Alex Rao
David Shau
Austin Sun

CS 170 Phase 1 Design Doc

After reading the spec detailing the problem, the immediate strategy that came to mind was to reduce the problem to the Travelling Salesman Problem (TSP). Our reasoning was that TSP was similar in that it required us to find a tour that visited specific nodes and we knew we could reduce the problem to TSP since the latter is NP-complete. However, reducing the problem to TSP is difficult since the two problems are notably different in a couple ways. TSP requires the tour to visit all nodes whereas this problem only requires us to visit specific nodes (the homes of the TAs). In addition, the problem does not require us to travel to every node unlike the TSP since we can simply drop off the TAs and let them walk there by themselves. As such, reduction was potentially complicated enough that we decided to consider other approaches.

One approach was a modified brute force approach. Brute force would require generating every possible tour, and trying every possible drop off location for every possible combination of TAs. This is clearly too exhaustive, so we considered finding a limited number of random tours instead of all tours. For each tour, we would choose random sets of vertices to be designated as drop-off locations. We choose the best set for each tour, and then choose the minimum cost tour. This strategy therefore lets us find a local minimum in to approximate the global minimum of the solution space.

We also attempted to formulate the DTH problem as a linear problem, since there are many powerful solvers out there, but have not successfully done so yet. Since Simplex is assumed to be a good optimizer, we thought that this would be a good idea.

We lastly considered an approximation algorithm. We would take around 1.5 times the number of TAs and select that many random vertices. We would then run DFS on the MST of those vertices and, for each vertex, run the All-Pairs algorithm to get the distances between that vertex and each TA's home. We will keep track of which vertex is the most optimal for each TA in terms of dropping them off, and return the tour with the minimum cost. The main issue is that although DFS does give you a tour, there are likely other edges that are more optimal to take other than the back edges in the tree. This method gives us the best dropoff locations for each TA given a tour, which we thought would be a good approximation assuming the chosen tour is decent.