RaoTA Design Doc, Phase 1

Kevin An, Joe Zou, Matthew Tran

November 2019

To tackle this project, we started with a naive algorithm and branched out from there. Since we knew this problem was NP-hard as TSP reduces to it, we mainly focused on optimization with accuracy and runtime used as our metrics for performance. From experimentation, we observed a high variance in runtimes and optimal costs for various graphs, so we intend on deploying multiple algorithms and picking the best from each. Here are some of our ideas so far.

1 The Lazy Rao (Naive Algorithm)

Idea: Drop everyone off immediately and have them walk home. On randomly generated graphs, we conjecture that on average, this will do better as there are more TAs. This will serve as the baseline we compare our other algorithms with.

2 The Friendly Rao (Naive Algorithm 2)

Idea: Rao drops everyone off at their house using TSP to determine the path. We haven't tested this algorithm yet, but we plan to use a TSP approximator (localsolver) to find the path Rao takes.

3 Θ(Heat Death of the Universe)

We wrote a brute force algorithm to compare against other algorithms. Conceptually, we use Dijkstra's to explore the graph of possible dropoffs and locations in terms of minimum cost. With some optimizations, we estimate runtime to be $O(n^2 2^n)$ in number of nodes. After nearly frying our computers, we decided that 30 nodes and 15 TAs is as far as we're willing to explore. For small inputs, it achieves around a 60% improvement over the naive solution.

4 Sorta Sorted

Basically, we start by sorting TAs by distance to their homes first. When choosing possible dropoffs, we only drop off TAs that are farther from their homes if we drop off the closer ones too. This reduces the number of dropoffs checked from exponential to linear, which we conjecture brings our runtime to polynomial. Indeed, with some adjustment this algorithm works on inputs of size up to 200. With high variance, it achieves a 25% improvement over the naive solution.

5 Travelling Rao's Men

Idea: Start with the plan to drop everyone off at their house in the order returned by TSP. For every house, consider all points within distance x. If dropping the TA off at any of these points can reduce the energy cost, choose to drop the TA off at the most optimal point.

We wanted to try an algorithm which optimized off of TSP. We knew that for every house, it wouldn't be reasonable to consider every vertex as a drop off point, so we decided on the idea to draw circles of radius x centered around each house as a region of potential drop off places. Our idea for the heuristic x would be for the i^{th} house returned by TSP, x_i would be proportional to the distance from the $(i-1)^{th}$ house to the $(i+1)^{th}$ house.