Design and Implementation of A TTE System

First Report

11812804 董　正

11813225 王宇辰

11811305 崔俞崧

Supervisior: 宋轩

Department of Computer Science and Engineering

Apr. 2021

Content

1 Preliminaries

1.1 Review

1.1.1 TTE

Travel Time Estimation (TTE) is one of the most important researching topic in the traffic forecasting field. Estimating the travel time of any path in a city is of great importance to traffic monitoring, route planning, ridesharing, taxi dispatching, etc. On Sep. 2020, DeepMind published a blog named *Traffic prediction with advanced Graph Neural Networks*. This blog briefly described the whole industrial structure of estimated times of arrival (ETAs) techniques applied in Google Map but did not given any detailed implementation or any code. Our work is based on the model structure of TTE proposed in the blog.

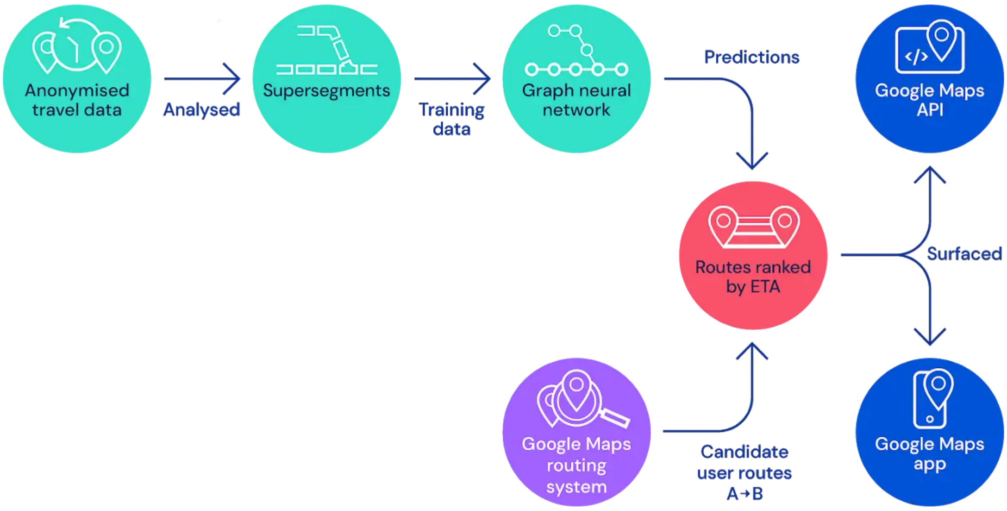


Figure 1: Architecture

1.1.2 Goal

Our ultimate goal (tentative) is to implement the industrial structure and apply it to the open source databases in China, then compare the performance with the state-of-the-art structures and find its application value. This semester, we will implement a TTE system base on the work we done in the last term, combining *Supersegment* and TTI. We will try to work out an interactive application with graphical user interface.

1.2 Introduction

In the last stage, we have processed some open source data. And we got deep in the code of *DeepTTE* and researched on a new concept called **Travel Time Index (TTI)**. In addition, we proposed a new computing process of *Supersegment* and gave a simple demo.

Breifly, we will state our work in this report as

- Code implementation of *Supersegment* by 董正
- Computing TTI by 崔俞崧
- UI Design by 王宇辰

2 Code Implementation of Supersegment

2.1 Dataset

- Data source: Didi Chuxing GAIA Intiative
- Region: Chengdu
- Time: 2018-10-01 to 2018-12-01
- Content:
  - GPS track of taxis with timestamps
  - Coordinate of road and district boundaries
  - TTI and average speed of districts

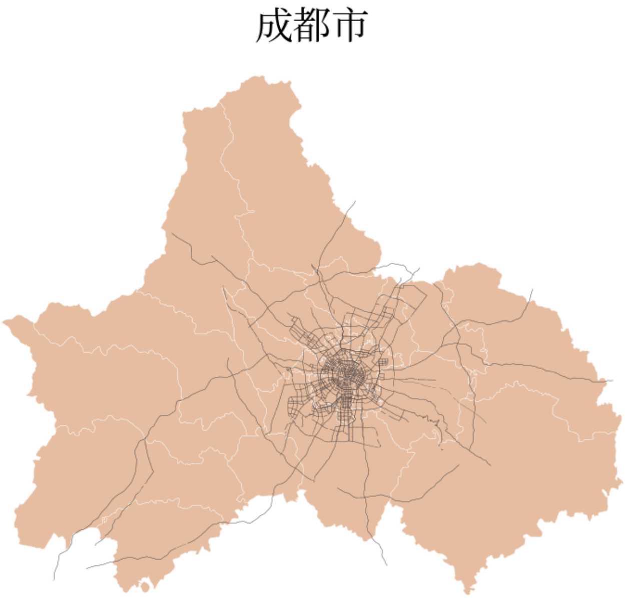


Figure 2: Cheng Du

2.2 Model Design

In this section, I will propose a model to compute Supersegments. At the very first, I need to ensure that I will base on **speed** to do the division. There is a three-step approach of my design:

1. Locate GPS coordinates of taxis into corresponding roads
2. Use the timestamps to calculate average speed and regard it as the instantaneous speed of midpoint
3. Apply clustering algorithm to these midpoints
4. Intersect road and clusters to get segments

First, since we know the boundaries of the roads, we can determine the coordinate is on which road. After that, we draw the GPS points on the road. Because we know the timestamp of each point, we can calculate the average speed of two adjacent points and just consider it as the speed of their midpoint. Note that the closer the points are, the closer the average speed is to the actual instantaneous speed. So the best way is to select adjacent points if our data is abundant.

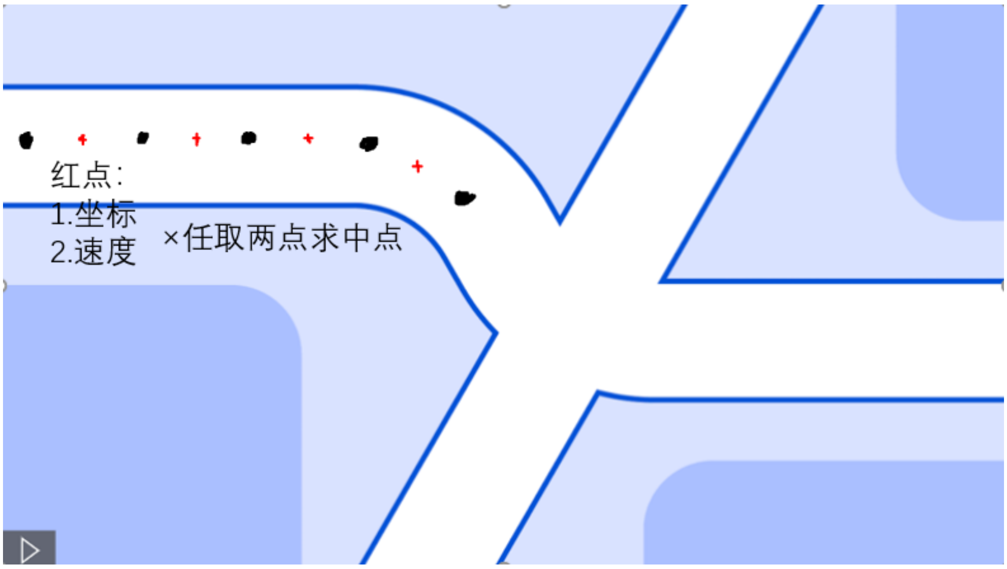
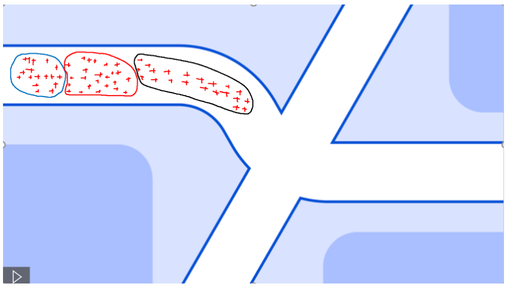


Figure 3: Find the midpoints

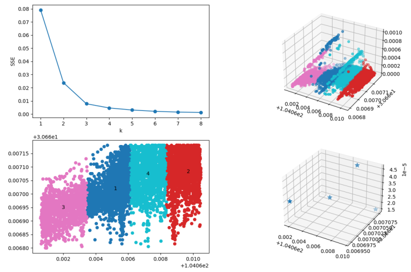
Next step is to apply clustering algorithm like *K-Means* or *MeanShift* to find a certain number of clusters of the midpoints. Every midpoint has three features:

- longitude
- latitude
- speed

So actually this is a three-dimension clustering, but the result we need is two-dimensional, which means we need to balance the weight of these features carefully to get a correct partition.



(a) Clusters



(b) K-Means Example

Finally, compute the convex hull of each cluster and find its intersection with the roads. Therefore, we have partitioned the roads into several segments.

2.3 Code Implementation

2.3.1 Map Matching

The first step is to locate GPS point into roads. This problem is called **Map Matching**.

In our dataset, a road is represented as a line, however, it should be an area in real world. Besides, it is hard to match a point to a 2-D line. Therefore, we need to convert a road to an area in advance.

Use method `buffer()` in *shapely* package.

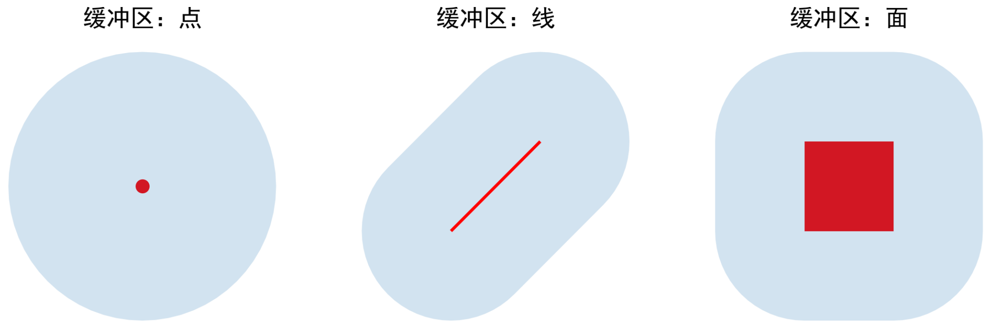


Figure 4: `buffer()`

The `buffer()` method will convert a `Line` to a `Polygon`.

Take 羊市街 + 西玉龙街 and the tracks of 10,000 taxis as an example. Actually there are three roads. Apply this function to the roads and we will get

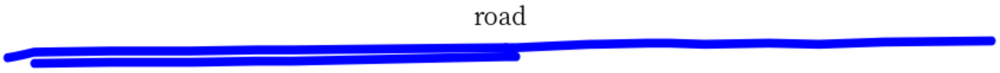


Figure 5: Buffered Roads

Secondly, filter the data. Use method `geopandas.GeoSeries.total_bounds` to get the bottom left corner and the upper right corner of these roads, then screen out the points which are not in the rectangle area.

Finally, for each point, find which road it in by `contains()` method in *shapely* package.

2.3.2 Calculate Midpoints

In this step, we calculate the coordinate and speed of the midpoints.

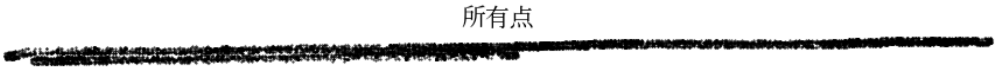


Figure 6: Matched Points

First thing to notice is that we should use the point in the same track, otherwise the timestamps are not continuous, resulting in a wrong speed.

In addtion, after working out the midpoints, we need to match these points again into the roads because there are some arc-shaped roads so that the midpoints may be in the interior of the arc.

Figure 7: Midpoints

2.3.3 Clustering

In this step, we use *KMeans* or *MeanShift* in *sklearn* package.

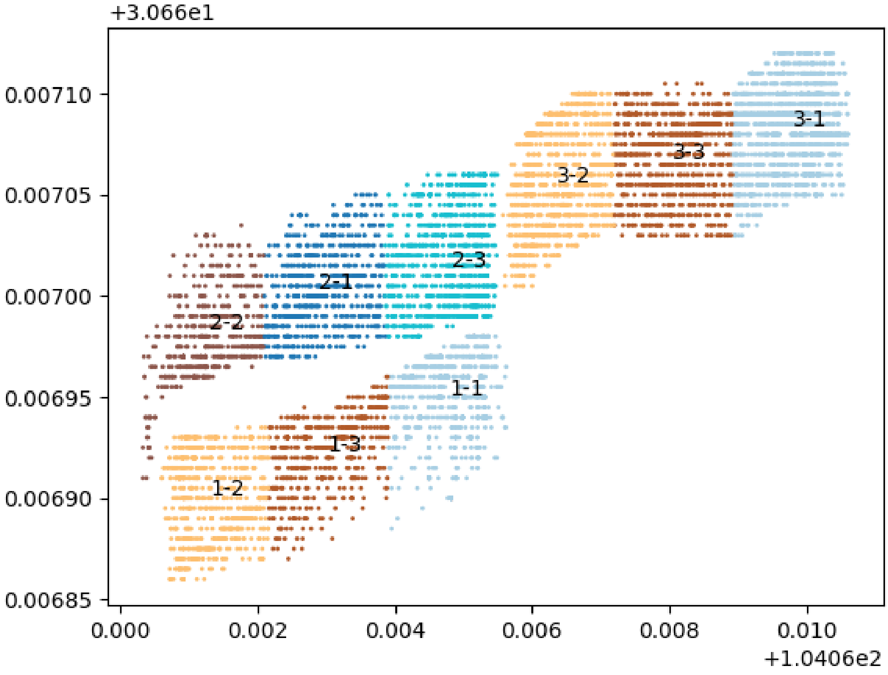


Figure 8: KMeans

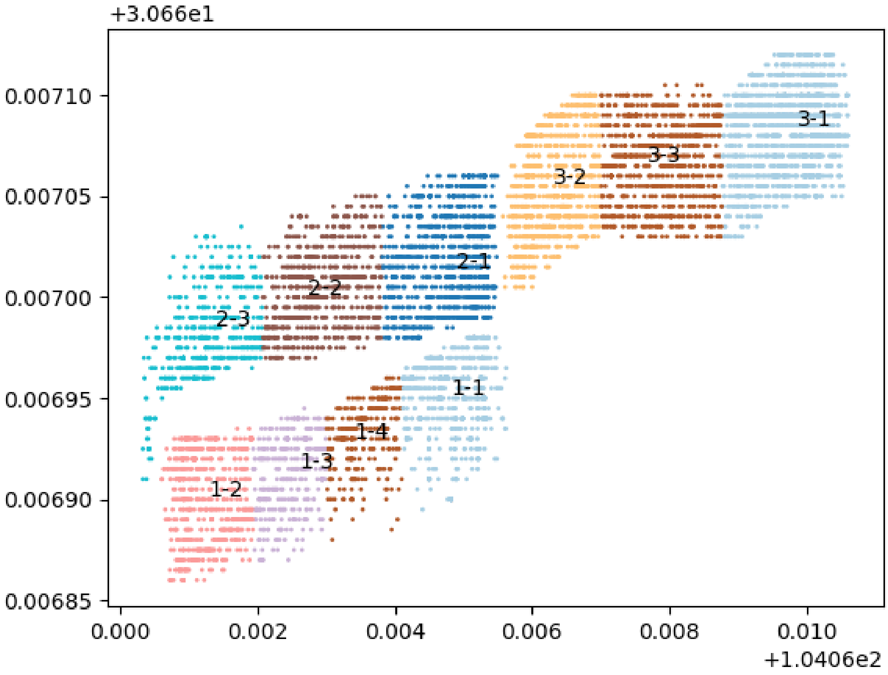


Figure 9: MeanShift

2.3.4 Intersection

The last step is to compute the intersection of the clusters and the roads.

First we need to calculate the convex hull of the clusters, because we cannot use the intersection of line and points. Use `geopandas.GeoSeries.convex_hull` to compute the convex hull of the clusters. Then use `intersection()` method in *shapely* package to get the intersection.

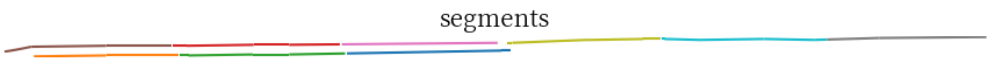


Figure 10: Segments

3 Computing TTI

The TTI (Travel Time Index) is industry's most used evaluation index of urban congestion degree is the ratio of the actual travel time and the free flow time.

$$TTI = \frac{\sum_{i=1}^{N} \frac{L_i}{V_i} \cdot W_i}{\sum_{i=1}^{N} \frac{L_i}{V_{free\_i}} \cdot W_i}$$

- L_i: length of the road
- W_i: weight of road
- V_i: real time traffic information
- V_{free_i}: free flow velocity

3.1 Dataset

- Data source: Didi Chuxing GAIA Intiative
- Region: Chengdu
- Time: 2018-10-01 to 2018-12-01
- Content:
  - GPS track of taxis with timestamps
  - Coordinate of road and district boundaries
  - TTI and average speed of districts

3.2 TTI calculation

Firstly, the trajectory data and road data are matched to obtain the trajectory points on the current road. This part of the processing is similar to the previous SuperSegment trajectory matching, and will not be repeated.

After trajectory fitting, corresponding velocities of each trajectory can be calculated. After the velocity is calculated, these points need to be further processed to remove the velocity value that will cause a large error to the result.

First, we need to deal with the point where the velocity is zero. Considering that this data set is Didi's taxi track, there are two kinds of conditions when the speed is zero: the driver is waiting for the passenger to get on and off the taxi, and the driver is waiting for the red light. As the process of the driver's waiting is not closely related to the road conditions, even if the road section is unimpeded, this situation will also occur, which will cause a large error to the result, so it needs to be removed. However, the time spent waiting for red lights, which is an important cause of traffic jams, needs to be reserved.

After the above processing, we also need to consider the processing of abnormal speed. Since the collection of position data depends on GPS positioning, and the deviation of positioning during data collection will have a great impact on the calculation of average speed, it is necessary to conduct statistics on the obtained speed, analyze the number of data with deviation, and then remove abnormal data reasonably.

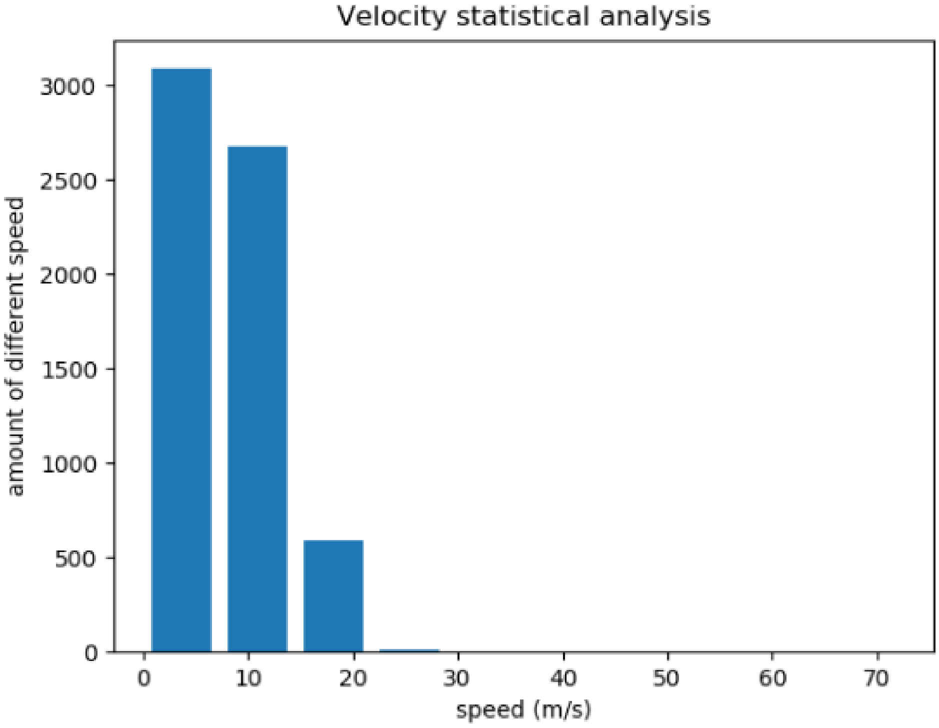


Figure 11: Velocity Statistical Analysis

In this statistical chart, there are a total of 6373 records. Among them, there were only 20 records whose speed exceeded 23m/s, accounting for less than 1%. Also, according to common sense of life, the speed of vehicles in urban areas is rarely more than 80km/h. It is reasonable to conclude that speed records exceeding this value should be discarded as outliers.

After the speed records are processed, the next step is to calculate the TTI. The calculation of TTI mainly focuses on the calculation of the average velocity of different time periods and the selection of free flow.

It is not difficult to calculate the average speed in different periods, but to divide the speed according to time and get the average value. The main problem of this part is that the traffic flow in different time periods varies greatly. If the time period is divided too carefully, the data volume in this time period may be too little. And when the amount of data is too small, using these data to represent the whole period of road conditions will fluctuate greatly.

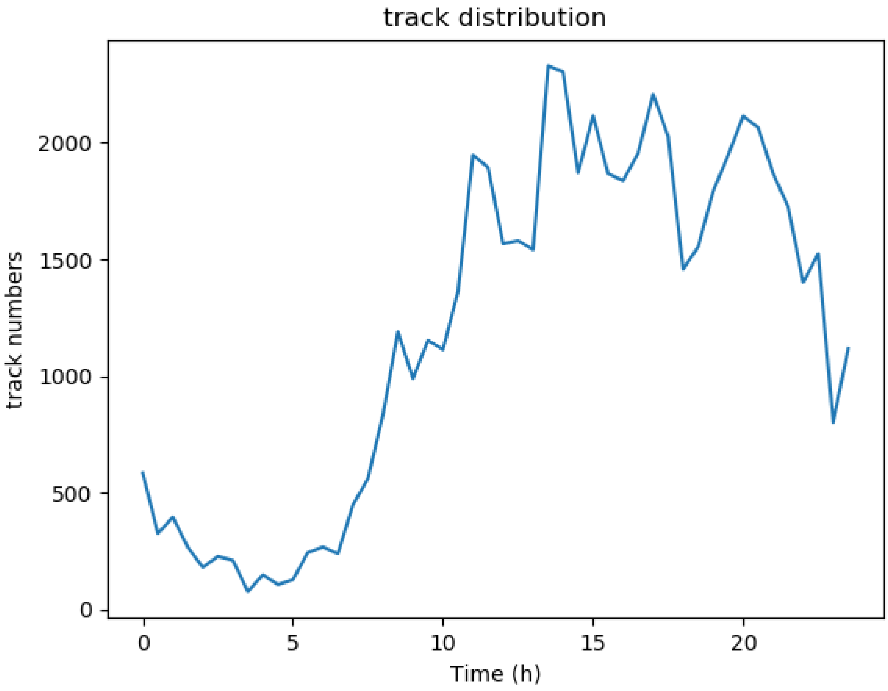
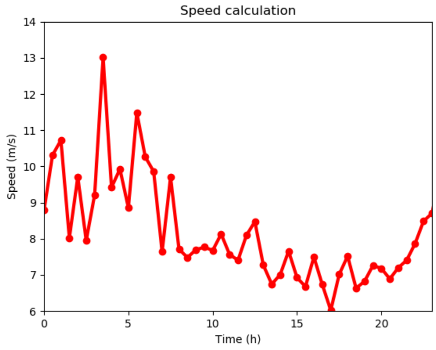


Figure 12: Track Distribution

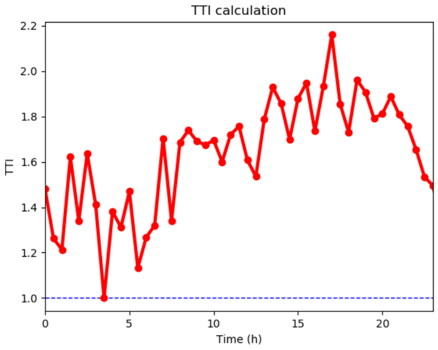
As can be seen from the distribution of the number of tracks, the number of tracks in different time periods varies greatly.

At present, the main solution to this problem is to use multi-day trajectory data to calculate. When the number of trajectories increases, there will be more trajectories even in a small period of time, thus reducing the fluctuation of TTI and making the calculation more accurate.

When the time interval is half an hour, the value of the average velocity and TTI:

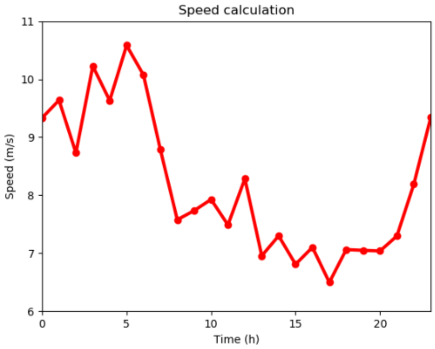


(a) Speed

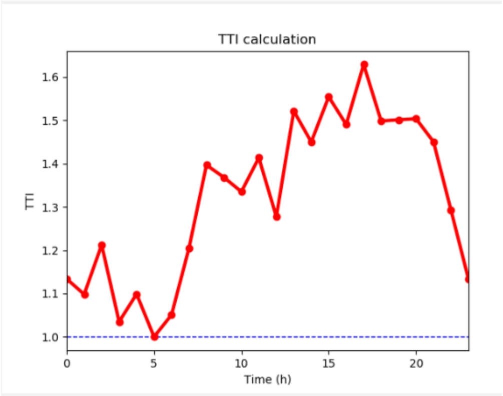


(b) TTI

When the time interval is an hour, the value of the average velocity and TTI:



(c) Speed



(d) TTI

In addition, it is also important to divide your time properly. Too detailed time division will aggravate the above problems, while rough division will not correctly reflect the TTI index of each time period. At present, after many experiments, the time interval is about an hour best.

4 UI Design

Our project based on database provided by DIDI. We mainly use driving track in Chengdu city.

I design this part as based on three parts of this project, supersegment, TTI calculation and TTE implementation.

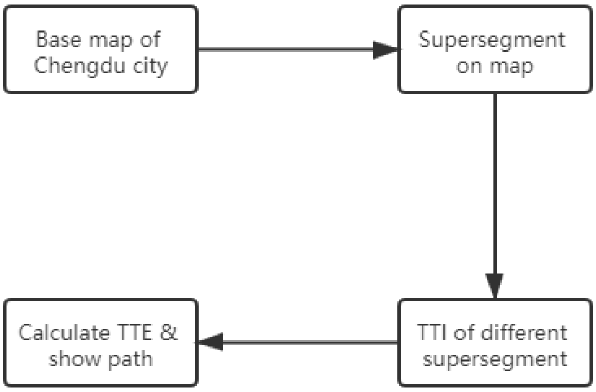


Figure 13: Baseline of Procedure

Firstly, the base map is shown as below:

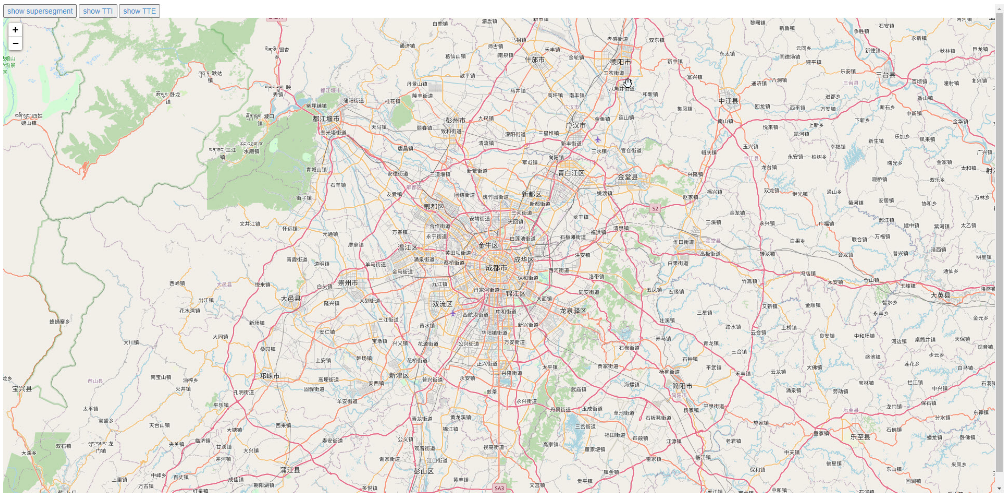


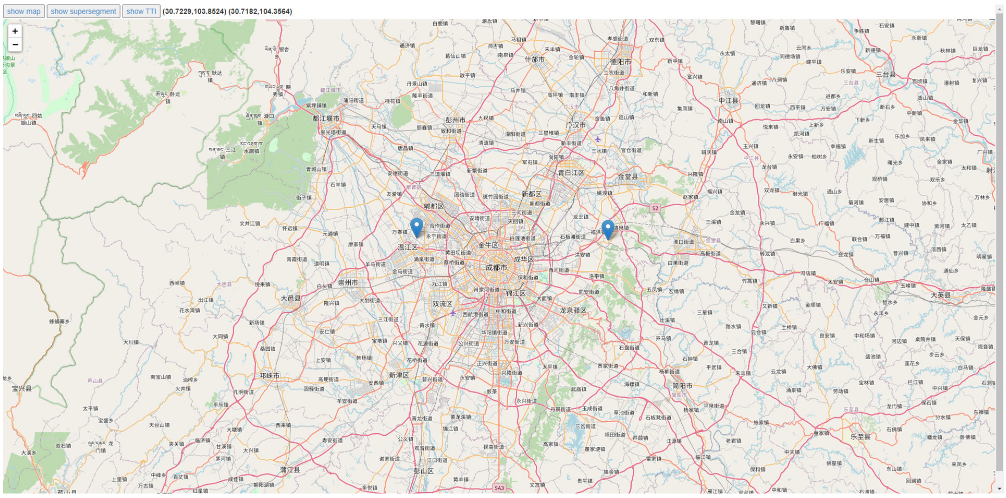
Figure 14: Chengdu Map

We can choose points from the map and it will show the latitude and longtitude of the point.

Secondly, we display supersegment of this map, this will be done in this semester.

Thirdly, we display TTI of this map using different color to represent the congestion level, this will also be done in this semester.

Lastly, we can mark two points on the map, and with front end and back end interaction, shows the best path to go from one point to another realtime.



(30.7229,103.8524) (30.7182,104.3564)

Figure 15: Choosing points