CS170 Project Design - Matthew Tang, Christianna Xu - 11/22/19

1 Algorithm

Our final algorithm will consist of two parts: a novel algorithm called the Footsteps algorithm and a variant of Christofides' with an additional condition to see if dropping someone off is more optimal. The best output between the 2 algorithms will be saved.

1.1 Walking vs Driving

Note that we never want to traverse a path that is to only drop off 1 person. It would be more energy efficient to just drop that person off($d < \frac{2}{3}d + \frac{2}{3}d$).

1.2 Footsteps Algorithm

1) First let everyone walk (no car used) to their homes using shortest paths and an adjacency matrix S to keep track of how many people cross each edge on their shortest path. If multiple people walk on the same road that means their homes are located in the same direction so it is optimal to drive on that road.

2) We want our car to travel through edges that are walked on by at least 2 people. Create $U \subseteq V$ which is all the vertices that are the ends of edges walked on by at least 2 people.

3) Run all-pairs shortest path (APSP) between all vertices(V).

4) If we are at vertex v, use S to find the edge (v, u) that has been traversed the most. Denote this as the "greedy step".

5) For every edge $(v, w), w \neq u$ run DFS to a depth of 5 and add all home vertices that you visit to a list H

6) We simulate the "greedy step" multiple times. Each time, use APSP to calculate the sum of distances to all vertices in H. If this value increases over time, then don't use edge (v, u) and instead find the shortest path to the nearest unvisited home using A^*

7) Run steps 4-6 until everyone is driven home. Run the path of the car dropping people at the closest vertex(using APSP) to their homes and calculate the total energy cost.

1.3 Christofides' Variant

1) Create an MST M of $G(V, E)$

2) Create O, a Minimum weight perfect matching of the odd degree vertices of M.

3) Combine the edges from M and O to form H

4) Find an eulerian path in H

5) If we've already visited the next vertex in the eulerian path, find the next vertex that we haven't visited and calculate the shortest path (A^*) to that next vertex

6) We can keep track of 1-person drop offs by adding all edges we traverse onto a stack. If the current edge traversed = top edge on the stack, pop the edge add this to a temporary list. Keep doing this until your traversed edge differs from the top of the stack. Instead of going through that route and back, subtract the cost of that traversal and back and add the length of the path (equivalent to dropping that person off)