Amelia Graves
Kevin Huddleston
Michael Sellers

MATH/CMSC 420
March 10, 2020

Project 2

The Traveling Salesman Problem (TSP)

With a list of cities and distances between each city given, what is the shortest route one can take to visit each city once and return to the starting city?

Making this into a graph theory question is straightforward: turn cities in to vertices and the distance between each city into weighted edges. Begin searching for the shortest Hamiltonian circuit (or tour). Generally, these graphs are complete, but do not need to be to become NP-Complete.

NP-Complete Proof

NP Proof

Proof.

Lets look at the decision version of TSP, which has a caveat. It asks if the potential solution has a total weight less than an input m. This input is determined by algorithms talked about later.

Let G be our graph and m be our upper bound. Given a possible solution, we need to verify or reject two things; is this solution Hamiltonian and is it less than m? To check if each vertex is visited once and if the first and last vertex are the same: we compare two lists, vertices of G and vertices of our solution. Secondly, we check if the sum of the edges is less than m. Both of these can be done in polynomial time. Thus, the decision version of TSP is in NP. ■

NP-Hard Proof

Proof.

We need to show that TSP can be reduced from the well known NP-Complete problem Hamiltonian Cycle (HC). Hamiltonian Cycle $\leq_p$ Traveling Salesman Problem

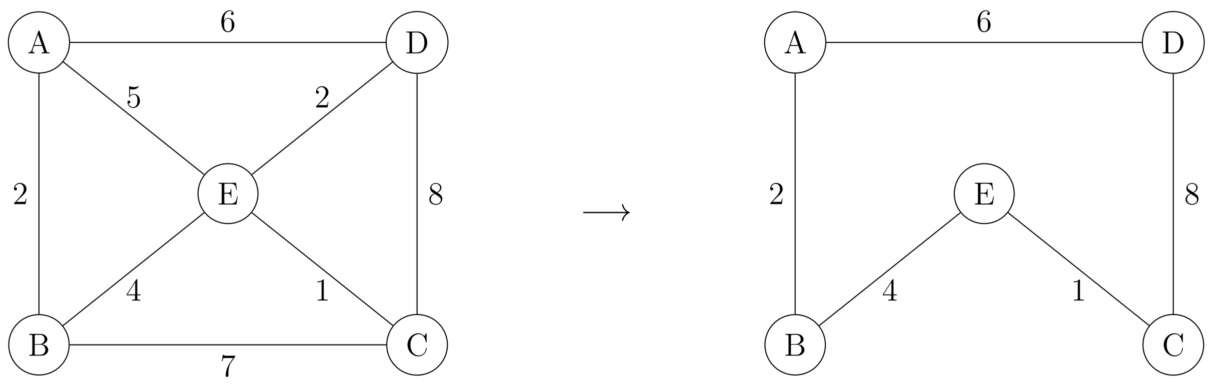
Assume $G = (V, E)$ to be an instance of HC. Let $G' = (V, E')$ be the complete graph of G where each edge in G' that is also in G has weight 0 and every other edge in G' have weight 1. (This process takes $O(n^2)$) Suppose an HC exists in G, its clear that the shortest tour in G' is 0, since it traverses along edges in E. Conversely, assume G' has a tour of 0 weight. Since all edges in G' are weight either 0 or 1, We know each edge must have 0 weight (since there are no negative weights in G'). So, each edge in this tour must be in E. So, G has a HC if and only if G' has a tour of 0 weight. Thus, the decision version of TSP is NP-Hard and is NP-Complete. ■

Algorithms to Approximate Best TSP Solution

Because the only known solution to solving for TSP is to brute force $\frac{(n-1)!}{2}$ possible solutions, scientists have created heuristic (good enough) algorithms to find solutions for the problem. These heuristic techniques are not guaranteed to find the optimal solution.

Nearest Neighbor Algorithm

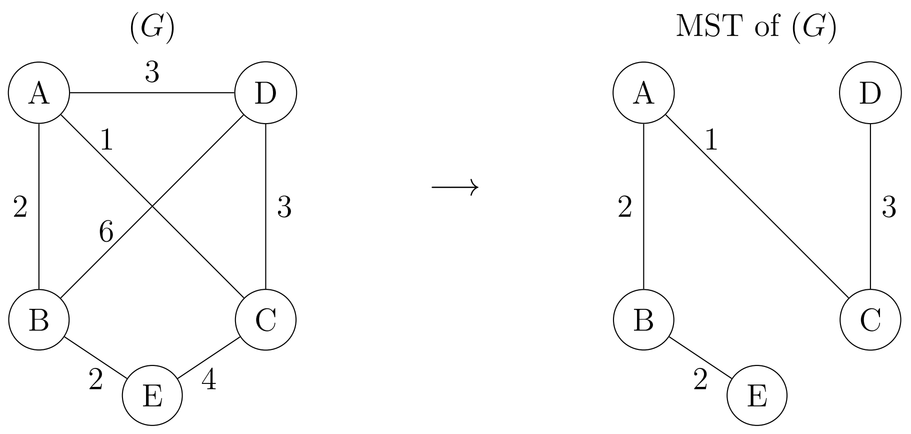
Nearest neighbor is one of the most common ways to solve this problem. First you pick a starting point, then pick the shortest edge from that point to the next. Find the next shortest edge to a city that has not already been visited. Repeat until the last vertex left is the starting point.



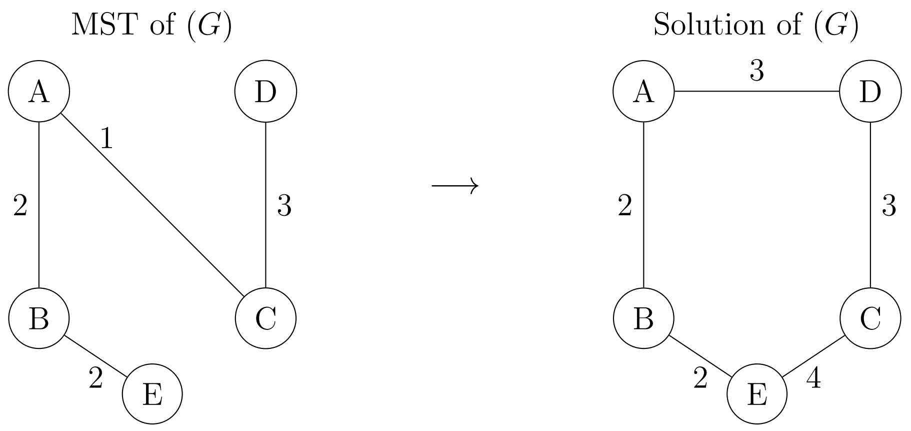
Given the above graph on the left, to start the algorithm, suppose we call A our staring point. We look at all the edges of A and pick the shortest edge, (A,B). At B we repeat looking for the shortest edge. Since the shortest edge at B is (B,A), we can't use it, so we go to the next closest edge, (B,E). We continue the process until we hit point D. We have no other points that we have not visited, therefore we go back to A, adding (D,A).

Using MST and Pre-order

Given a graph, find the minimal spanning tree of the given graph. Now that we have the minimal spanning tree, use pre-order traversal to generate the path as our solution (and add the root to the end so that it is Hamiltonian).



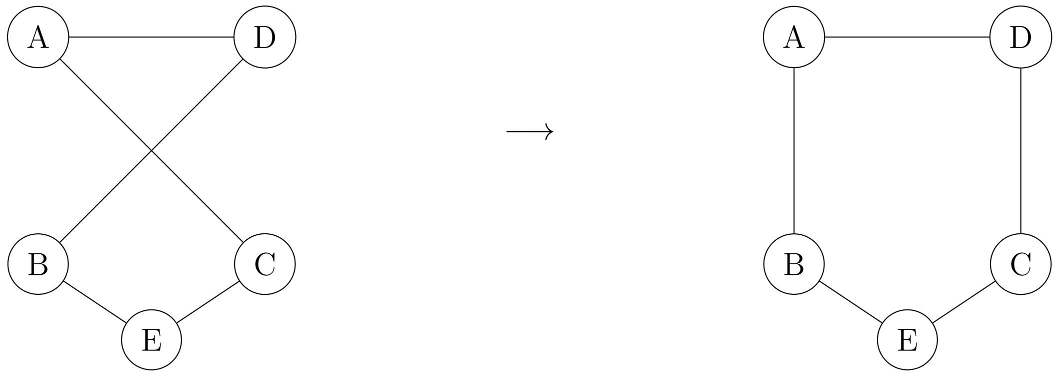
So now our pre-order traversal with our root being A is A-B-E-C-D. The solution becomes, A-B-E-C-D-A. We need to add any edges back in that follow the flow of the pre-order traversal and remove any that do not follow it. For instance, we need to add (E,C), (A,D) and remove (A,C).



This solution will be no better than the minimal spanning tree and can be no worse than twice the cost of the most optimal solution.

2-OPT Swap

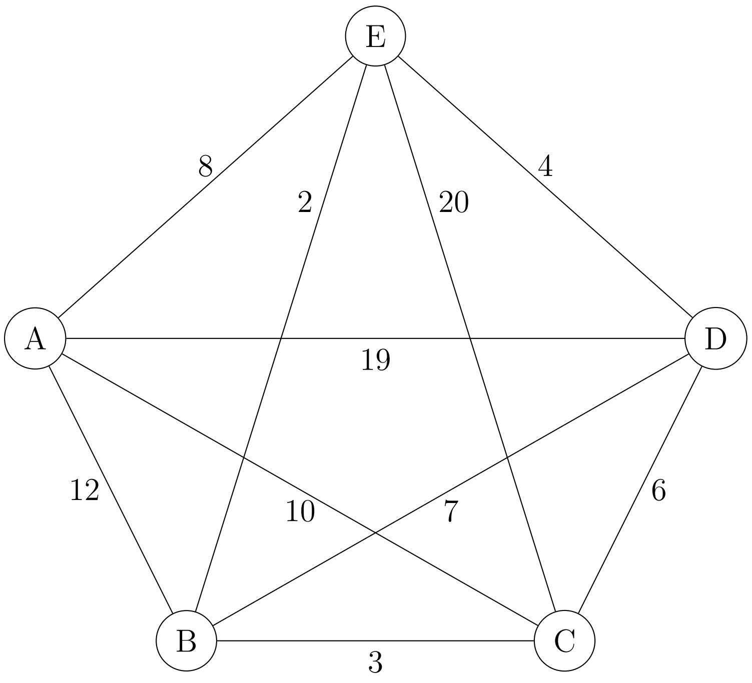
To implement the 2-OPT swap consider a random Hamiltonian cycle drawn on a graph as a potential solution to TSP. If there are two edges that intersect, reroute the edges such that they no longer intersecting, the graph remains Hamiltonian, and the swap results in a shorter tour. Repeat until there are no more swaps to be made.



As you can see we need to swap edges (A,C) and (B,D). There are two possible swaps to make for edge (B,D), however, to keep the graph Hamiltonian we must reorder to (A,B) and (C,D). Since this requires us to look at sections of the graph,like in the example we disregarded vertex E, this will produce a local optimum, which is not guaranteed to be the optimal solution.

Activity

Below is the graph being utilized for the activity. Each group can use one of the algorithms from above, or try to find the optimal path for traveling. Give the class 3-5 minutes to work, then see what paths and how long the path they found was. once everyone shows their paths, show them the optimal path is EACBDE with length of 32.



Applications

TSP is an optimization problem to hit all points of interest and return to the start in the shortest distance, these points do not have to be cities. Direct applications of TSP can be planning city or school bus routes, placing power cables, or finding the shortest path from transmitting data from your computer to a server. Astrologists use TSP to minimize the fuel usage of satellites to target different points in space. Genetic engineers apply TSP to convert a collection of DNA strings to create universal strand with minimal length. Given the complexity of this problem researchers in the last 50 years have used computers to find the optimal tours with an increasing amount of vertices. Their progress is captured here. In 2016 researchers were able to find the optimal UK pub crawl featuring almost 50,000 pubs. In 2015 researchers found the optimal tour to visit almost 50,000 historical sights in the US.

References

[1] William Cook, *The traveling salesman problem*, www.math.uwaterloo.ca/tsp/index.html, 2019.

[2] Eric Kuang, *A 2-opt-based heuristic for the hierarchical traveling salesman problem*, 2012.

[3] Emre Kucukoglu, *Tsp is np-complete*, 2016.

[4] N/A, *The traveling salesman problem*, N/A.

[5] PranamLashkari, *The travelling salesman problem set 2 (approximate using mst)*, 2018.