1 A* search

We started with a naive application of A* algorithm on this problem. Given a list of locations L, a list of homes of all TAs H_0, the problem can be formulated as the following search problem:

· State space: $\{(v, H) | v \in L, H \subset H_0\}$
· Starting state: (s, H_0), where s is the starting location
· Successor function: $(v, H) \mapsto \{(v', H') | v' \in \text{neighbour}(v), H' \subset H\}$
· Goal state: $(s, \emptyset)$,
· Cost function: defined in the spec

In principle, we can solve the problem exactly and guarantee optimality with A* search; However, a qualitative analysis of the size of the state space, which is $\mathcal{O}(|L|2^{|H_0|})$, and the branching factor for each state, which is also $\mathcal{O}(|L|2^{|H_0|})$, shows that we cannot possibly expect A* to terminate in any reasonable amount of time due to the exponential runtime bound $\mathcal{O}(b^c)$, where b is the branching factor and c is the steps it takes for the optimal solution to reach the goal. Indeed, our implementation fails to terminate within hours even for on smallest inputs with $|L| = 20, |H| = 10$.

2 Cluster search

To improve upon the naive A* search solver, we designed the following algorithm, which we will refer to as cluster search, that combines A* with a greedy algorithm to look for suboptimal solution efficiently. The main idea is to significantly reduce size of our state space and branching factors by breaking the problem down to the following(much easier) subproblems.

1. Partition the set of locations into clusters of size at most 10, so that in each cluster there exists a center that is close to all other homes in this cluster. To do this, we first find a spanning tree of the graph, and delete edges to obtain a partition of the graph into connected components. Repeat until we find a satisfactory partition.
2. For each cluster in the partition, run A* search to find a inner tour within the cluster starting at its center that sends home all TAs who live in this cluster.
3. Find an approximate minimum outer tour starting at the starting location that go through the center of each cluster.
4. Piece together the outer tour with all inner tours to obtain a solution to the original problem.

The cluster search solver performs much better than the naive A* search solver, and can find an approximate solution within 2 minutes on average for randomly generated valid inputs of all legal sizes. Compared to randomly generate solutions, cluster search produce an energy cost 30 times lower, averaging over inputs of various sizes.

3 Reduction

After preliminary analysis of the problem complexity, we believe the problem to be NP-hard as there likely exists a reduction from the travelling salesman problem to it. In phase 2, we will attempt to reduce the problem into some NP-complete problems, after which we can make use of available approximate solvers to look for better solutions. Potential reductions that we currently consider include Integer Linear Programming, Minimum Set Cover, and Travelling Salesman Problem.