Meera Mehta
Ismail Javed
Vidhan Jain

CS 170 Final Report: Drive The TA’s Home: Approximation Algorithm for an NP-Hard Problem

I. How the Algorithm Works

Our final algorithm involved reducing the Drive the TA’s home problem to a metric TSP problem, which we solved using an Integer Linear Programming approach. We were able to further optimize this output by considering the case where 2-cycles were occurring in the path, which could be removed, as 1 person walking across an edge to their home in one direction is cheaper than having the car accrue a cost of 4/3 to get to the home and get back.

Our algorithm works by first creating a new set of nodes and edges for a new graph, G', such that V' contains the home indices and start index, and E' contains edges between every pair of (u,v) in V' with weight equal to the shortest path between u and v in G (so d(u, v)). (Note: we also keep track of the nodes that are actually traversed in G for an edge in G' using a dictionary.) Next, we create a model for an ILP using Gurobi. The ILP model variables are Boolean values for each edge, and the constraints are that, for each vertex, it has at most degree 2. That is, the sum of the Boolean values for all edges that contain a vertex v is less than or equal to 2. This ILP model is used to approximate the list of edges traversed that have the optimal solution for the TSP. Now, using these edges, our algorithm performs a DFS starting at the start_index to obtain the path in G'. Using the dictionary that translates edges in G' to paths in G, it transforms this path into a path in G. Finally, the algorithm removes all 2-cycle instances. The cost of this output is also compared with the output of the naive solution/output where each TA is dropped off at the start vertex. The costs for these two outputs are then compared and the optimal one is returned.

II. Design Process/Initial Ideas

Initially, we were interested in reducing DTH to TSP and using the Christofides Algorithm, an approximation algorithm for metric TSP that has an approximation ratio of 3/2, in order to find the optimal cycle through the home vertices. However, we also tried using Integer Linear Programing to approximate this, which ultimately resulted in a more optimal solution. Additionally, there were two main factors we were considering to optimize the cycle through the TA’s homes: finding furthest common ancestors of TA home nodes and finding instances of 2-cycles in the path. Both of these cases involved traveling distances in the car that would always be non-optimal compared to the TA just walking to their home from that point. We decided to look for the 2-cycle instances in our final algorithm since this case provided the optimal solution. In an attempt to further optimize the algorithm, we decided to add a condition to the inner while loop of the remove_two_cycles method to check if the 2-cycle being removed had just one TA being dropped off, rather than multiple, which would make it then not optimal to remove the cycle. This extra check ended up giving a few (about 8) invalid outputs from all the outputs we generated, but on the rest of the outputs, it produced better results. So we used the previous implementation for those outputs on which it failed.

III. Resources Used

The Gurobi library was useful to us for implementing ILP. After reducing the problem to an instance of TSP (of the TA houses), we formulated it as an Integer Linear Program (as described in the first section) and used the Gurobi library to implement it.