Final Report on Bus Congestion

Contributor:

Name	**Email address**
Chengjun Wu(Lead)	simonwu@bu.edu
Annan Miao	annan@bu.edu
Jing Li	jingli18@bu.edu

1. Summary

With a huge amount of public school students taking school buses in Boston for daily commuting, the BPS bus routes map needs to be improved to reduce disruption to student arrival times and keep students out of dangerous situations. Tackling this problem will not only help students arrive to school on time but also lower the risk of traffic accidents. By leveraging historical traffic data, our team aims to build a model to fit into this route system and utilize it to evaluate the variance of switching from one route to another.

For this project, our goal is to optimize the BPS bus routes in order to decrease the amount of time that students are on buses and therefore decrease the number of students arriving late to school. We will analyze the dataset of MBTA and BPS to analyze the traffic pattern within the greater Boston area and apply the result to the BPS bus routes map to optimize it.

2. Data analyze

2.1 BPS Sensor Data

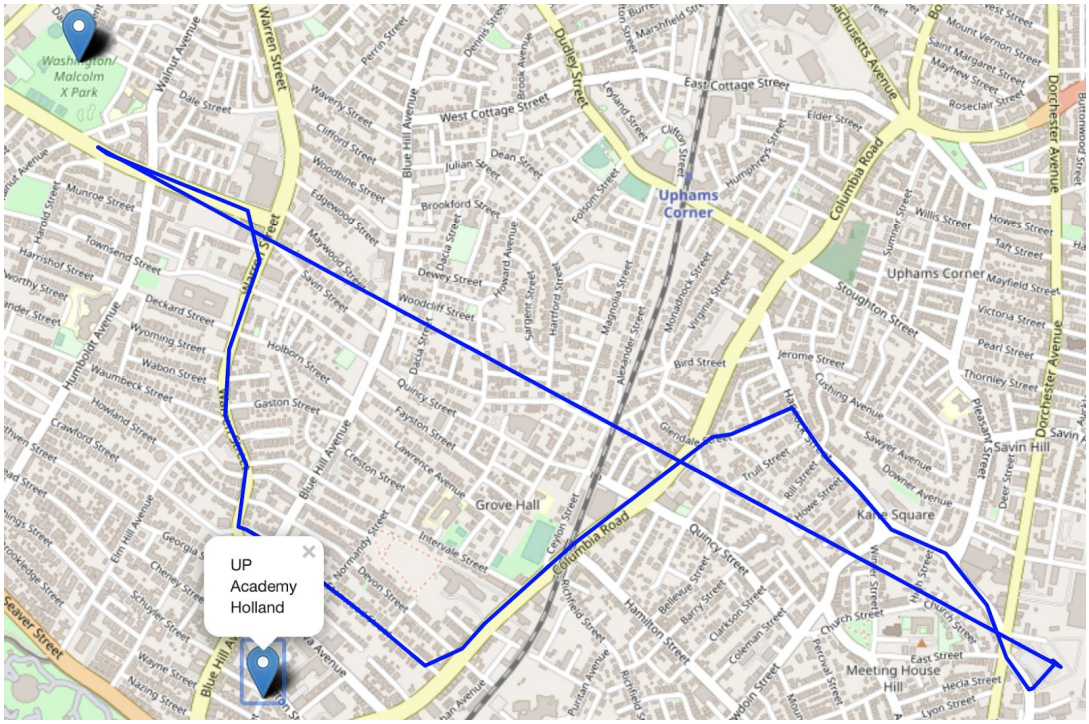
This data set was given in a CSV file, which has the data frame shows below:

Logtime	**Latitude**	**Longitude**	**Heading**	**Speed**	**VendorhardwareID**
2018-02-07 00:23:48	42.33514	-71.0803	0	1	MS122

This dataset has the information that we need about school buses. It has more than 60,000,000 rows so that we use chunks to deal with it. The core information that we want for each row is the timestamp and location for a vehicle, which is specified by the machine ID of that vehicle. So we filtered this dataset and got this data frame:

Logtime	Latitude	Longitude	VendorhardwareID
2018-02-07 00:23:48	42.33514	-71.0803	MS122

And then we grouped this dataset by the VendorhardwareID and sort them by timestamp. In that way, we are able to identify the route of a particular vehicle through iterate through the sorted data points. The figure below shows a sample route of the BPS bus.



The map shows that a vehicle will run different routes. We need to do more analysis to get the routes map of the BPS.

2.2 MBTA Historical Traffic Data

Data Source: https://api-v3.mbta.com/docs/swagger/index.html

As we've been checking out from any web portals online, there's no such a web portal that can provide us with the MBTA historical traffic data in the form of a real-time

location. Thus, we decide to scrape these real-time location data for every vehicle(bus) in the MBTA system for two weeks to construct a new traffic data set for training our model.
If we keep track of every bus by looking at its location every 3 minutes and save that location as well as some other useful attributes as an entry, the new traffic data set will include quite a lot of entries even though it's just 14 days in length. We think that the amount of data is informative enough to give us a traffic pattern in Boston.

Here's the form of each entry in the output file and the descriptions for each attribute:

id	The identifier for different buses
current_status	whether this bus is in service or not
direction_id	The direction is bound to. Inbound or outbound.
latitude	real-time location
longitude	real-time location
updated_at	The timestamp
day	The day
time_slot	The hour of the timestamp that used to indicate what time interval it lies in. E.g The time_slot of 17:00:01 will be 17.

From MBTA APIs, we find there are 168 bus routes within the MBTA system where almost all routes here may have impacts on the routes of the Boston public school bus. In that case, we decide to keep them all to build a model rather than filtering any out.

Our script has been deployed to auto-fetch the real-time information of all vehicles running on the 168 routes described above every 3 minutes from MBTA APIs. On average, there will be over 5000 entries written in the output file for an hour.

Each entry in the output file represents a piece of real-time information for a certain bus at a certain time. The first six attributes are included in the response from MBTA APIs while the latter two are variables defined and computed on our own for further researching.

2.3 Google Maps API

After the analysis of the current BPS routes, we need to find some alternative routes and compare them to give optimize routes recommendations. We align the Google Maps API with our framework to achieve the goal.

a. Googlemaps.directions
   We can get the alternative routes with this. To do that we give the API the start and end points together with 1) direction mode (drive, bus, train, bicycle…), 2) departure time, and set the 'Alternative' to be true.

b. Googlemaps.geocode
   We take advantage of geocoding API to convert geographic coordinates(like longitude and latitude) to a human-readable address of a street name. By doing so, we can convert the BPS sensor data points into a list of the street they lie in and then merge adjacent items with the same street name to get a route based on a list of street names it passed through.

3. Questions to be answered

There are several strategic questions in the project description needed to be answered to better understand the bus congestion problem.

1. Which streets have the greatest impact on bus schedules neighborhoods have the highest rate of traffic congestion?
2. Which route has the lowest time to distance ratio?
3. Which route has the highest idling and hence emissions?
4. Can we identify problem spots for bus routing? Do these problem spots extend to traffic in general?
5. What are the characteristic attributes of the problem area and what is the impact of time-of-day? Are there any creative solutions beyond rerouting?

Unfortunately, we are not able to tell answers to all the questions above. More specifically, the strategy we are going to present won't be able to advise any answers toward questions 2, 4, 5 above. The following shows why these three will be skipped.

- For question 2:
  Although this problem would suggest the route condition in general, it actually can't provide more insights into which spots a certain route is likely to get in congested and then to be optimized. Our model though can't tell the general case but is more helpful in taking a close look at the routes and advice to optimize it.

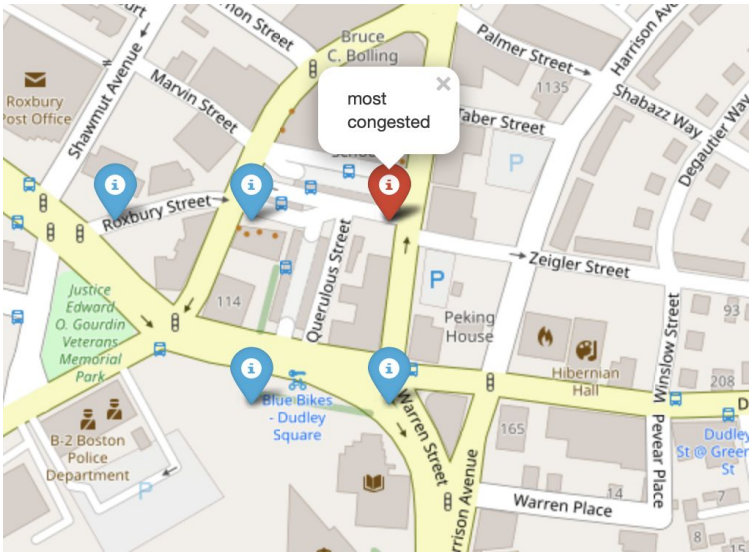
- For question 4:
  We are not going to solve this problem simply because we're not able to find enough relevant data sources for traffic incidents. Although there are some at MassDOT data, those data are not fine-grained and not sufficient to build a convincing model. To address this problem, we need historical traffic incident(crash) data within the Greater Boston Area rather than Massachusetts with labeling the geographical location of the traffic crash.

- For question 5:
  Although we are not going to research problem spots for this project, we will be discussing traffic patterns, most of which are congested conditions within the Boston area, in terms of the time of day. Different time periods would lead to different traffic conditions, bus routes should, therefore, be optimized in different ways accordingly.

However, we are able to reply to question 1 and 3 clearly based on our method.

Question 1

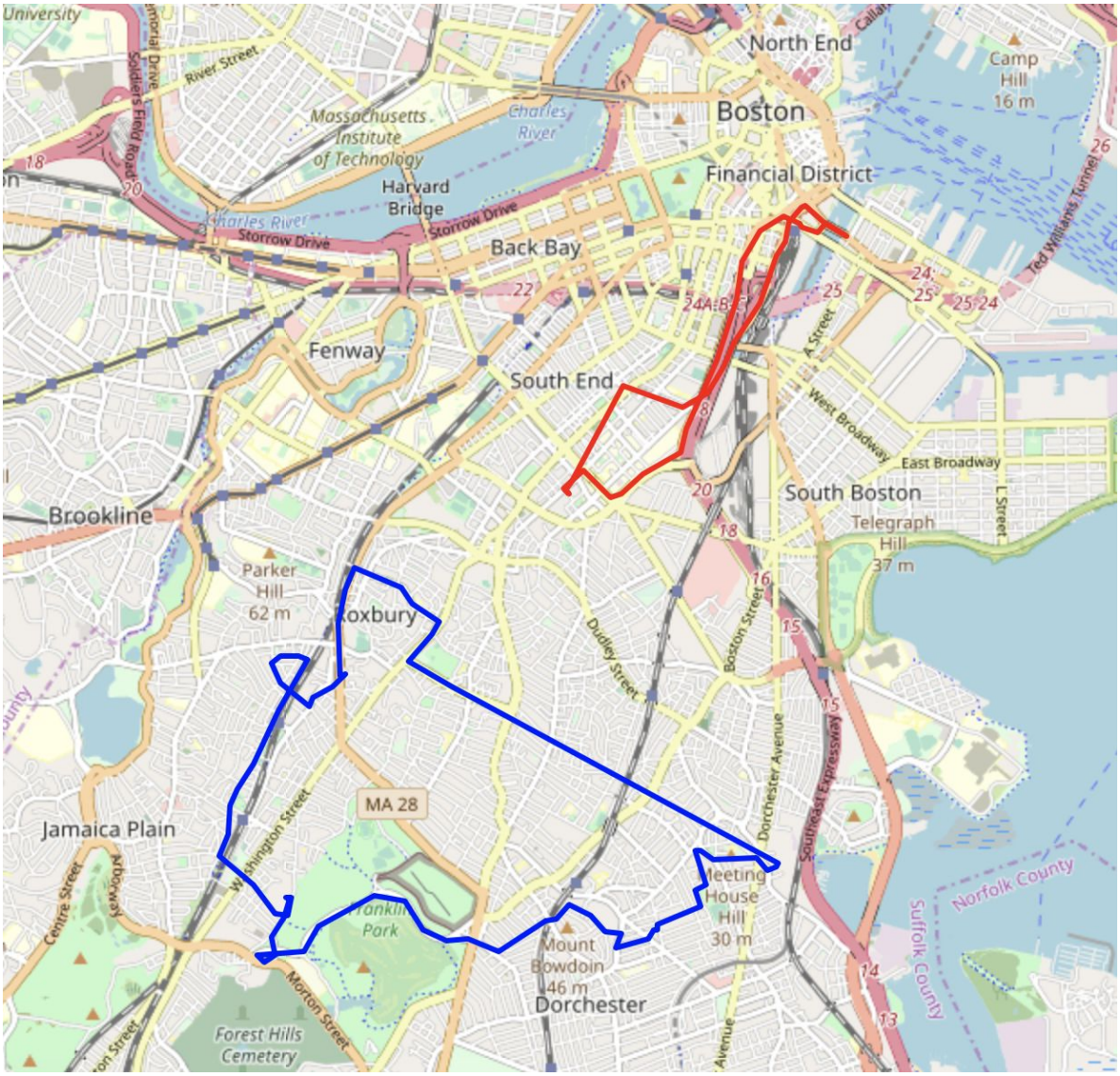
Basically, we separate the Boston area into a bunch of grids with around 100 feet in width and 100 feet in length. By estimating congested conditions for each grid based on our approach we quantify the congested condition with a value to indicate how likely it will be congested. After all, we iterate through all grids to find the one that has the highest value as shown in the screenshot below. From our analysis, the connection between Roxbury st and Querulous st has the highest rate of traffic congestion.



Question 3

We are almost using the same grid notion above for the problem as well. Besides computing congested value for each grid, we also score a BPS route by adding up all grid it passes by to compare different routes.

As indicated through visualization, we find the most idling traffic conditions shown as the blue one in the figure and the red one as the busiest route.



4. Approach

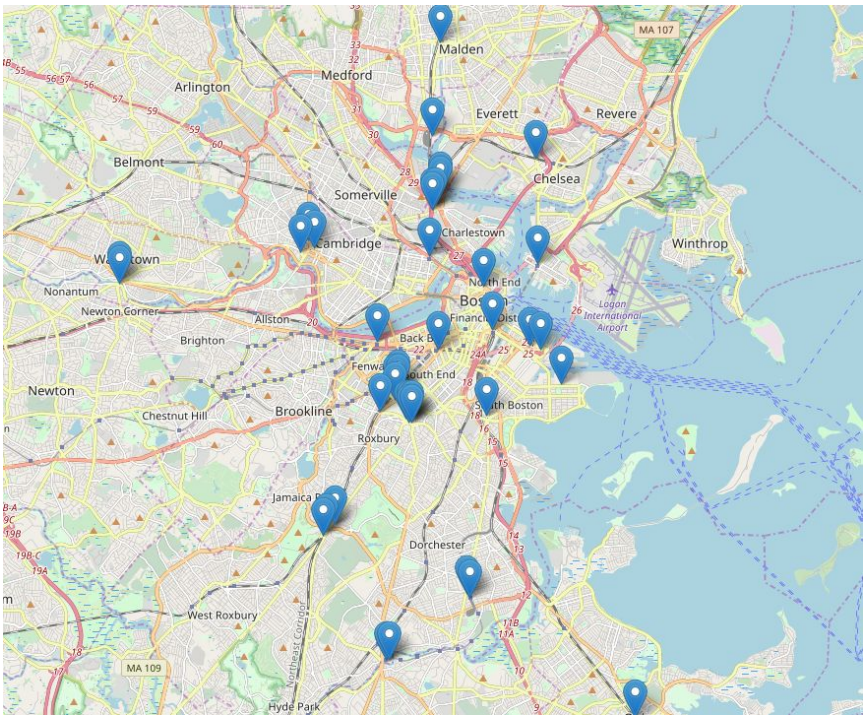
To convert BPS and MBTA into numerical values for analysis, we separate the map of Boston area into grids and then fit the data points with position and timestamp into the grids to find out the traffic pattern, where each grid is 0.001 degree in both latitude and longitude and that is 111m x 111m in measurement. Whether a block is a congestion area is determined by the number of points in the corresponding grid: the more data points in the grid, the more likely the grid corresponds to a congestion area.

We generalize our approach by three main parts as below.

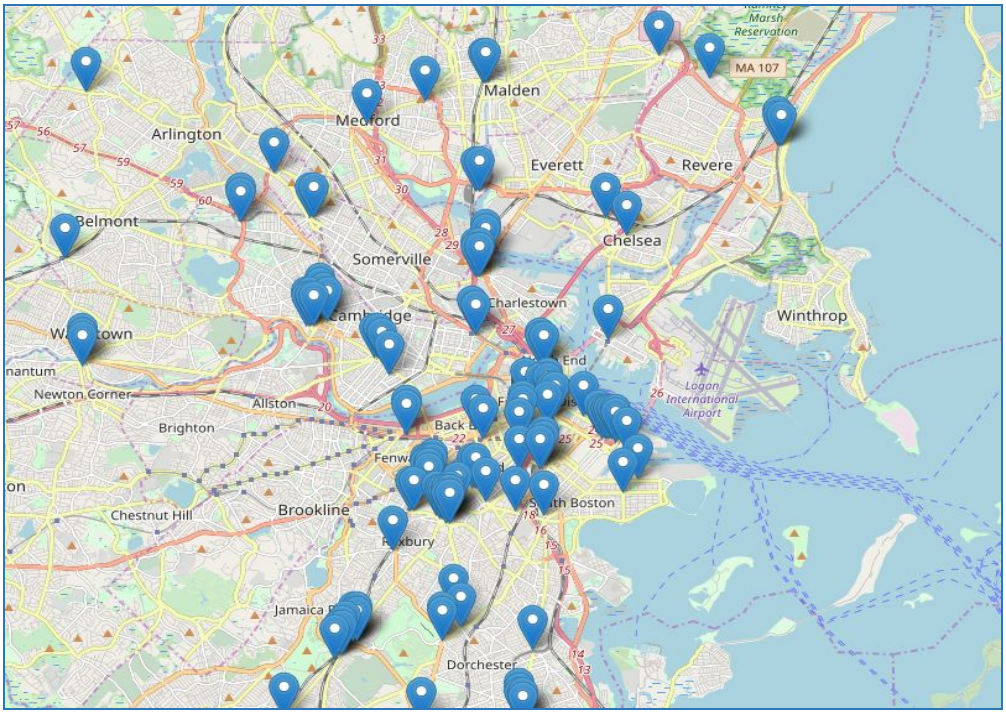
Quantify Congested Condition

Separate Boston area into grids of a block size, and then fit data points of position and time into the grids. Congestion is determined by the number of points in the corresponding grid and a threshold (e.g. top 1% of total grids that contain highest number is considered as congested). Congestion ratio of a grid is defined as the possibility that a grid is congested. Congestion ratio of a grid is calculated by the percentage of time periods that the grid is congested. We can change the time period to make the output more reasonable. For instance, the time periods can be defined as 1 hour or 1 day. Below are the outputs of congestion ratios with 2 different definitions of time period:

1-hour based congestion areas (congestion ratio > 0.3):

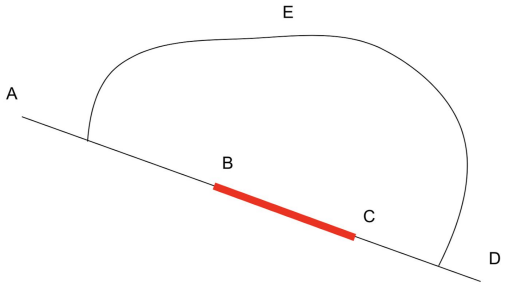


1-day based congestion areas (congestion ratio > 0.3):



Street-wise optimizing strategy

Although we are able to indicate if a certain grid passed through by a route has relatively high congested value or not, it's still not likely to say at what points should we substitute the route. Assume a short part of a long street has pretty high traffic volume but the rest of it does not. It probably doesn't work if we try to replace an alternate route between two exact endpoints of congested places. Intuitively, a driver has to turn to another route in advance to avoid the congestion. We, therefore, come up with a method to address such a problem based on describing a route by a sequence of streets it passed through rather than a bunch of geographical coordinate points.

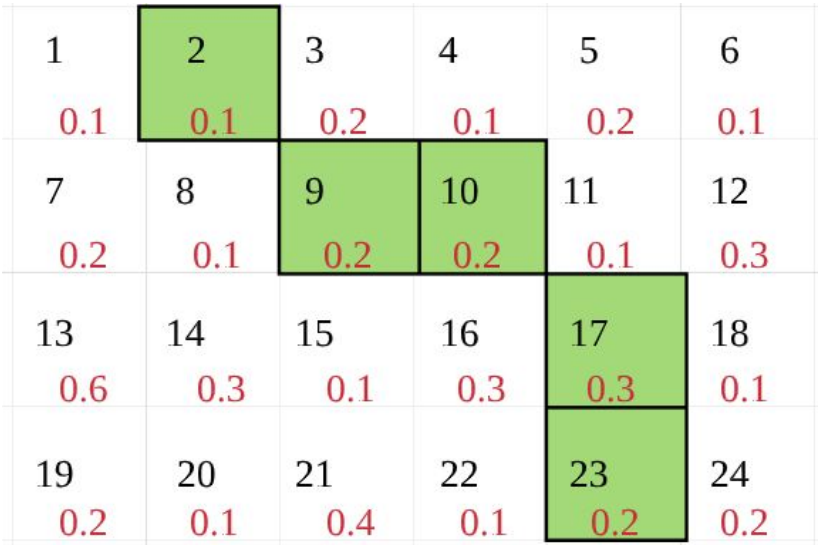


As shown in the figure, the red road as marked as BC is under heavy traffic. However, an optimized route AED can not be found if we use point B and C as endpoints to request an alternative route.

We benefit from this approach because we can then quickly find the starting point and the endpoint of a street if it has been congested at a certain point. As long as we can identify congested places, we are able to avoid that street by getting an alternative from the starting point to the endpoint of the street - places where no congestion meet yet.

Scoring Routes

After the process above we get some key points which will lead to the congestion area. We treat them as the start and end location for googlemaps.directions API and provide the time slot information to get alternative routes between these points. For the time slot information, we do that because our analysis is based on different traffic patterns in different time slots and we could also take advantage of the traffic dataset in Google Maps to get better recommendations. After we get some alternative routes. For each alternative route, we map these points on the region and get the index of the grids that this route goes through. Then we can get the score of this route by computing the mean-congestion-ratio of these grids.



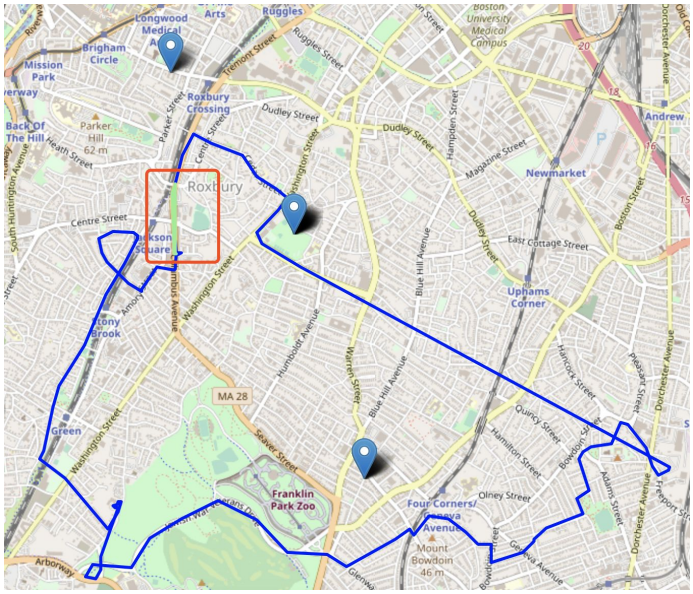
As shown in the figure, these are part of grids in our region.
The black numbers are the index of each grid;
The red numbers are the congestion ratio

If the route goes through these green grids then the score of the route is 0.2 (1.0/5).

5. Findings

As we finish all three steps above, we are able to figure out if any routes can be optimized by computing the score of a BPS route and see if any substitution routes, which are recommended by Google, have a lower score than the original one. However, due to limited computational

resources and a large amount of time for coordinating with Google APIs through the network, we are not able to compute routes of all BPS buses and see if any optimized route exists. Even computing one route for one bus within a day would be expensive in running time. We try to compute several routes of tens of buses but unfortunately can't find an optimized route.



Ideally, any alternative route will be indicated as a green line plot above over its blue origin BPS route. However, from our results, we find the alternative routes, which are advised by Google APIs, always overlap with the original ones in our cases. In other words, even though the route is likely to be in heavy traffic, there are no other routes better than that can work better.