HSR Algorithm

Vinit Patel

December 10, 2023

1 The Problem

The United States' network of passenger rail service lags far behind those of other industrialized nations. Indeed, the only high-speed rail (HSR) line currently in operation is the Acela Express, operated by Amtrak on the Northeast Corridor. The remaining Amtrak service that exists is often slow, unreliable, and expensive. These factors preclude rail transportation from being a viable alternative to driving or flying between cities in the US, even on medium-distance and regional routes.

In this project, we wanted to explore what an optimal national HSR network could look like for the US, given various performance metrics and budgetary constraints. In particular, we are interested in the answer to two questions:

1. Given a fixed construction budget, what is the optimal selection of intercity rail corridors to build? **(Problem 1)**
2. Given a goal of improving national metrics (emissions, population served by rail, etc.) by a specified amount, what is the least-cost investment necessary to achieve this, and which arrangement of intercity rail corridors would satisfy this? **(Problem 2)**

We may choose to define optimality in various ways. We aimed to define an optimal HSR rail network as one that satisfies the following conditions:

1. Maximizes emissions reductions compared to flying and/or driving.
2. Minimizes the cost of construction.

Our project aims to find an appropriate balance between these demands, and also explore how prioritizing them differently may affect the solutions we find.

2 Technical Statement

Our technical problem revolves around optimizing the layout of a high-speed rail (HSR) network in the United States. To address this, we propose a mathematical optimization model that considers various factors such as construction costs, emissions reduction, and travel time savings.

2.1 Problem Formulation

The problem can be mathematically formulated as follows:
Given:

- A set of potential intercity rail corridors.
- Construction budget constraints.
- Goals for population served, emissions reduction, and travel time savings.

Find:

- The optimal selection of intercity rail corridors that maximizes population served, minimizes emissions, and maximizes travel time savings, while adhering to the budget constraints.

3 Datasets and Exploration

Our project requires access to **numerous** quality datasets. Therefore, our first step was to research the data that was available for free. We wanted to explore a state space of cities in the US, and therefore had to find datasets that were formatted according to city codes, airport codes, etc.

The United States Census Bureau publishes numerous relevant datasets, including one that lists the population estimates for every city in the nation. Unfortunately, the data they provide is not the easiest to understand due to poor formatting and a lack of cohesion. Instead, we found this dataset published by ESRI (a market leader in GIS) that includes all the basic columns we need: city name, class (town, city, etc.) state, population, and **geographic coordinates**.

This data was very helpful and also provided a shapefile that let us actually graph out and visualize the findings from our search problem using geopandas, which proved integral to actually visualizing our data due to the nature and scale of the problem we were trying to formulate.

The more difficult-to-find data concerns flight routes, flight schedules, and airport-to-city mappings. Fortunately, we found a few companies that offer APIs specifically designed to provide this kind of data. They are not free, but we reached out to Sergey from AirLabs, who graciously offered us a short-term student license for this project with 50,000 API calls. This is very exciting since the API provided the following relevant functions for us:

- Searching for airport schedules by airport code
- Searching for nearby airports by latlong

- Finding the aircraft configuration by flight (useful for estimating passenger volume)

From Transtats, we got this dataset which we derived a few columns from to fit our needs:
Quick note, each row represents a "market", which is a single one-way intercity route for a single fare class for a single airline. Also IATA codes are simply the codes for specific airports, e.g. LGA for LaGuardia Airport in NY.

- **Origin** = IATA airport code of origin
- **Dest** = IATA airport code of destination
- **NonStopMiles** = flight distance (we used km for uniformity)
- **Passengers** = 10% of the passengers who flew that market during the given quarter (which we then multiplied by 10)

We grouped this data by Origin and Dest, then summed the passengers for each group - this computes one-way passenger counts. To get 2-way passenger counts, we combined rows where (Origin, Dest) = (Dest, Origin).

From GitHub, we found detailed airport data. We formatted the csv as such:

- **iata**: 3-letter Location Code (7,588 entries) or an empty string
- **city**: city's name in latin script (ideally the local language)
- **subd**: subdivision (state, province, etc.)

This let us pair airports to cities, which in turn lets us assign geographic points to the origins and destinations we are trying to plot using the ESRI shapefiles mentioned earlier.

Beyond these, we still needed data to get an idea of things such as time savings compared to taking a flight, emissions saved, and the construction costs. Unfortunately, we weren't able to find a good dataset for estimating the flight times on each segment, so we assumed average flight speeds by distance:

- dist0: <300 km = 200 km/h
- dist1: 300-900km = 400 km/h
- dist2: 900-2000km = 600 km/h
- dist3: 2000-4000km = 700 km/h
- dist4: >4000km = 800 km/h

We considered this as realistic as we could be since it reflects that shorter flights will not reach top speeds due to lower cruising altitudes.

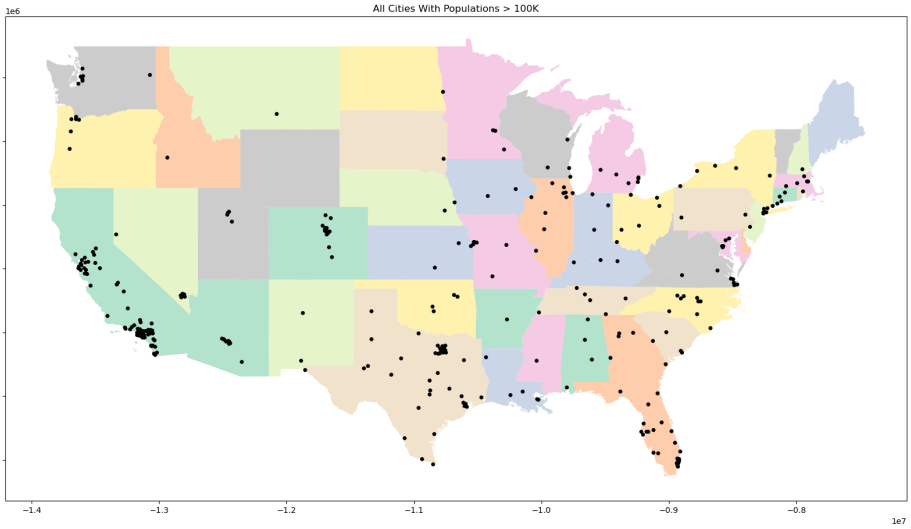
Luckily however, we were able to find relevant data for emissions and construction costs. From the emissions estimates, we used Scheduled flight (Economy) vs Electric train (Europe) to get an idea of the comparative emissions from both forms of travel. In regard to construction cost, we found these metrics:

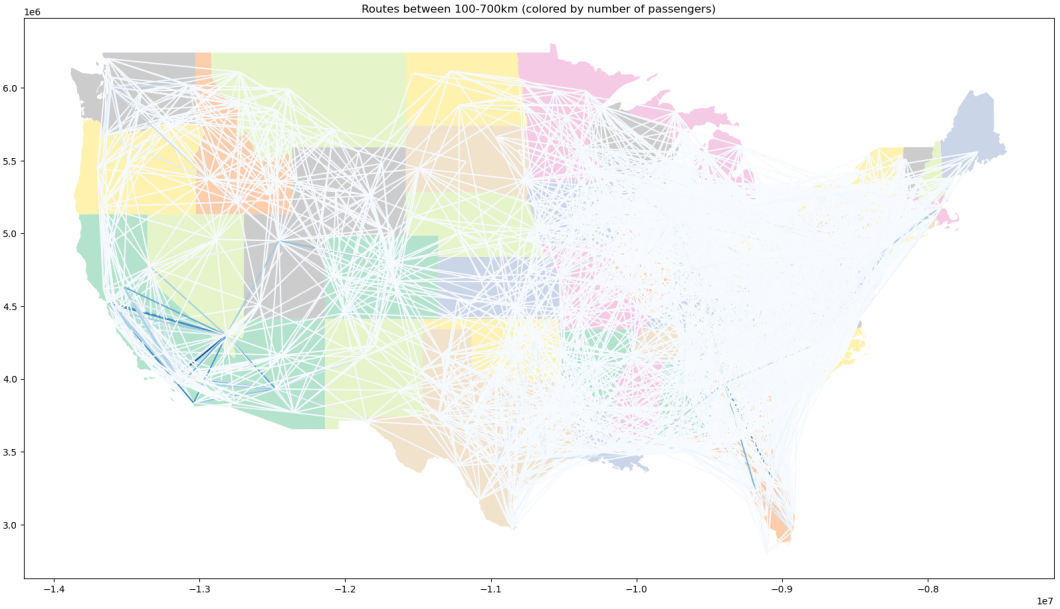
High-speed double track on new stone rail road stone bed high cost/km = 1650000
Install a Centralized traffic control system double track – high cost/km = 257825

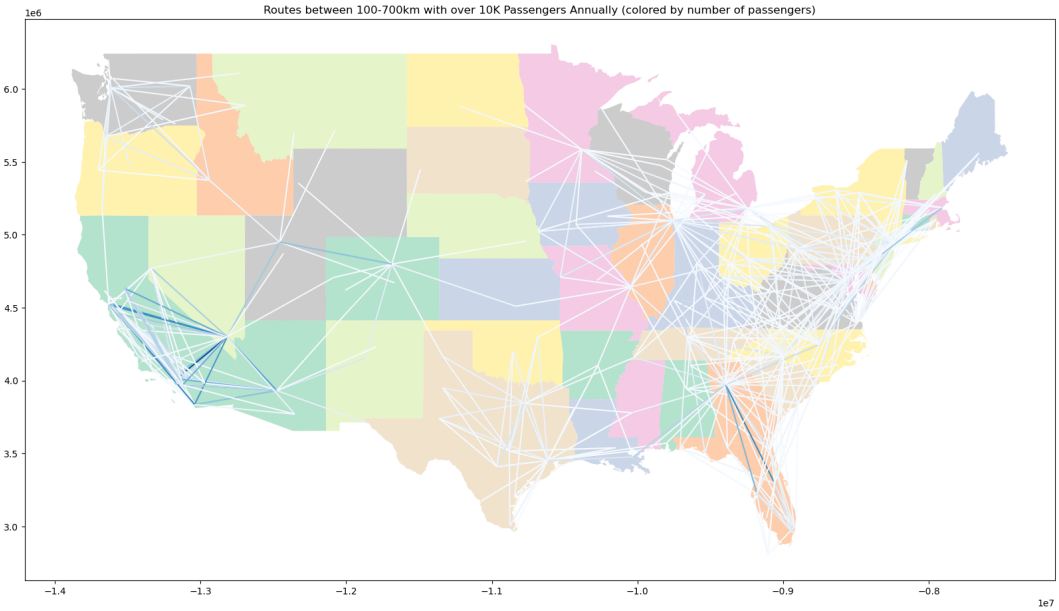
and determined our cost per km to be **165000 + 257825**.

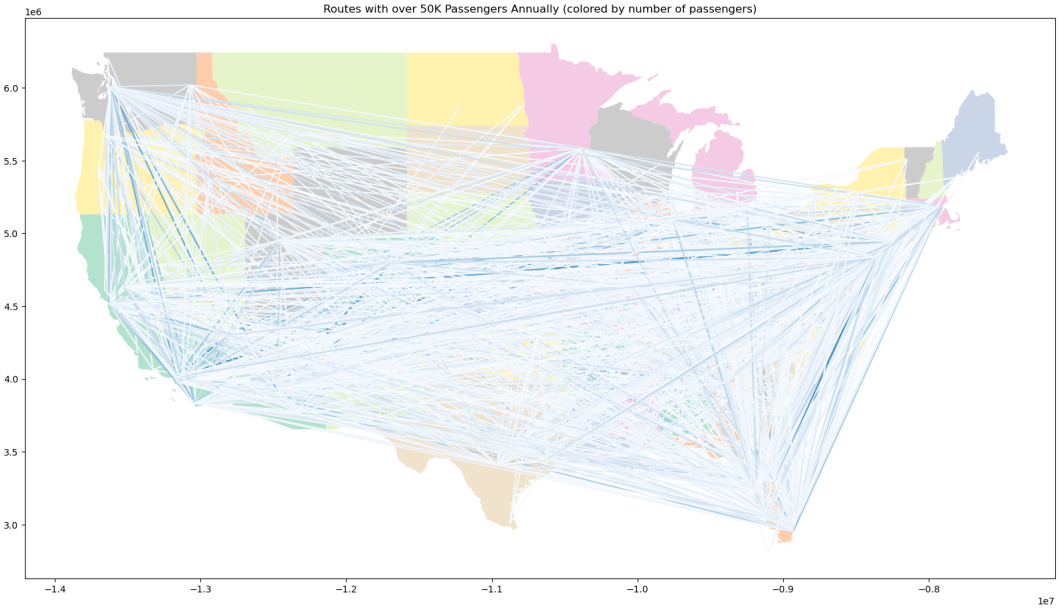
Originally we had intended to use APIs for flight emissions, schedules, and passenger volume our project, but we chose not to due a few reasons. Firstly, we found that estimates of emissions were sufficient for our purposes, the DB1B survey from earlier provided enough volume data, and it would have been slower to use APIs, not to mention we didn't have enough calls anyway. For these reasons, we ultimately proceeded without APIs for those parts.

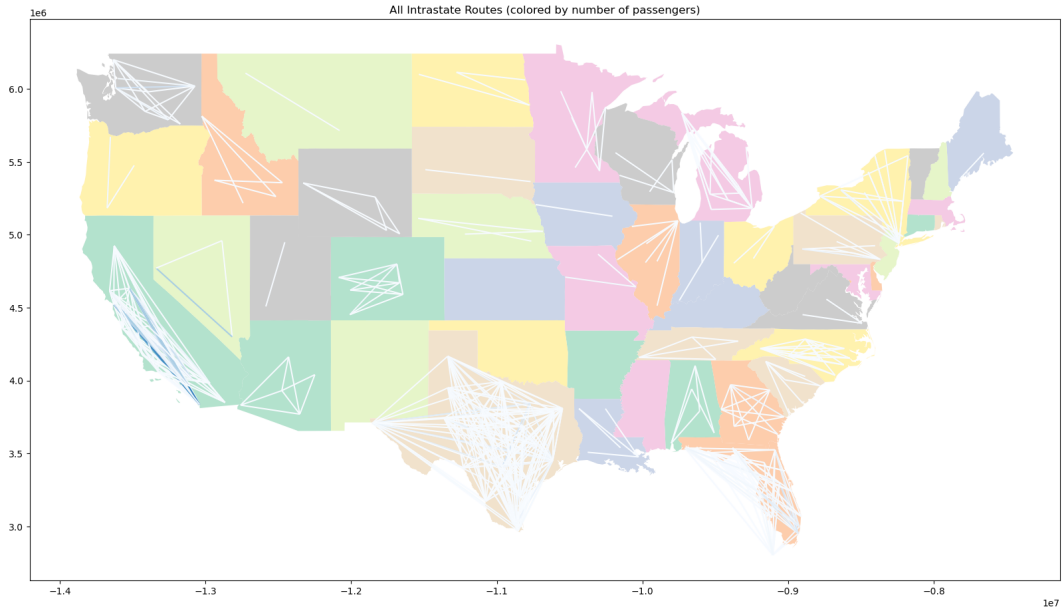
However, a large part of our project was just how much we **explored** the data we had. Here are a few of the plots we considered the most relevant albeit messy.


All Cities With Populations > 100K
1e6
6.0
5.5
5.0
4.5
4.0
3.5
3.0
−1.4
−1.3
−1.2
−1.1
−1.0
−0.9
−0.8
1e7


Routes between 100-700km (colored by number of passengers)
1e6
6.0
5.5
5.0
4.5
4.0
3.5
3.0
−1.4
−1.3
−1.2
−1.1
−1.0
−0.9
−0.8
1e7


Routes between 100-700km with over 10K Passengers Annually (colored by number of passengers)
1e6
6.0
5.5
5.0
4.5
4.0
3.5
3.0
−1.4
−1.3
−1.2
−1.1
−1.0
−0.9
−0.8
1e7


Routes with over 50K Passengers Annually (colored by number of passengers)
1e6
6.0
5.5
5.0
4.5
4.0
3.5
3.0
−1.4
−1.3
−1.2
−1.1
−1.0
−0.9
−0.8
1e7



4 Methods/Algorithms

Our algorithmic approach is based on Uniform-Cost Search (UCS), a variant of Dijkstra's algorithm. In the absence of a heuristic function, our exploration of potential intercity rail corridors relies solely on the actual costs associated with each corridor.

The algorithm systematically expands the search space, prioritizing nodes based on the various factors from the start node. This ensures a comprehensive exploration of the solution space, considering population density, emissions reduction, and travel time savings without introducing heuristic estimates.

While the current implementation lacks the heuristic guidance present in A* search, it allows us to establish a baseline optimization framework. We can later explore the integration of heuristics to potentially enhance the efficiency and solution quality of our algorithm.

4.1 Problem 1 Inputs

The inputs to our algorithmic approach include data on:

- Potential intercity rail corridors.
- Construction costs for each corridor.
- Population of cities in the contiguous US.
- Emissions reduction potential for each corridor.

- Travel time savings compared to flying.
- A budget.

4.2 Problem 1 Outputs

The outputs of our algorithm include:

- The optimal selection of intercity rail corridors.
- A map of the US with the optimal rail corridors drawn on it (using shapefiles and geopandas)

4.3 Problem 2 Inputs

The inputs to our algorithmic approach include data on:

- A threshold between 0.0 and 1.0 that represents how much of the original emissions we want remaining.
- Potential intercity rail corridors.
- Construction costs for each corridor.
- Population of cities in the contiguous US.
- Emissions reduction potential for each corridor.
- Travel time savings compared to flying.

4.4 Problem 2 Outputs

The outputs of our algorithm include:

- The optimal selection of intercity rail corridors.
- A map of the US with the optimal rail corridors drawn on it (using shapefiles and geopandas)

We formulated our search space as any possible arrangement of intercity rail corridors. With n cities, this gives $\binom{n}{2}$ possible intercity corridors. Our search space is therefore the powerset of this, which has a size of $2^{\binom{n}{2}}$. This makes our state space very large, since even with just 10 cities, it becomes 2^{45}. Since there are around n=33 cities in the Us with populations over 500k, and around n=316 with populations over 100k, attempting to search the state space is impossible without powerful heuristics or ways to cut down the space even further. We did not have the time to implement a good heuristic, which led us to limit the problem to very few cities to produce a result.

The important decisions we made were that we chose to represent our state space as a dataframe where each row is a rail segment, and that each rail segment is bidirectional, i.e. connecting NY to LA means LA to NY cannot be another rail segment.

Originally our cost of a state was determined by summing the benefit of emissions saved and passengers served along with the negative benefit of the cost of construction for each rail segment in our state. The original method to get emissions saved was also by summing the emissions saved by each individual rail segment, which proved naive since there was no way to consider the possibility of transfers, i.e. being able to traverse from one city to another even if there isn't a direct rail segment between the two.

We defined a cost metric algorithm that is used in both the problem formulations we have. In the context of the algorithm, we consider a state as a set of intercity rail segments, produce a graph with the cities as vertices and segments as the edges between them (edge weights being distances), and then traverse that graph to find the shortest paths between every pair of **connected** vertices. Taking that shortest path, we can then calculate its cost metrics.

This algorithm returns the following metrics:

- delta_travel_time

The cost metrics for a state are the sum of cost metrics over all the paths in the graph of the state.
For a single path, the algorithm calculates **v_p**, which is the proportion of flight passengers who will take this path instead of flying.

- If t ¡ 1.0, v_p = 1.0
- If t is between 1.0 and 2.25, v_p = 0.9
- If t is between

Using v_p we can then derive a few other metrics. We can get the number of passengers who will be taking the HSR, **hsr_passengers**. We can then derive the new number of flight passengers by doing total_passengers - hsr_passengers to get **flight_passengers**. For clarification, total passengers refers to all the passengers who currently fly the routes being observed, flight passengers refers to passengers who will still choose to fly once rail segments are built.
We can then further calculate **new_co2**, which is the emissions (in grams) on this route given the number of people who have chosen to either fly or take the train. Effectively, new_co2 = (co2_pkm_flight * flight_passengers * flight_km) + (co2_pkm_hsr * hsr_passengers * hsr_km). On the other hand, **old_co2** is the measure of emissions from before any rail segments were constructed.

From a previously mentioned dataset where we compared emissions from a plane to those of a train, we determined **co2_pkm_flight** = 133 and **co2_pkm_hsr** = 24. We take **delta_co2** = new_co2 - co2_g_flight to be the resulting amount of emissions once we construct the rail segments.

Once we have cost metrics for each path p in a state, we sum them up column-wise to produce scores for the entire state. **THIS** is the table of cost metrics for the state, from which we use the variables we've just defined in the cost functions and goal states of our aforementioned Problem 1 and Problem 2.

In Problem 1 this cost algorithm provides metrics about a given state that are then used by the cost function to evaluate the overall cost of the state. Problem 1 chooses to equate 1000 kg of co2 to 1 passenger, and therefore the cost function takes hsr_passengers and new_co2, and determines that
cost = (new_co2 / 1e6) - hsr_passengers.

The goal state of this problem is determined entirely by the budget constraint and is met when there is no longer a rail segment that can be constructed within the budget. Therefore, the goal test checks if the sum of the construction cost of all rail segments is >= our budget. This is flawed because the first state found that satisfies the budget is returned, even if it's not optimal.

In Problem 2, the cost function is simply the total construction cost of all the rail segments.

The cost metrics algorithm comes into play in the goal test. The metrics let us determine if a threshold we give is met, i.e. a threshold of 0.5 means we want to cut emissions in half, which would then be verified with metrics from the cost algorithm above. The metrics we use towards this end are new_co2 and old_co2, and our goal test checks if (new_co2 / old_co2 <= threshold.

5 Hypotheses/Results

When we originally formulated these problems, our intention was to produce a rail network that prioritized/favored medium distance routes with many passengers. We'd hoped for some balance between passengers served, construction costs, etc.

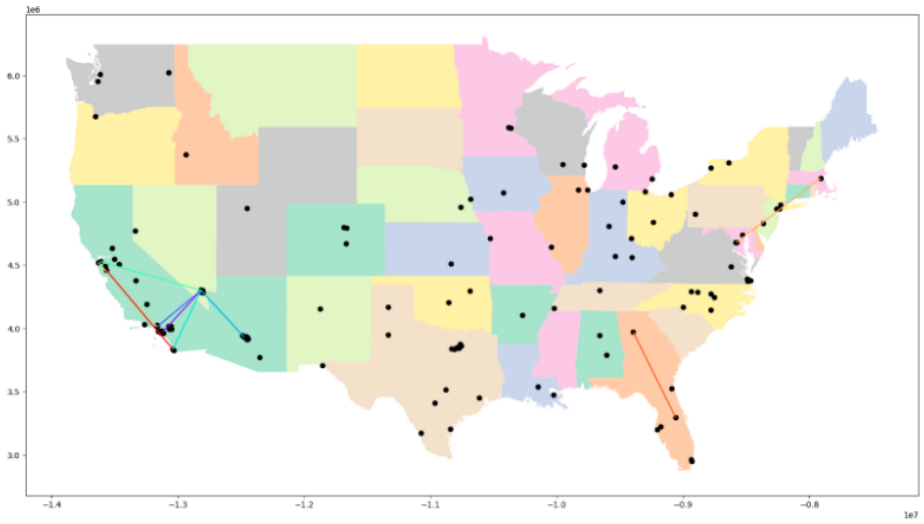
However, we recognized difficulties in finding a good way to assign weights to the costs, and therefore we hypothesized that the results would be heavily skewed toward certain costs, i.e. prioritizing routes that served a large population, saved a lot of emissions, etc.

For problem 1, we did not manage to get any results - the state space was simply far too large. This proved an issue for Problem 2 as well, where in order

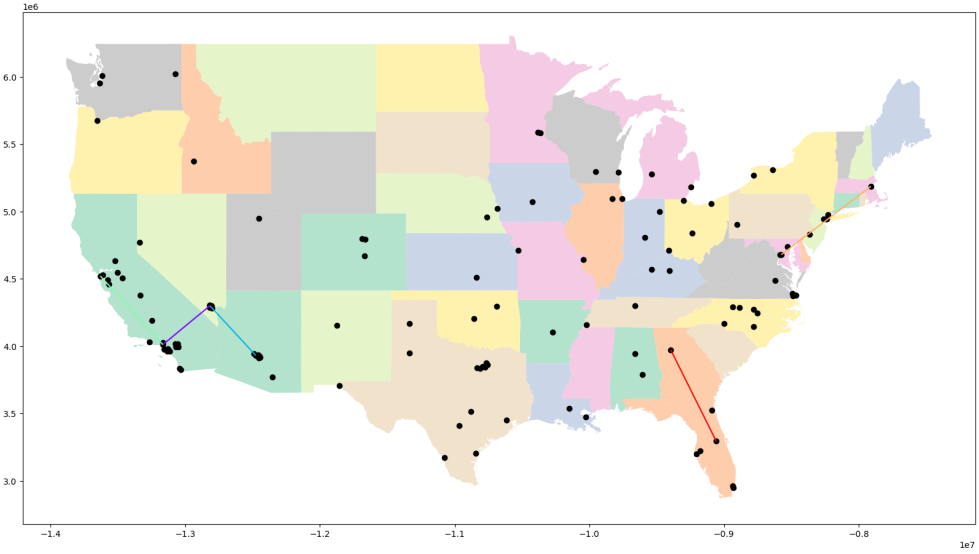
to get results, we needed to shrink the state space to an extent since even a size of n=10 cities gives us a search size of 2^{45}.

We did however get results for small versions of Problem 2 with varying thresholds. We set constraints so that we only considered the top 12 routes where the NonStopKm (flight distance) is between 100-700km, and the annual passenger volume is over 10k. They aren't useful, but they demonstrate what a solution would look like, and that the algorithm is working correctly for the most part. Our solution format provides a list of segments, and we can plot them using the shapefiles from our datasets as intended. Here are some images that illustrate this:

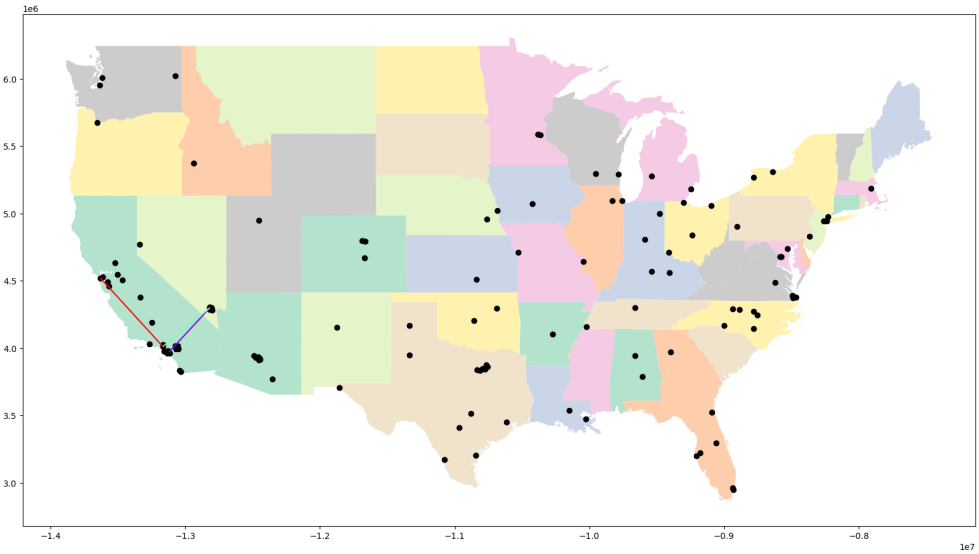
Threshold: 0.25



Threshold: 0.50



Threshold: 0.75



Keep in mind that our thresholds represent how much we want to reduce emissions by. A threshold of 0.25 implies we want a goal state where emissions are reduced by 75%.

There is also a file called solution.csv that will be in the "out" folder of our code which contains the segments being constructed in another scenario of Problem 2 with a threshold of 0.25, as well as the various metrics regarding emissions saved, passengers served, and so on as we'd hoped.

6 Analysis

While our results are not a definitive answer to the most ideal rail network, we do have a functional algorithm to a certain extent, and metrics to support our logic in determining the involved costs.

There are a few scenarios where the algorithm will not finish, but that is not because of the cost function being intensive, but once again because of the sheer size of the search space we are handling.

In terms of observations of our results, we noticed that CA is the highest city in terms of emissions, which makes it clear we need to be more careful in weighing our costs as we'd suspected initially. The varying units in our metrics, from emissions to populations to time saved caused quite a bit of difficulty in determining how we could weigh our costs properly and proved to be a major influence on the results we did receive.

7 Discussion/Future Directions

We had a few major difficulties at various parts of our project. At the very beginning, we struggled in finding good passenger volume data, and eventually settled on DB1B as our source. Later on, once we'd found the majority of the data we wanted to work with, we still had to find a good way to map cities and states to IATA codes. This caused an issue where airports were being considered as cities. For example, if LGA, aka LaGuardia airport in NY was connected to an airport in LA, NY as a city wasn't considered to be connected to LA. Rather, other airports in NY, such as JFK, could still be connected to an airport in LA and have them be considered a unique rail segment regardless of the city they were in. This meant cities with multiple airports could be counted multiple times, so the error was that our airport data was not aggregated. This skewed our results because JFK-LAX and LGA-LAX are both huge routes, but we would want the model to aggregate the passenger data from NYC. This issue was not unique to NYC, as shown below:

80	US	CHI	Chicago	MDW	Midway International
81	US	CHI	Chicago	ORD	O'Hare International
82	US	DFW	Dallas	DAL	Love Field
83	US	DFW	Dallas	DFW	Dallas/Ft Worth Intl
84	US	HOU	Houston	HOU	William P Hobby Airport
85	US	HOU	Houston	IAH	George Bush Intercontinental
86	US	NYC	New York	JFK	John F Kennedy Intl
87	US	NYC	New York	LGA	LaGuardia
88	US	WAS	Washington	DCA	Ronald Reagan National
89	US	WAS	Washington	IAD	Dulles Intl

There are quite a few cases where a city is listed multiple times since it has more than one airport, and this list doesn't cover all of them either.

Overall though, the biggest challenge was of course the sheer magnitude of the search space we were trying to work within. Considering n as the number of cities we wanted to consider, n=5 gives us a size of 1024, n=6 gives us 32768, n=7 gives us 2097152, and so on. Attempting to search the entire state space proved impossible without a powerful heuristic or a way to cut down the space further. Unfortunately, in our case we did not have enough time to implement a good heuristic, and therefore had to produce results by limiting the problem to very few cities.

Assuming we'd had more time, we would have further cleaned the data to combine airports to cities (aggregate the data), and figure out a powerful heuristic to make the search space more navigable for our purposes.

If we were to advise future students who attempt this project, the major suggestion we'd give would also be to come up with a heuristic, and ensure that the data they choose to work with integrates well so they don't have issues as we did with airport data.