Problems for Recitation 9

1 Traveling Salesperson Problem

Now we're going to talk about a famous optimization problem known as the Traveling Salesperson Problem[1] (TSP). Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route that visits each city exactly once, and returns to the starting city?

One special, though very natural, case of this problem is when the costs of traveling between cities obey the *triangle inequality*. That is, if a, b, and c are distinct cities, then $d(a, c) \leq d(a, b) + d(b, c)$. This corresponds to our intuitive notion that the distance of traveling to a from c should be no larger than that of first traveling to a from b, and then from b to c. The triangle inequality holds if the cities correspond to points in the plane, since in this case the line segments joining a to b, b to c, and a to c form a triangle, and in high school we learned that the sum of any two edges of a triangle is larger than the third edge.

Let us abstract away the distractions and formulate the problem as follows. The cities will be vertices of a graph. We will then consider the complete graph on these vertices, and give edge (a, b) a weight of $d(a, b)$. We want an ordering of the vertices $(v_1, \ldots, v_n)$ that minimizes

$$COST = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) + d(v_n, v_1).$$

Consider the following greedy algorithm for TSP. This might be the first thing you'd think of. It is called greedy since at each step we are choosing the locally-optimal best way to continue, though our overall actions may not be collectively optimal.

1. Start at an arbitrary city.
2. While there is still an unvisited city, go to the city with smallest distance from the current city. If there are no more unvisited cities, return to the starting city.

Even when the cities are points in the plane, the greedy algorithm sometimes outputs a suboptimal solution.

[1] Note that this is sometimes referred to as the Traveling Salesman Problem, but we take a gender-neutral stance.

For example, take $4N$ points on the perimeter of an $N \times N$ square (each at distance one from the next along the perimeter) and add one point at distance 2 (outwards) from the perimeter at each of the corners (for a total of $4N + 4$ points). (To make it consistent with our abstraction, one would take these points on the plane and consider the complete graph with their pairwise distances as weights, but its easier to think of it directly in this case). The Greedy Algorithm will go around the perimeter and then have to go around again to pick up the four corners (which will be skipped the first time around). For large N, this will be nearly a factor of 2 from optimal (which will pick up the corners as it goes around the first time).

It turns out that if you can design an efficient algorithm which solves TSP on every possible input, you can win a million dollars. Many researchers believe that in fact this is impossible, though no proof of this fact is known (this proof would also get you a million dollars). Thus, researchers have contented themselves with finding a good *approximation algorithm* to TSP. More precisely, if OPT is the cost of the optimal solution (the total distance traveled on an optimal route), then an algorithm outputs an α-approximation to TSP provided its output, on any input, is a valid round-trip route visiting each city once, and has cost at most $\alpha \cdot OPT$.

In what follows, we will describe a 3/2-approximation algorithm ALG for TSP, in the case where the triangle inequality holds and the distances are non-negative.

3/2-Approximation Algorithm ALG

1. Construct a graph G whose vertices are the N cities with an edge between every pair of cities $A \neq B$ with corresponding weight $d(A, B)$, where $d(A, B)$ is the distance between A and B in the plane.

2. Compute an MST T of G (Recall that a *minimum spanning tree* (MST) of a graph G is a spanning tree whose sum of edge weights is as small as possible).

3. Compute, for each city $A \in G$, the degree d_A of A in T.

4. Let $S = \{A \in G : d_A \text{ is odd}\}$.

5. Compute a minimum weight perfect matching M on the vertices in S (using the distances $d(\cdot, \cdot)$ as weights).

6. Compute a new set of edges $E' = M \cup T$. Note that the resulting graph $G' = (V, E')$ is not necessarily a simple graph since it might contain multiple edges.

7. Take the subgraph $G' = (V, E')$, and compute an Euler circuit on it.

8. Use the Euler circuit to give an induced ordering of the vertices (i.e., the order in which the vertices appear for the first time), and do a TSP tour on this order.

Try to run this on the $4N + 4$ vertex example and see what it gives.

Note that this algorithm is reasonably efficient, even for large graphs, since computing an MST, computing a minimum weight perfect matching, and computing Euler circuits can be done efficiently. Since the Euler circuit visits all vertices and then returns to the starting vertex, the output of our algorithm visits every city once, and returns to the starting city, as needed. Now let's show that the algorithm is a 3/2-approximation.

Let OPT be the optimum TSP tour cost.

1. Show that the cost of any tour is at least the cost of an MST of G. Hence conclude that the cost of an MST is at most OPT.
2. Prove that the size of S is even.
3. Prove that the weight of the min weight perfect matching is at most $OPT/2$. *Hint: Consider a minimum cost TSP tour on just the vertices in S.*
4. Prove that G' has an Euler circuit, and its cost is at most $3OPT/2$.
5. Show that the length of the TSP tour is at most $3OPT/2$.
6. Conclude that our algorithm outputs a 3/2-approximation.

In fact, for any constant $\epsilon > 0$, it is known how to efficiently achieve a $(1+\epsilon)$-approximation in the plane, but it is a **lot** more complicated.

6.042J / 18.062J Mathematics for Computer Science
Fall 2010