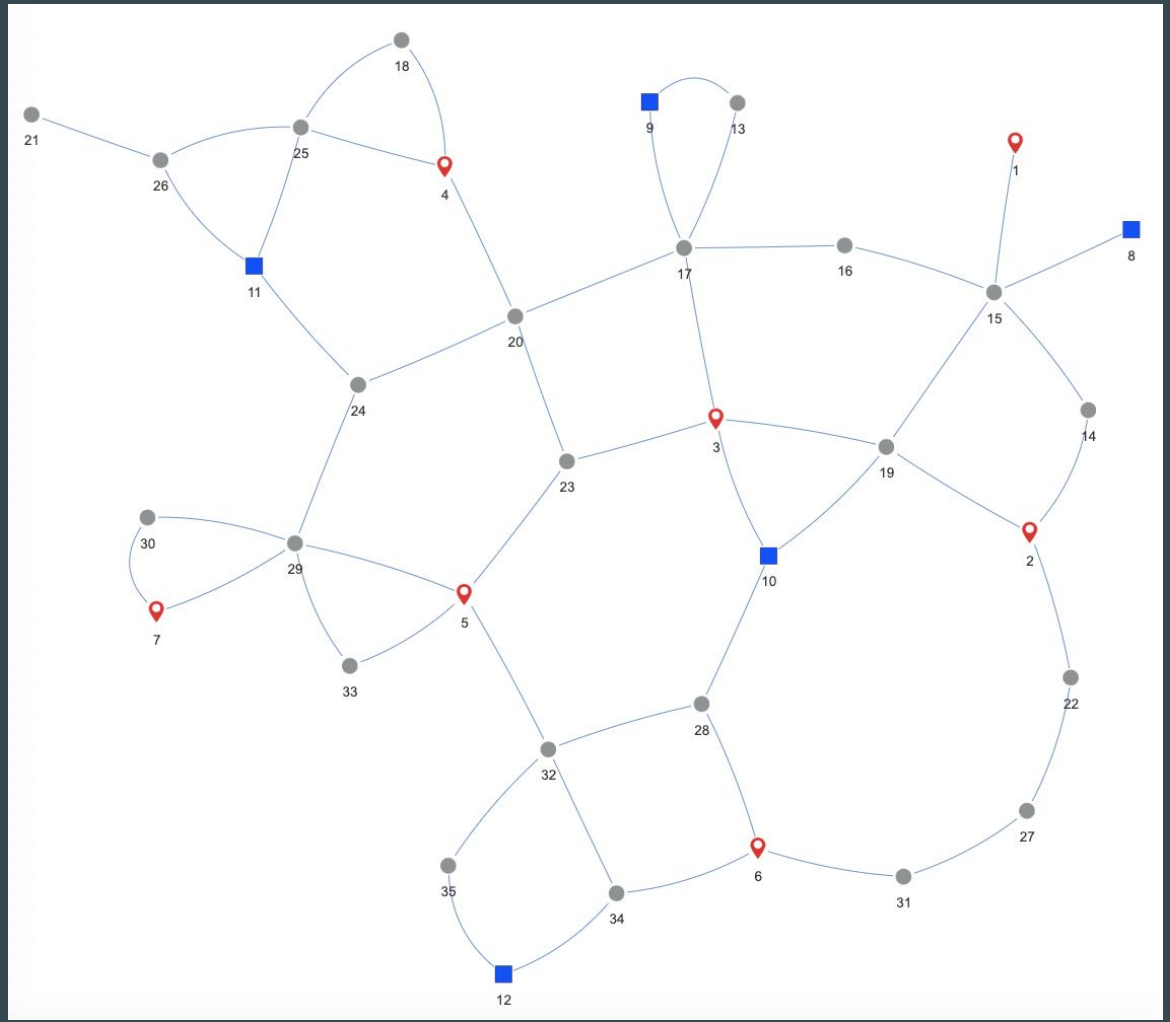
Routing Algorithm

Kick-Ass in the Pacific Theatre Team

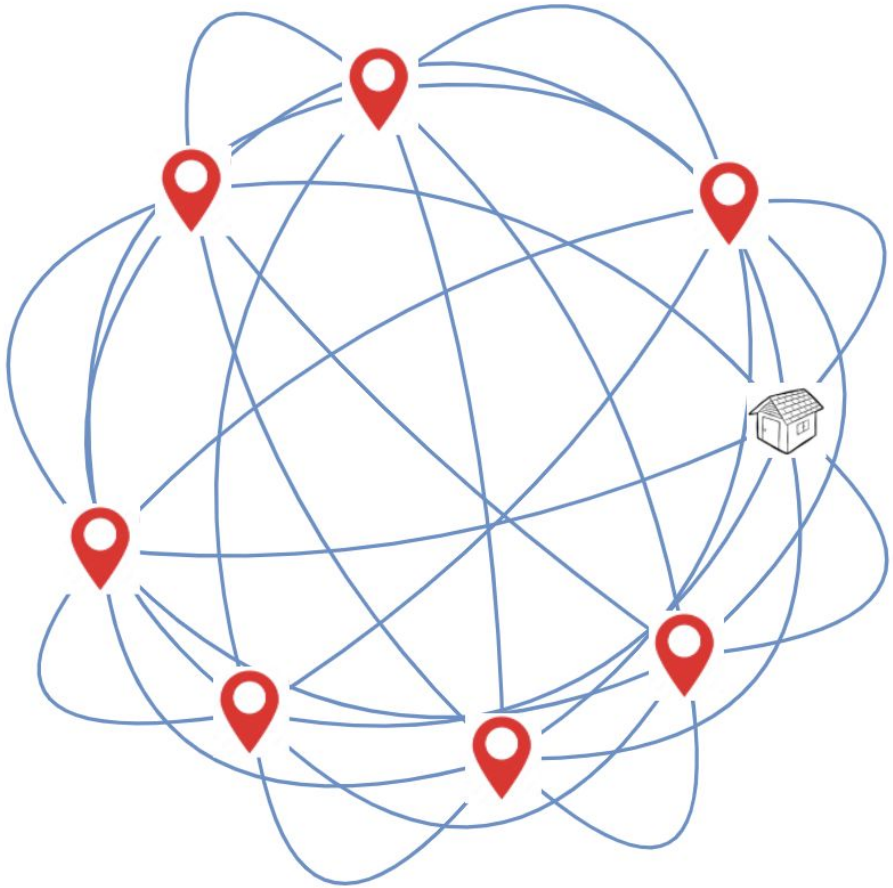
Overview

- The purpose of this is presentation is to describe the algorithm I provide for routing with visuals for easier understanding.
- In creating this, it helped me to understand the process better and I hope that after this presentation, we may view this algorithm as a black box and use it as a subroutine as we develop the model further.
- Also, this could be useful when we present down the line and write our paper.



ORIGINAL GRAPH

We begin with an original graph. The red pin icons represent assembly points, the blue squares represent sinks, and the grey nodes represent other nodes that make up the network but do not have any special designation.

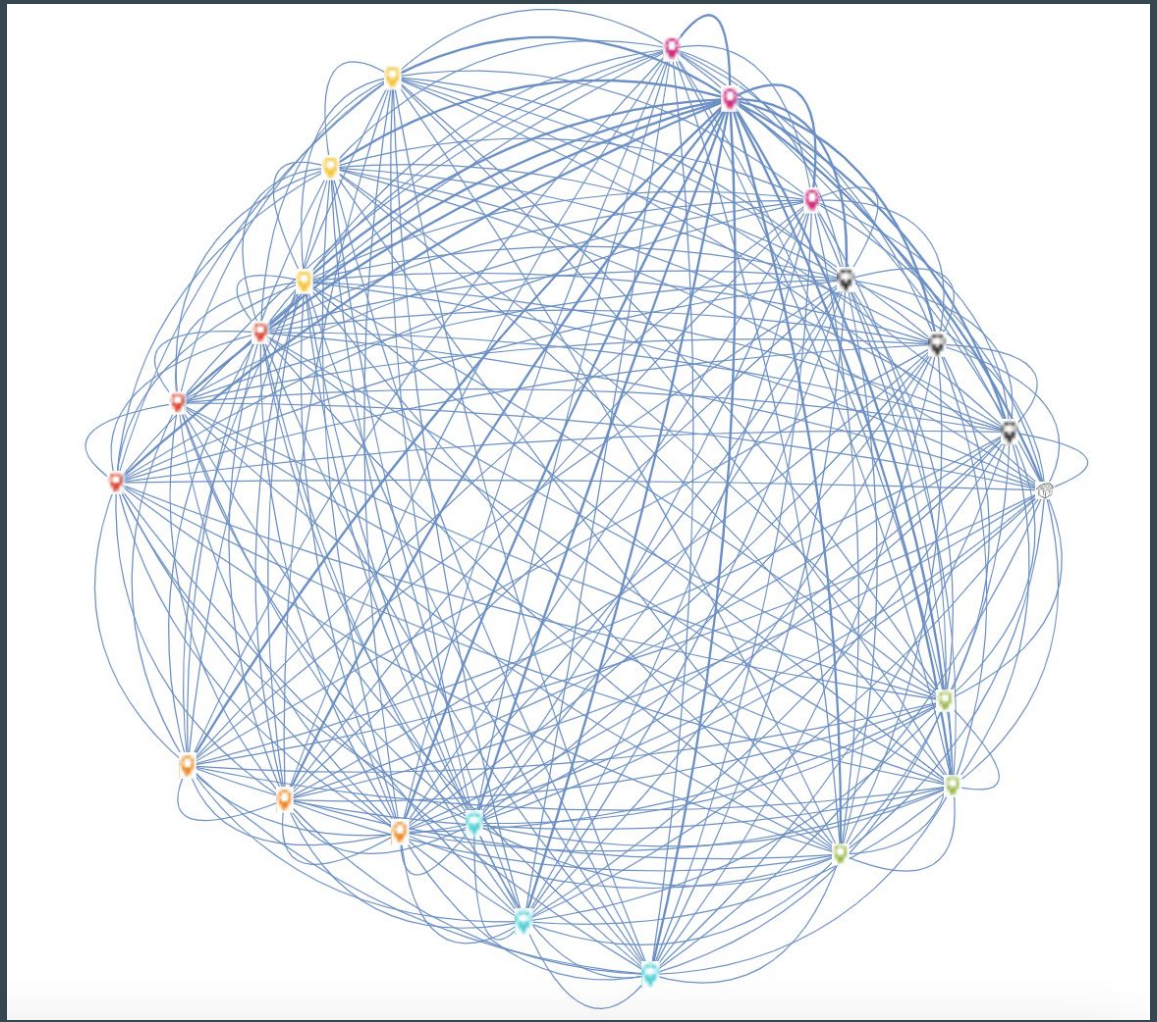


Decrease the Complexity!

We like complete graphs because even though the in and out degree of each node may increase, the graph is made more uniform.

Between each AP there is a directed edge, with weight given by taking the minimum of the summed distance from the pre-AP to a sink and the same sink to the post-AP.

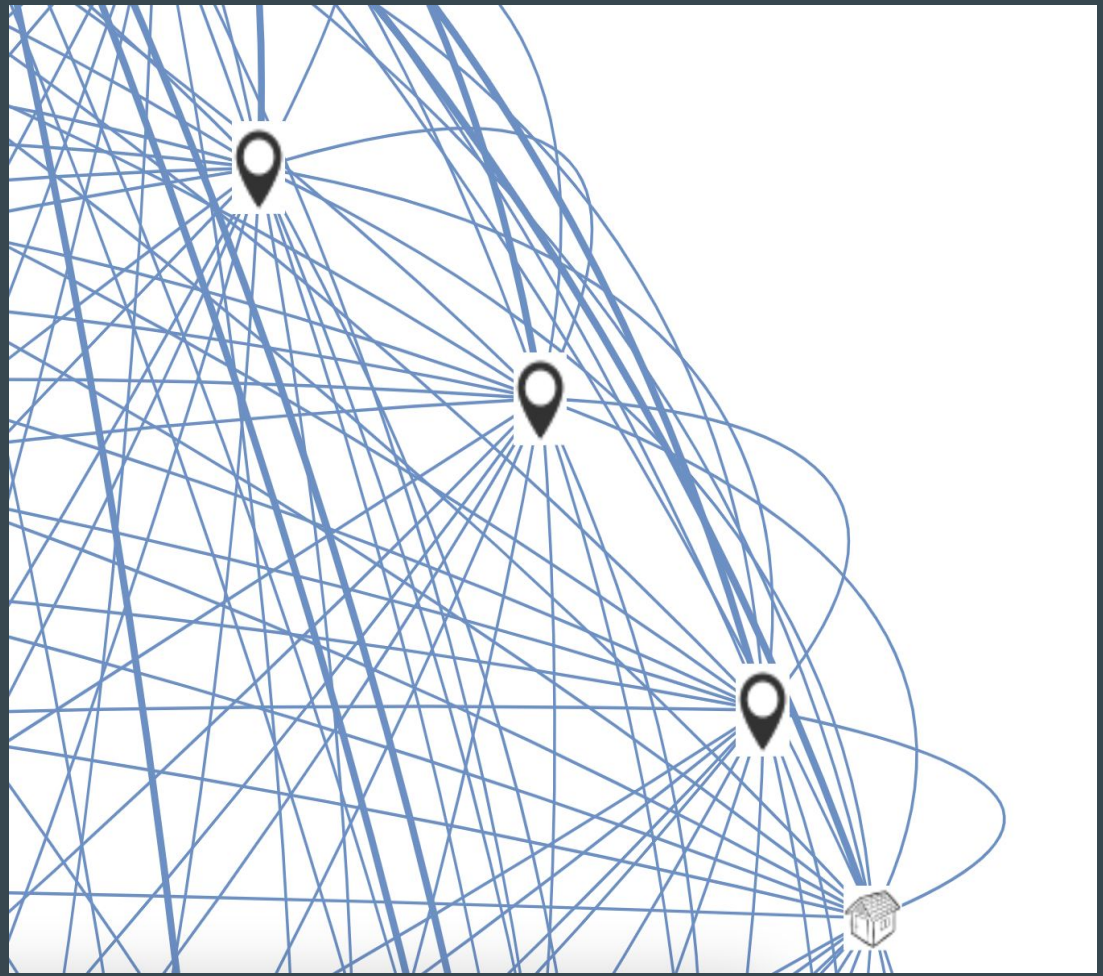
We also introduce a vehicle depot. The distance from the depot to any AP is 0, while the distance from any AP to the depot is the distance of the nearest sink to that AP.



Increase the complexity!

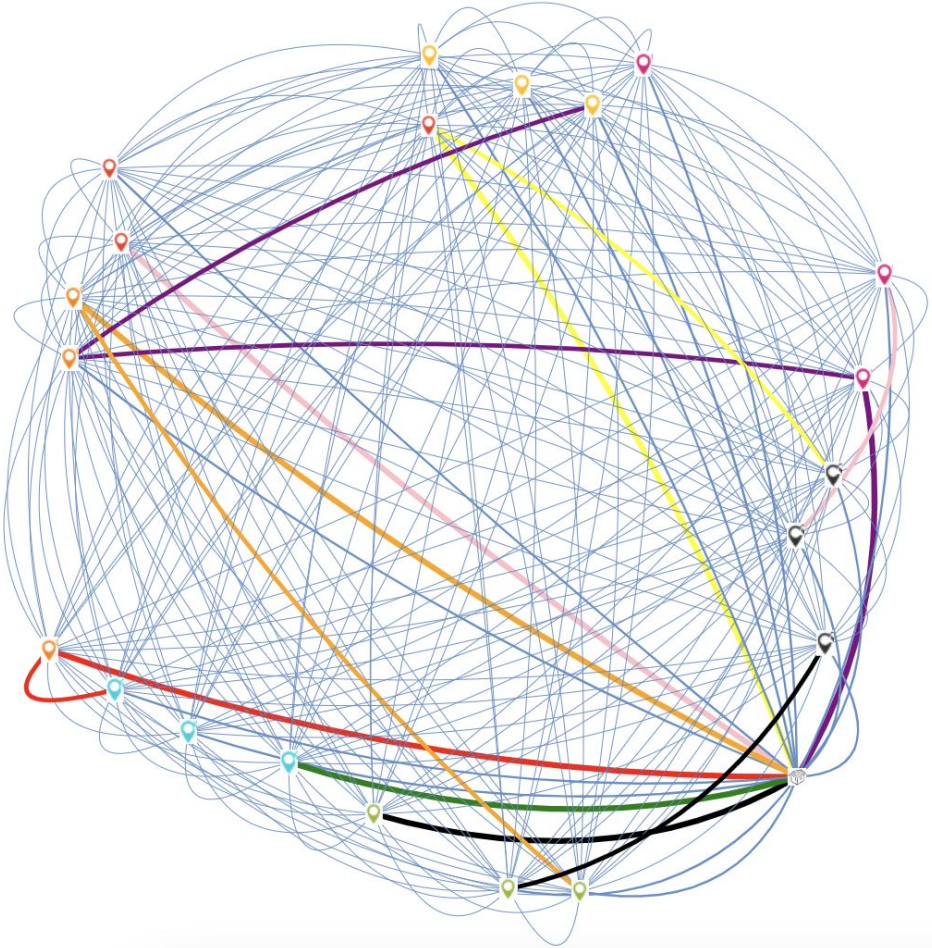
We now duplicate the assembly points. If C is the capacity of the vehicle, and f(v) is the number of people to gather at assembly point v, then the result of the duplication turns v into roughly f(v)/C copies of itself. Remainders are taken care of in the code but don't add to intuition.

We maintain the completeness of the graph in this duplication, so the complexity increases because we increase the number of nodes and edges, but the type of graph still remains in our favor.



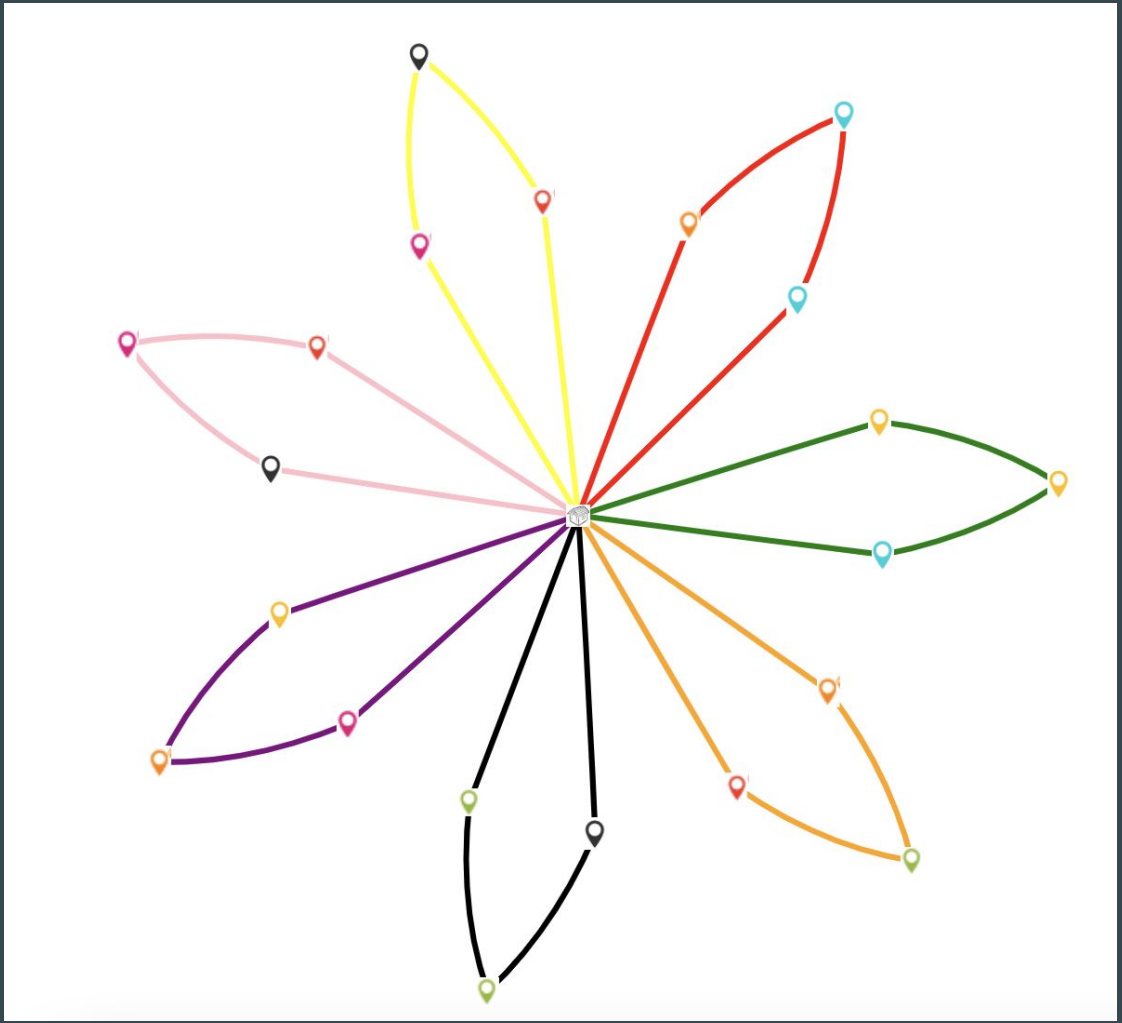
Time Intervals!

In the duplication step we also introduce time windows, which we do by using gamma random variables. Note, to each AP there is assigned a rate which is reflected in the code by a function g(i) which takes in an assembly point and spits out a rate.



Vehicle Routing Problem!

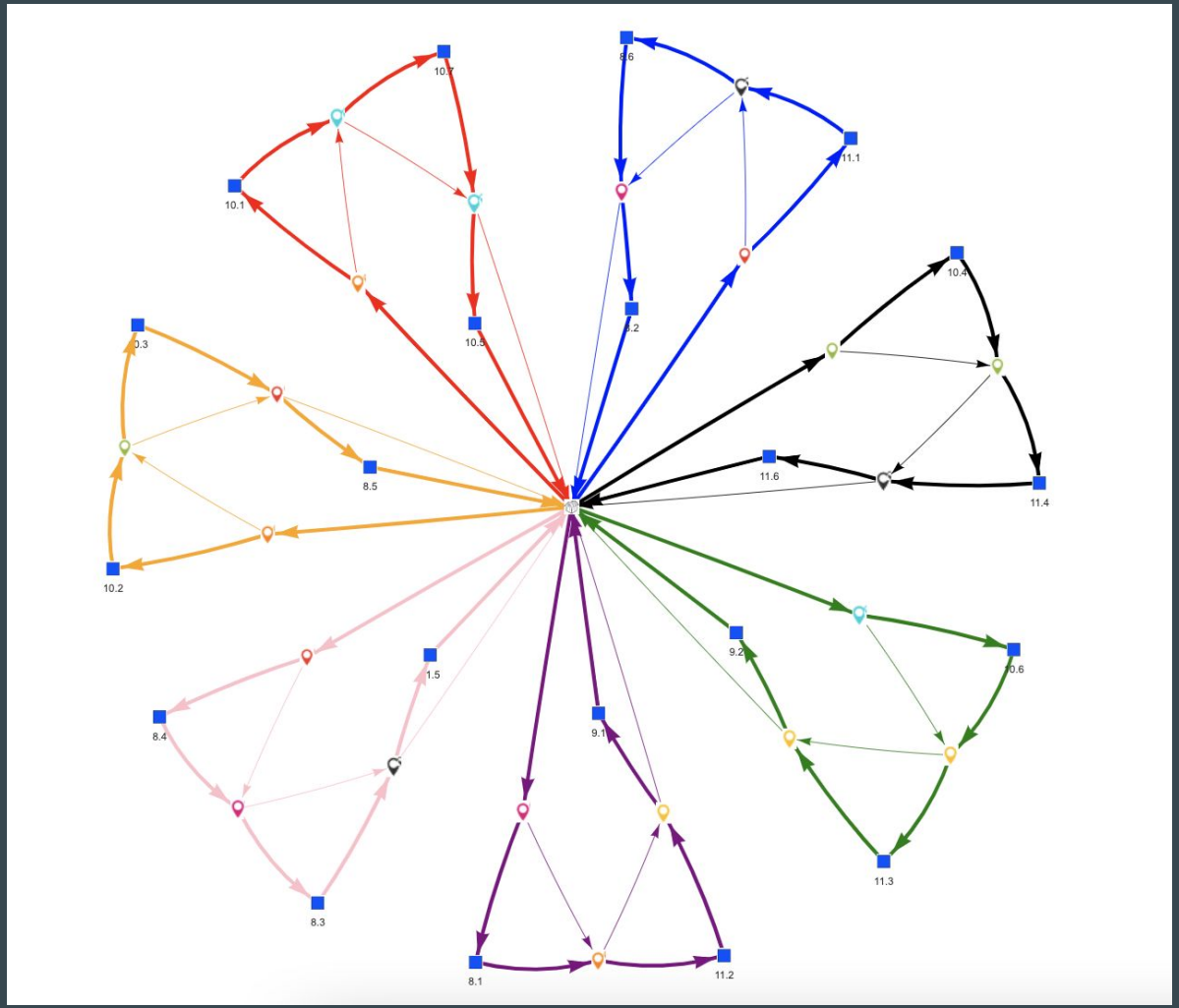
We have now transformed this into an instance of the vehicle routing problem with time windows. We solve this in the code with a hybrid genetic search algorithm. Note in the schematic shown some of the edges in the routing scheme don't render, but it is valid as seen in the next slide.



Journey out of Hell

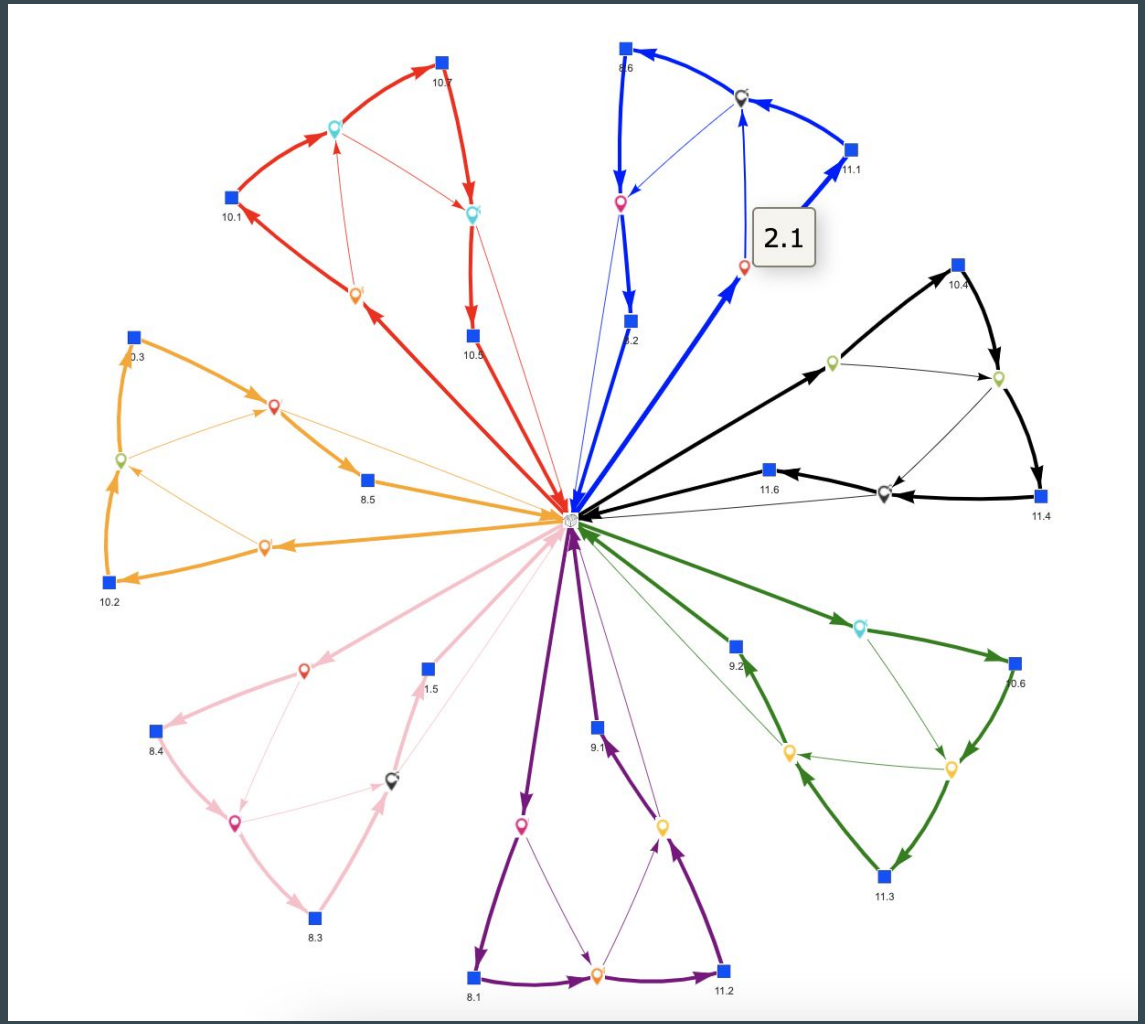
Now, from the previous complete graph, we prune any edges that aren't used in the routing scheme. This is what we are left with.

To finish off, we just need to reconstruct the sinks in each route



FINAL OUTPUT

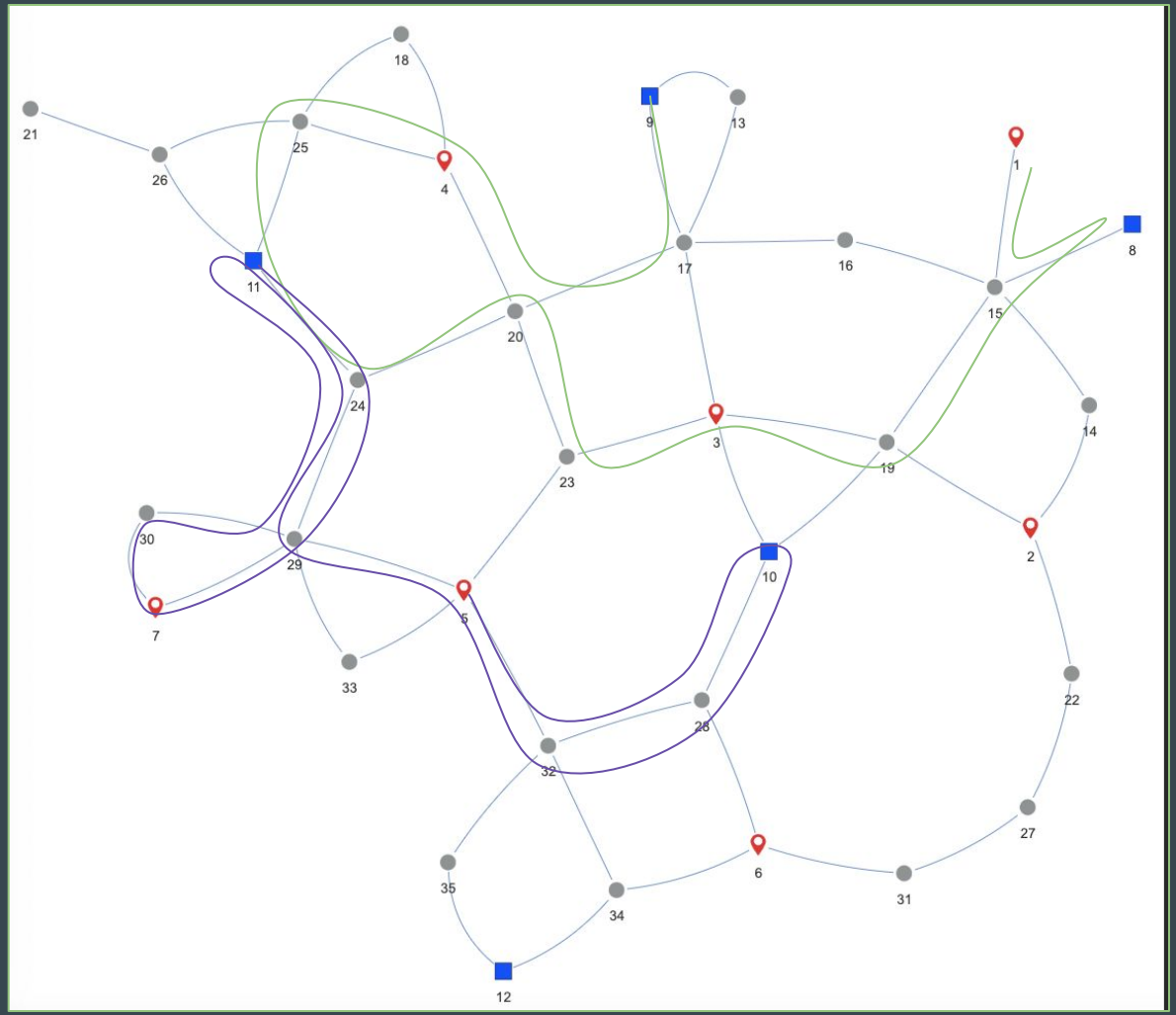
The final output of the algorithm is given. It is a graph object,and a proper routing document can be formed very easily from it.



Book-keeping!!

Note that we properly book keep all our info about the AP's and sinks throughout this process. If you run the HTML page and hover over any of the assembly points, it'll give you i.j, which is to denote the i_th assembly point in the j_th time slot for that AP. In this example, all AP's were duplicated 3 times so each has entry i.1, i.2, and i.3.

We also keep track of how many times the sinks were touched, so eventually we can calibrate our algorithm to touch each sink the desired amount of times.



Example

We now return to the original graph and demonstrate two of the routes shown in the previous slide on this graph. Watch as I walk through them. Note that the rest could be included very easily but I chose to show only two to reduce screen clutter.