RESULTS

Our project was incredibly eye opening when it came to connecting traversal algorithms with geometry and airline routes. The goal of our project was to find and return a valid airline route that traced a polygon defined by the user. This route, however, would not only resemble the geometric preferences disclosed by the user, but would also be the shortest possible route. Considering the amount of possible airports and routes worldwide, we decided that the best way to tackle this problem was to implement a few traversal algorithms, specifically DFS and the Landmark Problem. By utilizing these two algorithms, we were able to successfully determine the shortest worldwide flight route that matched the polygon defined by the user.

In order to find valid routes, we used a clock-based recursive approach. Every vertex object had a map that mapped a sector to a set of strings, specifically airport codes. The sectors are integer representations of a set of degrees, which was in our case set by our DOF (degree of freedom) variable to contain 5 degrees per sector. This map was precomputed during the constructor logic within MapRoute. This prevents repeated computations when the Depth-First Search executes (memoization). In order to perform DFS, we created a function that alphabetically iterates throughout the vertices_map vertex by vertex in order to call a helper function that recursively looks for inner angles that satisfies the user defined angle vector. Within this helper function, we initially look for two outbound edges that create the first user-defined angle. Then we perform a recursive call, using one of the destinations as the primary argument, as well as the origin and other destination as secondary parameters. This recursion continues until all the necessary angles are found with valid destinations that eventually meet back up with the origin, thereby closing the polygon. Potential vertices are stored into a stack, and when the path finally wraps around to the origin, the stack will be converted into a true solution (saved in solutions vector). This search will continue until every vertex is exhausted and all potential solutions are found.

Once a valid route was established with the correct angles, and the correct amount of edges and vertices, we then discovered that this solution can further be improved. What we didn't realize before was that some of these airports have neighboring airports that, if utilized as an alternative stop, could decrease the total distance of the route. The way we approached this idea was by implementing the Landmark Problem. The Landmark Problem is defined as the shortest path from "a" to "b" through "c". We considered "a" and "b" as every even-indexed airport on the route, and "c" as a region that included every odd-indexed airport between "a" and "b", as well as any neighboring airports. For every three airports, we would see if the second airport had any neighbors that had shorter routes between the first and third airports. If this was the case, then the original path would be modified so the alternative airport would be included instead of the original airport. An example of this is illustrated below in *Figure 1*. The algorithm would not make a change, however, if there was no valid route between the possible alternative airport, and the next airport on the route. We also discovered that we needed to set an acceptable angle range for possible alternative airports since different airport locations changed the geometric orientation of the route. After implementing this algorithm, we found that certain routes resulted in a further shortened in distance!

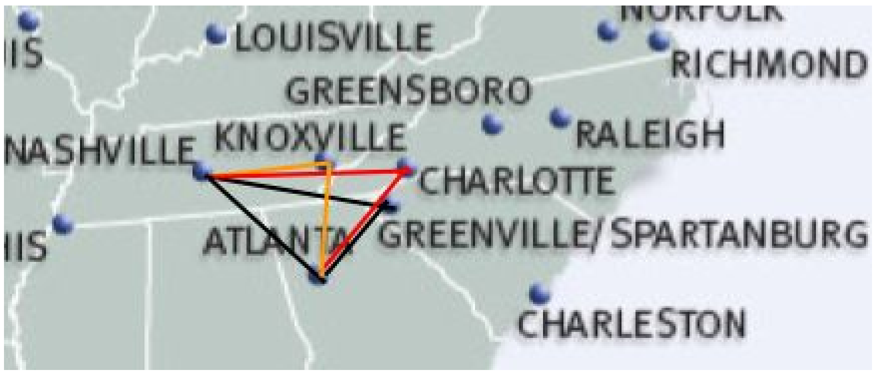

Figure 1: Original route is highlighted in black (Nashville -> Greenville -> Atlanta). Landmark Problem checks general region to see if the alternative route has a smaller distance (Knoxville or Charlotte)

Throughout this project, we discovered an effective and unique way to implement both BFS and the Landmark Problem. By utilizing BFS, we could efficiently traverse all of the airports and routes through backtracking. The Landmark Problem ensured the optimality of our solutions and successfully addressed our goal of returning the shortest possible route. Overall, it was very interesting to see the intersection of map traversals with geometry.