CS 170: Project Design Doc
Fall 2019

Alex Chan
Brian Stone
Jinwoo Park

Knowing that the problem is NP hard, we realized that the problem would be much easier given the optimal drop off locations. Therefore, one solution might just be using a random set of drop-off locations and tweak that to get better energy consumption. It might fail if the drop-off locations we pick are completely off but could work if we run it multiple times and have some way to narrow which locations we pick.

Structure of the algorithm:

- Generate a matrix describing the shortest path between every set of vertices.
- Create sets of drop-off locations to test.
- Run descent algorithm and choose the best performing one.

Descent algorithm:

- Trim off some unnecessary vertices - Reduction strategy
    - Remove dead-end roads with no homes (when a vertex has higher distances to all homes than an adjacent vertex). Achievable in O(mn^2).
- Generate a random set of drop-off locations and some “good” initial set of drop-off locations.
- Compute the energy consumption of the optimal path using these drop-off locations.
    - Use one of fast TSP algorithms
    - Nearest NN, O(n^2), with 2-opt, expected O(n) moves.
        - “Expected to perform 5% better than Christofides algorithm.”
- For every set of drop-off locations satisfying one of these conditions, compute the optimal energy consumption: let set of homes be X and set of locations be V.
    - [X \ v] for v in V. Gives us O(|V|).
    - [X U v] for v in V if size of X is not the maximum. Gives us O(|V|).
    - [(X \ v) U u] for u,v in V and u and v are distinct Gives us O(|V|^2).
- Choose the one giving us the minimum energy consumption and repeat the whole process. Terminate the algorithm the current set is the optimal one.

We know that this algorithm produces a “local-min” solution. Run this algorithm with many randomized initial inputs and hopefully one of these local minimums is the global minimum.

- One thing we can do it testing this method with a randomly generated initial set of drop-off locations and if the optimal solution occurs multiple times, then the algorithm is probably good.
- We can choose some “good” initial set of drop-off locations such as homes or a single list of the starting location.

Tool for choosing optimal initial drop-off locations for descent search:

- Greatest Roads problem where we make k copies of G (layer count k depends on how we use this) and construct a k-layer graph, which we will eventually run shortest paths on to get the best path starting and ending at the same location. We'll refer to it as G'.
- G is undirected, but we can make each edge in G into two edges in opposite directions. This duplicates the number of edges per layer, and there are k layers, so the total number of edges in G' is 2k|E|. The total number of vertices is k|V|. O(k(|V| + |E|)).
- This alone does not take TA dropoffs into account, as expanding G' to account for all of the possibilities would be intractable.

Tool for finding shortest path matrix (memoization):

- Keep a hashmap<other vertex,length> for each vertex that describes the shortest path from that vertex to the other vertex. Run Dijistra on initialization on every vertex to keep these hashmaps updated -> O(1) for finding the shortest path between two vertices. Equivalent to having a matrix M in R^num(TA) where M_ij = shortest distance between the location i and j. Initialization is very heavy, O(n^4). It should still be faster in all means.
- Useful for computing the distance between drop-off locations and the distance between the drop-off location to homes.
- Reduces the whole problem to finding optimal drop-off locations <= (# TA). Given drop-off locations we find find the optimal combination of TA paths.
- Use a doubly-dictionary if it is better performing.

Given a list of drop-off locations, we want to use a TSP algorithm to find a fast cycle around the drop-off locations. We can spend some more time finding this TSP then the above algorithm because this one is only iterated once. For each home, we also want to choose the best drop-off location which will just take O(# TA).