirically test the typical behavior. On a set of random examples (see the book by Cook, Cunningham, Pulleyblank, Schrijver, Ch.7), Christofides has an average approximation factor $l = 1.09$. For comparison, the greedy approach (repeatedly picking the closest unvisited vertex) has $l = 1.26$ on the same examples, and the Double-Tree algorithm does worse (it wasn't even included in this test). So Christofides' algorithm does pretty well in practice, but it has the drawback of being pretty slow, since it has to find a minimum weight perfect matching, which we know is a pretty complicated algorithm. It is also terrible for non-metric graphs.

In the last lecture, we'll see some of the approaches that do not have worst-case bounds, but are faster, have l as low as 1.01, and do all right on non-metric graphs.