Lecture 3: The Traveling Salesman Problem

1 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a challenge to the salesman who wants to visit every location exactly once and return home, as quickly as possible. Each location can be reached from every other location, and for each pair of locations, there is metric that defines the time between them.

Input: An complete undirected weighted graph $G = (V, E)$ with $|V| \geq 3$. Weights on edge $e = \{v_i, v_j\}$ are written as $w(e) = w_{i,j} = w_{j,i}$. All weights are positive.

Output: Consider all cycles C' where all vertices are visited exactly once.

$$C' = v_{\pi(0)}, v_{\pi(0)}, \ldots, v_{\pi(n-2)}, v_{\pi(n-1)}$$

$$i \neq j : v_{\pi(i)} \neq v_{\pi(j)}$$

The weight of these cycles $w(C')$ is the sum of the weight of the edges used to walk the cycle.

$$w(C') = \sum_{i=0}^{n-1} w_{\pi(i),\pi(i+1)} + w_{\pi(n-1),\pi(0)}$$

The output is the cycle C together with its weight $w(C)$ where C is the cycle of minimum weight, OPT. Note that this cycle is not necessarily unique.

$$\forall C' : OPT = w(C) \leq w(C')$$

The output can also be written as a tour T, or sequence of edges:

$$e_i = \{v_{\pi(i)}, v_{\pi((i+1)\%n)}\}$$

$$T = e_0, e_1, \ldots, e_{n-2}, e_{n-1}$$

$$w(T) = \sum_{i=0}^{n-1} w(e_i)$$

Complexity: This problem is NP-hard.

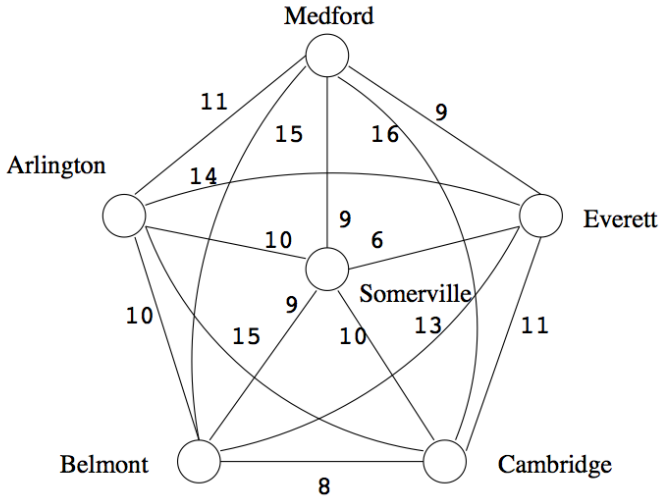

Figure 1: Example input to the TSP

2 Metric TSP

In general, we can say nothing about the weights of edges in a given TSP graph except that they are positive. Metric TSP is an special case of the TSP that satisfies the triangle inequality. The triangle inequality ensures that a direct path between two vertices is at least as short as any indirect path. This is called the triangle inequality because no 1 side of a triangle can be longer than the sum of the other 2.

Definition 2.0.1 *A complete graph satisfies the* **Triangle Inequality** *if* $\forall i, j, k : w_{i,k} \leq w_{i,j} + w_{j,k}$.

The example given in figure 1 satisfies the triangle inequality and therefore this is a case of Metric TSP. On the other hand, there are commonly occurring examples on non-metric TSP. If the weights represent something like the cost of airline tickets, for example, there is no reason to expect the input would satisfy the triangle inequality; it is often the case that taking an indirect route is less expensive than a direct route.

Complexity: Also NP-hard.

However, we can get close to the solution in polynomial time using approximiation algorithms. In the following 2 sections we'll see approximation algorithms that come within $2 * OPT$ and $1.5 * OPT$.

3 Approximation Algorithm 1

Goal: Given a metric TSP graph G, give a cycle C with $w(C) < 2 * OPT$ in polynomial time, where OPT is the weight of the solution T to the metric TSP. This cycle must be a feasible solution to the metric TSP (i.e. it must visit each vertex of G exactly once).

3.1 Step 1: Minimum Spanning Tree

Our first step will be to create a minimum spanning tree (MST) for the graph G. This can be accomplished using algorithms such as *Kruskal's algorithm* or *Prim's algorithm*. Figure 2 shows the MST for the example input given in figure 1.

Definition 3.1.1 *A* **Spanning Tree** *S of a graph H is a connected subgraph of H with no cycles and containing all vertices of H. The* **Minimum Spanning Tree** *is the tree S such that $w(S) \leq w(S')$ for all spanning trees S', where the weight of a spanning tree is the sum of the weights on its edges.*

Claim 3.1.2 *The weight of the MST M is less than OPT, the weight of the TSP solution T.*

Proof:

- Take T and remove an edge e. T is now a spanning tree.
- Because M is the MST, $w(M) \leq w(T-e) = w(T)-w(e) = OPT-w(e)$

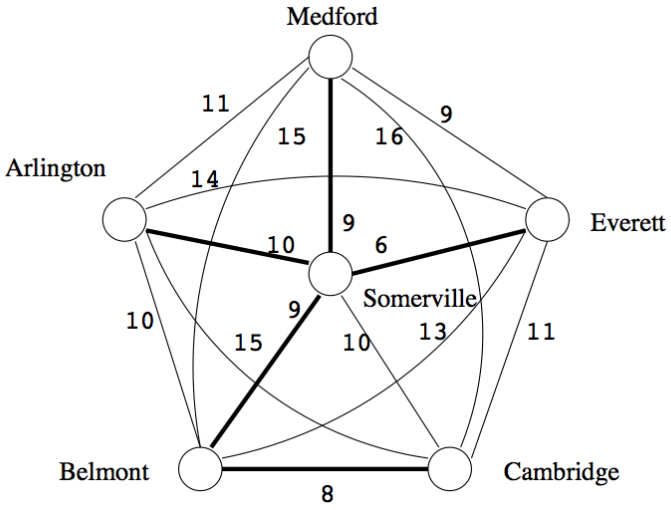

Figure 2: Minimum spanning tree

- $\forall e : w(e) > 0$
- Therefore, $w(M) < OPT$.

3.2 Step 2: Creating a Cycle

Now that we have our MST M, we can create a cycle W from it. To do this, we perform preorder tree walk on the MST, and list vertices in the order in which they are visited. Graphically, this means outlining the tree. An example cycle is given in figure 3. Using the first letter of the city names, this cycle is S,M,S,E,S,B,C,B,S,A.

Because each edge is used exactly twice during a depth first search on a tree (once descending, once ascending), $w(W) = 2 * w(M) < 2 * OPT$. However, we cannot call this our solution, because we are visiting vertices multiple times.

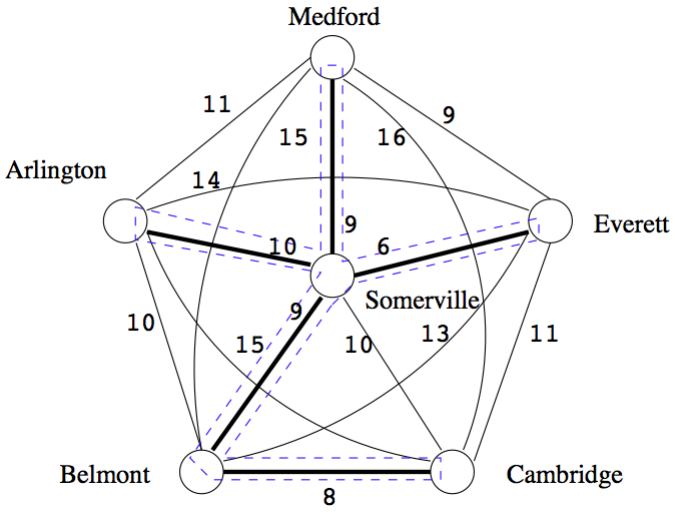

Figure 3: Cycle from MST shown in dashed blue

3.3 Step 3: Removing Redundant Visits

In order to create a plausible solution for the TSP, we must visit vertices exactly once. Since we used an MST, we know that each vertex is visited at least once, so we need only remove duplicates in such a way that does not increase the weight.

Using the triangle inequality, we can create our solution cycle C by using vertices only the first time that they are seen. This is possible because the triangle inequality allows us to remove any intermediate vertices in a path and not increase the path weight.

The solution to this approximation algorithm is shown in figure 4. The cycle is S,M,E,B,C,A with weight 64.

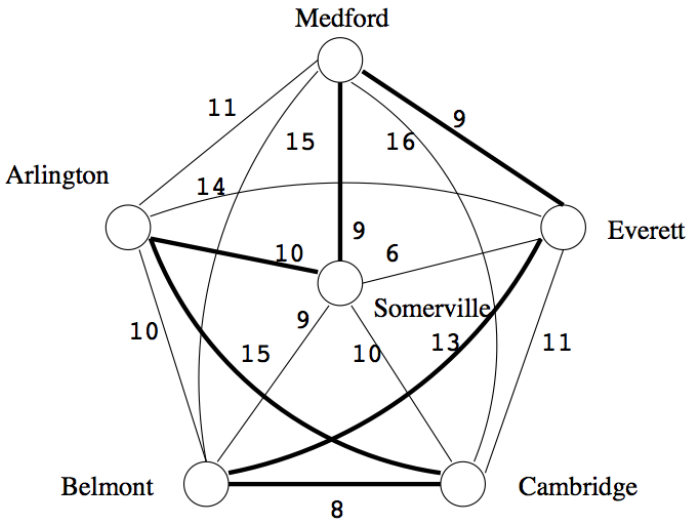

Figure 4: Solution with weight less than $2 * OPT$

4 Approximation Algorithm 2: Christofides' Algorithm

Goal: Given a metric TSP graph G, give a cycle C with $w(C) \leq 1.5 * OPT$ in polynomial time, where OPT is the weight of the solution T to the metric TSP. This cycle must be a feasible solution to the metric TSP (i.e. it must visit each vertex of G exactly once).

This algorithm is credited to N. Christofides [2]. In order to develop his algorithm, we need to establish 3 facts:

4.1 Fact 1: Vertices of Odd Degree

Any graph has an even number of vertices with odd degree.

For the proof, see any basic graph theory or algorithms text. The proof relies on the fact that every edge contributes exactly 2 to the sum of degrees. Therefore the sum of degrees must be even, and so the number of vertices

with odd degree must be even.

4.2 Fact 2: Eulerian Tour

Any connected graph whose vertices are all of even degree has an Eulerian Tour.

Definition 4.2.1 *A* **Eulerian Tour** *is a cyclic tour that uses every edge exactly once.*

For the proof, see any intermediate algorithms text such as the class text. The proof relies on the intuition that with all vertices having even degrees, when entering a vertex, it is always possible to exit it, unless it is the origin. If any edges are unused, repeat the process and merge the two cyclic tours.

4.3 Fact 3: Minimum Matchings in Polynomial Time

Given a complete weighted graph with an even number of vertices, a minimal weight perfect matching can be found in polynomial time.

For the proof see *Paths, Trees, and Flowers* by Jack Edmonds [3]. The proof goes beyond the scope of this class.

The running time of even bad implementations of Edmonds' algorithm is certainly bounded by $O(n^4)$ times a polylogarithmic factor. As you'll see, this dominates the running time of the other steps of Chrisofides algorithm, resulting in Christofides algorithm also running in $O(n^4)$ times a polylogarithmic factor.

4.4 The Algorithm

Step 1: Create an MST T of the input graph G in the same manner as for approximation algorithm 1.

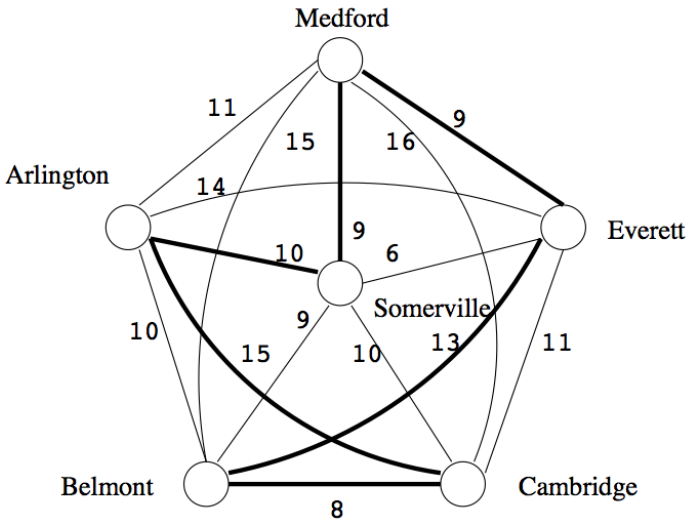

Figure 5: Minimal matching on vertices with odd degree in the MST

Step 2: Take G restricted to vertices of odd degree in T as the subgraph G^*. This graph has an even number of nodes due to Fact 1 and is complete because we have not removed any existing edges between the vertices that remain.

Step 3: Find a minimum weight matching M^* on G^*. This is possible in polynomial time due to Fact 2. See figure 5

Claim 4.4.1 $w(M^*) \leq .5 * OPT$

Proof:

- The solution S^* to the TSP on G^* has weight at most OPT due to the triangle inequality. That is, removing vertices on a path cannot increase the weight of the path.
- Since $|V_{G^*}|$ is even, S^* can be divided into two alternating paths that are both matchings. The lighter of these two matchings M_1 has at most weight $.5 * w(S^*) \leq .5 * OPT$.

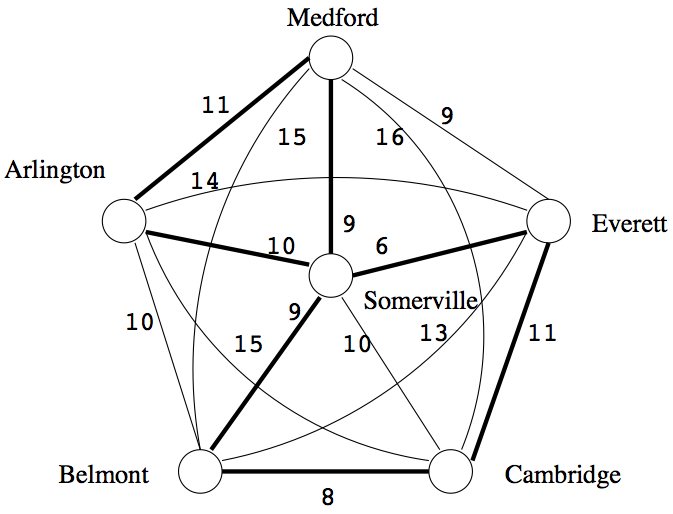

Figure 6: Union of MST and minimal matching

- Since M^* is a minimum matching, $w(M^*) \leq w(M_1) \leq .5 * OPT$

Step 4: Union the edges of M^* with those of the MST T to create a graph H with all vertices having even degree. This is a subgraph of G and has a most weight $w(T) + w(M^*) \leq 1.5 * OPT$. See figure 6

Step 5: Create a Eulerian Tour on H and reduce it to a plausible solution C using the triangle inequality as described in approximation algorithm 1. See figure 7

5 Food for Thought

1. Is 1.5 a tight analysis of Christofides algorithm? Perhaps its worst case is actually better than 1.5 but we have failed to analyze it correctly. *Challenge:* Find an example input for which Christofides algorithm will generate a solution that is exactly 1.5 times optimal.

2. Is 1.5 the best that we can do in polynomial time without having to prove P = NP? If a lower bound does exist, what is it?

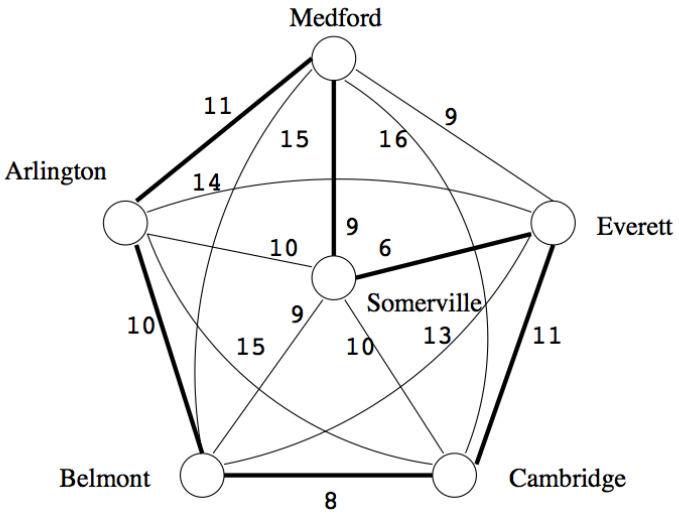

Figure 7: Solution with weight less than $1.5 * OPT$

6 Euclidean TSP

There is a special case of Metric TSP called Euclidean TSP. In Euclidean TSP, the weight of edges corresponds to ordinary (Euclidean) distances between their endpoints in the plane. Unfortunately, this problem is also NP-hard.

Vaidya has found an algorithm to find a minimum weight matching in Euclidean TSP that is $O(n^{2.5}log^4(n))$ [4]. This causes the overall running time of Christofides algorithm to come down to the same asymptote in the case of Euclidean TSP.

Sanjeev Arora has found a Polynomial Time Approximation Scheme (PTAS) for Euclidean TSP [1].

References

[1] Arora, Sanjeev. 1998 *Polynomial time approximation schemes for Euclidean travleing salesman and other geometric problems* ACM vol 45,

num 5. pg 753-782.

[2] Christofides, N. 1976 *Worst-case analysis of a new heuristic for the traveling salesman problem* Symposium on New Directions and Recent Results in Algorithms and Complexity. J.F. Traub, ed. pg 441.

[3] Edmonds, D. 1965 *Paths, Trees, and Flowers* Canadian Journal of Math vol 17, pg 449-467.

[4] Vaidya, P.M. 1989 *Geometry helps in matching* SIAM Journal on Cmoputing vol 18, num 6, pg 1201-1225.