Describe the algorithm you used to generate your outputs. Why do you think it is a good approach?

Our outputs were generated using a mixture of three greedy algorithms, each using a specific heuristic based on greedy leaf deletion. Because we predicted we would have issues with formulating a valid network from the bottom up, we started our algorithm by reducing G to one of its spanning trees. Then, every iteration of our algorithm would decrease the average pairwise cost as much as possible with the removal of a single leaf. The reason we focused on leaves is because we always began with a valid network (some kind of spanning tree) and realized that removing only leaves would maintain the connectedness of the working network. However, we still needed to check if removing the leaf would create a non-dominating set. The heuristics used were greedy leaf deletion from a random MST, from the SPT rooted at the node with lowest average outgoing edge weight, and from the SPT rooted at the node with highest degree.

For the random MST, the intent was to start with a low weight tree that was already a valid network and greedily reduce the number of nodes in the tree until there were no more leaves that could lower the average pairwise distance while maintaining a valid network. We started with the MST as we expected a tree of lower aggregate distance to produce a low average pairwise distance. We also realized that while the MST has minimum cost for a spanning tree, it does not necessarily contain the cheapest path between two given nodes. If we could choose a node that was likely to end up in the final network, then the shortest path tree rooted at this node could produce a cheaper network than the MST itself. For the SPT rooted at the node of lowest average outgoing weight, the intuition is that our SPT would include a node with mostly low-cost edges but would not punish the node for having a few higher cost edges. Although the node's incident edges may not be the cheapest in the entire graph, they give us a good connected baseline to root our network in. For the SPT rooted at the node with highest degree, the intuition is that this node is "highly-trafficked" so even though it may have high outgoing edge weights, it will potentially be a central connection in the network and reduce the size of the dominating set.

We decided to take the best network from several heuristics because we expected greedy approaches to have relatively fast runtime and reasonable accuracy, but at the cost of perfect correctness due to their short-sightedness. Since each of our greedy algorithms had good runtime, we were able to manage the issue of correctness by using all three of our greedy approaches at the same time. Each of our greedy heuristics would excel and perform poorly in different conditions, so by checking multiple heuristics and picking the best one for each input we would be able to combine the strengths of our three approaches.

What other approaches did you try? How did they perform?

We tried a number of greedy and randomized algorithms before deciding on the three listed above. Since it is not always beneficial to remove leaves — having more nodes than necessary to provide cell coverage can decrease the average pairwise distance — some of our approaches relied on probabilistic removal of nodes. The probability of removing a given node would be a function of node degree, incident edge weight, or some combination of the two and was intended to produce networks that did not simply have the minimum number of nodes. These approaches did perform better than our other algorithms on certain inputs, but not well enough to justify their added runtime during testing. While the idea behind the probabilistic approach seemed to hold some merit, we observed during testing that redundant nodes in a network almost always had a negative impact on average pairwise cost, even though they increased the number of vertex pairs. Hence, probabilistic approaches that tended to keep redundant nodes in the network generally performed much worse than our deterministic and greedy removal approaches.

What computational resources did you use? (e.g. AWS, instructional machines, etc.)

We used our personal laptops and did not use any outside computational resources.