Journal of Electronics and Information Science (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2023.080305
ISSN 2371-9524 Vol. 8 Num. 3

A Path Simulator Focusing on Time Consumption-Based on the Transport Network and the Data of Public Traffic Vehicles in Shanghai

Dingju Wang, Bolun Zhang

Shanghai Jiaotong University, Shanghai, 201109, China
wangdingju@sjtu.edu.cn

Keywords: Path planning, Algorithm, Trajectory big data, Python

Abstract: With the rapid growth of every aspect of our society, people's schedule is getting tighter and tighter. Thus, there is a will for the masses to spend less time on road and get the precise time consumption, in order to form a neat schedule with little time being wasted. What we want to construct is a program, which will return the minimized time consumption and the best route. This project centers on the mission of simulate the transport condition of any time and find a rout which costs minimize time to reach the destination. The time we are simulating is not only present, but also for the future. The project can be mainly divided into four parts, including data acquiring, data processing, map building, the usage and comparison of graph algorithms.

1. Introduction

With the rapid growth of every aspect of our society, people's schedule is getting tighter and tighter. Thus, there is a will for the masses to spend less time on road and get the precise time consumption, in order to form a neat schedule with little time being wasted. What we want to construct is a program, which will return the minimized time consumption and the best route. When searching for related papers, we found that most routing program is focused on the traffic between cities, rather than in cities and is hard to guide our daily commute. To solve this problem, our project centers on the mission of simulate the transport condition of any time and find a rout which costs minimize time to reach the destination. The time we are simulating is not only present, but also for the future (for example, to schedule a conference on a workday's morning, the final time consumption will surely surpass the number revealed on the navigation app because of the rushing hour). The project is finished by Python and can be mainly divided into four parts, including data acquiring, data processing, map building, the usage and comparison of graph algorithms.

2. Related Knowledge

2.1 All Modules, Packages and Library Being Used

To install a module, we can:
(1)Use pip to search for the module directly;

(2)Find modules in official documents
Download file (.whl) with same edition of python;
Use ‘pip install path\filename.whl’ to install X;

2.1.1 OS

‘os’ is the attributive of the word ‘operating_system’. It supplies random interfaces between various Python programs and operating systems. The os module can easily interact with the operating system, and greatly enhance the portability of code. [1]

2.1.2 NumPy

“Numeric Python” is a library consisting of multidimensional array objects and a collection of routines for processing array. [2]

2.1.3 Pandas

Python Data Analysis Library is a tool, origin from NumPy. It is used to solve data analysis tasks. Pandas incorporates a large number of libraries and some standard data models, providing the tools needed to operate large data sets efficiently. Pandas provides a large number of functions and methods that enable us to process data quickly and conveniently. You will soon find that it is one of the important factors that make Python a powerful and efficient data analysis environment. [3]

2.1.4 Osm2gmns

Developed by ASU trans+ai lab team of Arizona State University, Osm2gmns can process connected network and simplify the network or build module automatically. For data inputs, it supports POI, which can be download from websites. [4]With random network included, unified format, it is widely used to build all kinds of models.

2.1.5 OSMnx

OSMnx is a package built by geopandas, network and matplotlib. The capacity of OSMnx and Osm2gmns is alike, but the latter is good at routing. [5]

2.1.6 Datetime

Datetime is a time related module, it can process year, month, day, hour, minute and second, return to the gap between two spots and return to the information of the time zone.[6]

2.1.7 Shapely

Shapely uses Python's ctypes module to perform set theory analysis and operation on planar features. The functions used are from the GEOS library GEOS is the migration of Java topology Suite (JTS) and the geometry engine of PostGIS spatial extension of PostgreSQL RDBMS. Point, LineString, LinearRing, Polygon… is included. [7]

2.1.8 GC

Generational garbage collection can significantly reduce the memory usage and accelerate the program. [8]

2.2 Trajectory Big Data

Thanks to the rapid development of spatial positioning technology and sensor networks, spatial positioning sensors have been widely used in aircraft, ships, cars, and handheld devices, generating, and accumulating a large amount of space-time trajectory data of moving targets. Trajectory data is the trajectory generated by moving objects in geographical space, which is usually represented by some columns of spatial points with time order. Generally, the elements of PI include: positioning point ID, trajectory ID, longitude, latitude, height, speed, time, etc.

Whole project can be devide into two parts, including graph generating and routing. Taxi data is split time-wise and generate weighted and directed graph during each period of time. When a request of navigation is received, select appropriate graph and use routing algorithm to generate the route and export the expected time consumption.

3. Process of Building Project

The over-view of the project is shown in Figure1, more detailed step-wise illustrations are as below.

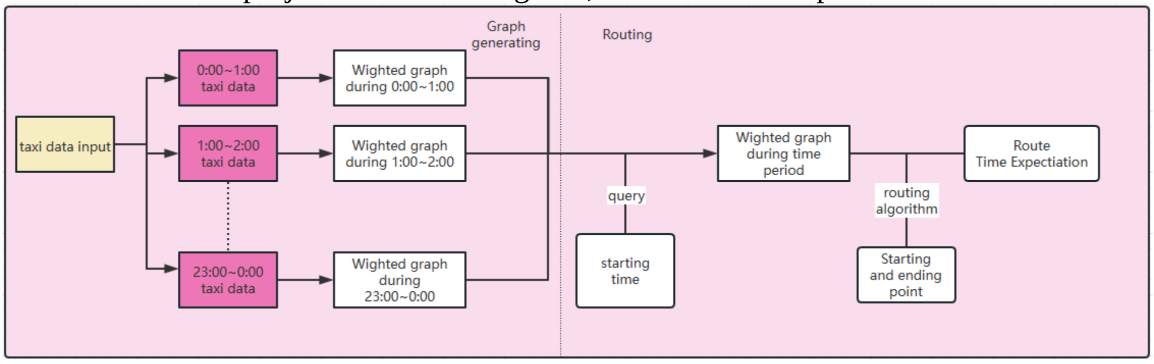


Figure 1: Process of Building Project

3.1 Data acquiring

Sources from Shanghai public data update platform[10]. The taxi data and road-net data is downloaded at this website.

3.2 Data processing

In this period, we are going to introduce the method we use.

3.2.1 Road net

While abstracting route planning problem as a TSP problem. Running the program, with all road included, time consumption is disastrous (over 2h). For the massive trajectory data set, there are a large number of redundant points. The graph generated with taxi data is shown in Figure 2.

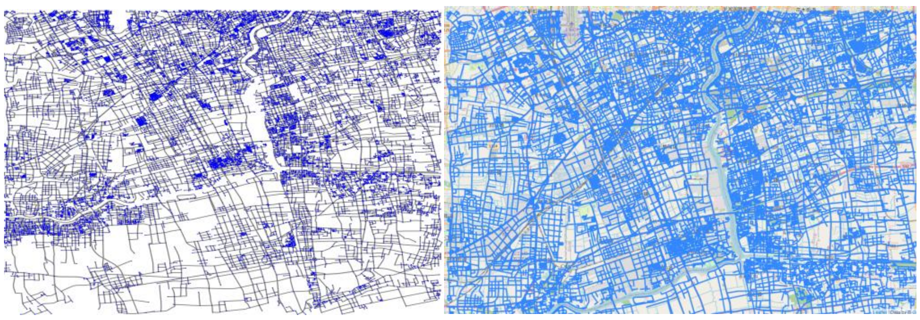


Figure 2: Graph Generated Overview (Left), Road Net Generated from Taxi Data Superposed with Ground Truth

3.2.1.1 Osm2gmns

Using osm2gmns to simplify the road net, return ‘node.csv’ and ‘edge.csv’. The running time of the program is proportional to the length of ‘edge.csv’, thus the running time of the program is significantly.

3.2.1.2 K-Means

Considering the feature of crossings: most of them have familiar distance, when searching for the same cluster, it is easy to get stuck at locally optimal value. More annoyingly, the constant k is hard to select, so we abandon this method.

3.2.2 Taxi data

Due to the running time of the program and the limitation of the performance of laptop, we decide to upgrade the data every 15min, so we place those data in a new folder. When shifting the data, we preserve the time, speed, and position of normal taxies and save the data as 'shanghai_taxi_spd_calculation_dict_byTime.npy'.

However, when I try to attach the speed value to every edge, no matter what border value is chosen, the average speed is always equal or missing (for a more precise view, please open the ‘spd’ folder, the number after k id the maximum value of the eccentricity of the oval, which stand for the range of the road). After the observation towards ‘node.csv’, ‘edge.csv’ and the real map, I found out that edge is not well-related with the real map. Also, because the imprecisely recorded GPS data (GPS has a precision of 15m, when cars travel through high buildings, the average error will increase significantly), taxies can’t be divided into related clusters by covering them with oral.

Finally, to separate the taxi data, we decide to use OSMnx to attach speed data to node.

3.2.3 Calculate Time Consumption for Each Edge

First, we attach all speed data to its nearest edges (with the format of ‘list’) and calculate average speed by function ‘.avg()’. Then divide the length of the road with average speed, forming a map with time as its weight.

3.3 Map Building (Visualization)

Using osmnx.graph_from_address (‘place’, dist = length, network_type= in[‘all’, ‘cars’,etc.]) to get street data and osmnx.plot_graph(graph, show=True) . An example of finding the shortest route from the campus in Minhang to the campus in Xuhui is shown in Figure 3.)

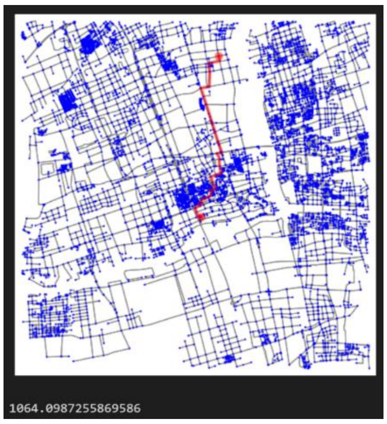


Figure 3: The Route Generated by Path Simulator (Red) and Time Expectation

3.4 The Usage and Comparison of Graph Algorithms

3.4.1 Ant Colony Algorism

3.4.1.1 Definition

Ant colony algorithm is inspired by the research on the foraging behavior of real ant colonies. Biological research shows that a group of cooperative ants can find the shortest path between food and nest, while a single ant cannot. Biologists have found through a lot of careful observation and research that the behavior of ant individuals is interactive. In the process of movement, an ant can leave a substance called pheromone on the path it passes through, and this substance is precisely the carrier of information transmission and communication between ant individuals. Ants can perceive this substance when moving, and are used to tracking this substance to crawl. Of course, pheromones will be released during crawling. The thicker the pheromone trace on a road, the higher the probability that other ants will follow and crawl this path, so the pheromone trace on this path will be strengthened. Therefore, the collective behavior of ant colonies composed of a large number of ants will show a positive feedback phenomenon. The more ants walk along a certain path, the more likely the latecomers are to choose the path. It is through this indirect communication mechanism that ant individuals achieve the goal of collaborative search for the shortest path. [9]

3.4.1.2 Problem Revealed While Trying to Use ACO Algorithm

After the simplification of road net, there are still 1348nodes left. The ant needs to ergodic every edge for several times to form the best result. Take a *Traveling Salesman Problem (TSP)* shown in Figure 4 as example, massive computation is required.

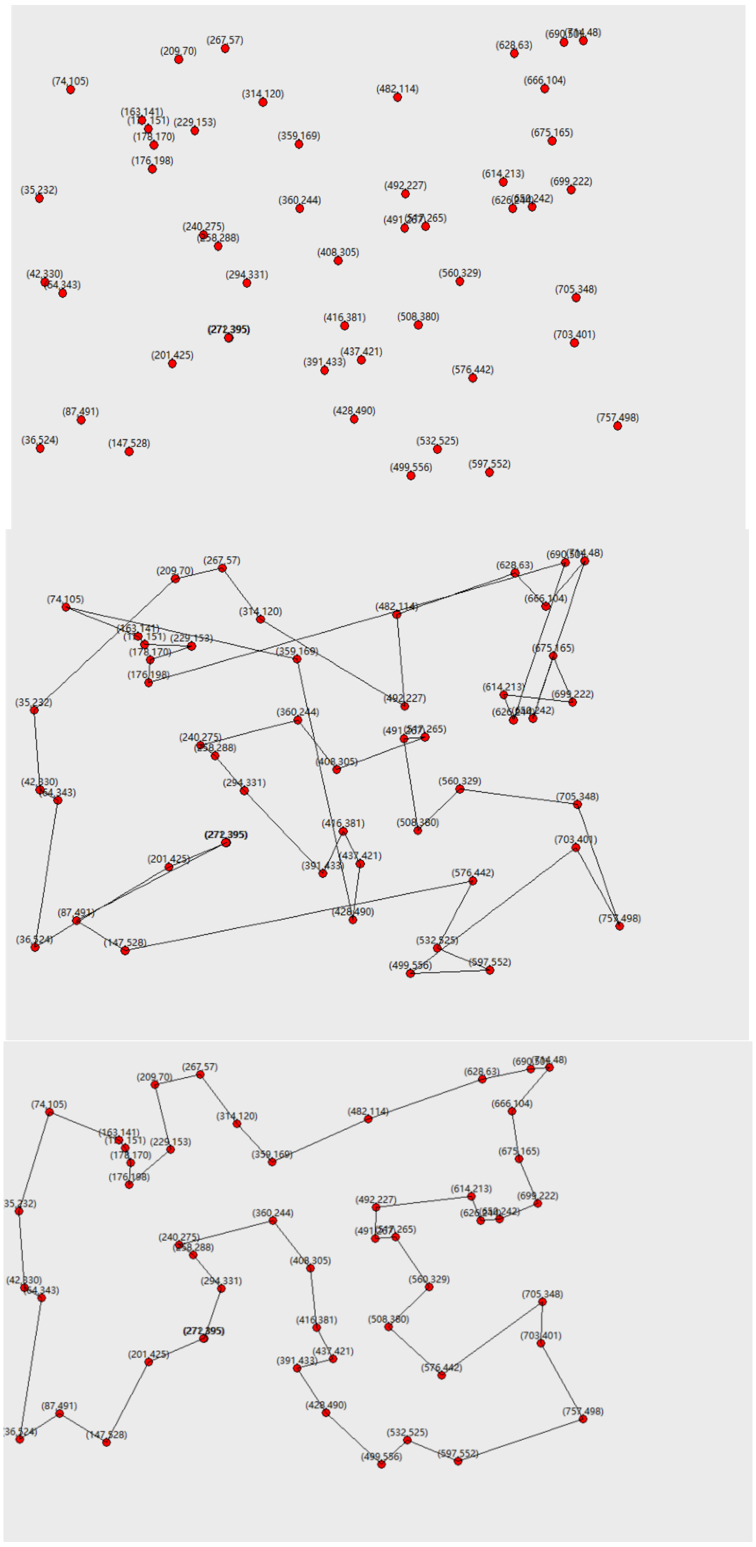


Figure 4: Traveling Salesman Problem (TSP)

Visualization of a 50-points TSP problem on three representative stages. The overall length reaches an optima in the 155th running. For the road net, the number of iterations and the possibility of converging to a local optimal point would be bigger for sure.

When the number of nodes is n, the number of all edges possible will reach $\frac{(n-1)!}{2}$, with the number of 1348, the number of possible edges is8.438299E3631. Unfortunately, for the nodes that are not linked with an edge, we can only set a high distance(eg.100km), but delete it, for the ant does not know those two nodes are not linked.

For the net of Shanghai: the total length is dropping, but in a relatively low speed and the time consumption is unbearable. The 74th running's graph looks just as same as the 4th. From the graph of the is obvious that it has a long way to go before reaching the best route, at least the middle will not be like a huge black spot. The number of iterations and the total distance is shown in Table 1.

Table 1: The number of iterations and the total distance of the TSP of weighted Shanghai road net

number of iterations	Total time(/sec)
1	134670
2	134644
34	115858
72	11473
87	11473
102	11473

The low productiveness while dealing TSP can somehow reflect the situation on dealing routing problem with an original a. Due to the limitation of computer performance, we stopped trying to solve routing problem with cute little ants and only give some ways for betterment:

(1)Inspired by the function 'has_path' in netwokx Set the track for the ant, so that we can reduce the number of edges significantly. We can define a function to test whether two node is reachable.

(2)Reduce some unimportant nodes after the designation of the start and end, which may reduce the number of edges by reducing the nodes.

3.4.2 Dijkstra Algorithm

3.4.2.1 Definition

The aim of this algorism is to find the shortest path from one vertex to other vertices in weighted graph. The main feature of Dijkstra algorithm is that it starts from the starting point and adopts the strategy of greedy algorithm. Each time, it traverses the adjacent nodes of the vertices that are closest to the starting point and have not been visited until it extends to the end point.

3.4.2.2 Advantage of Dijkstra Algorithm

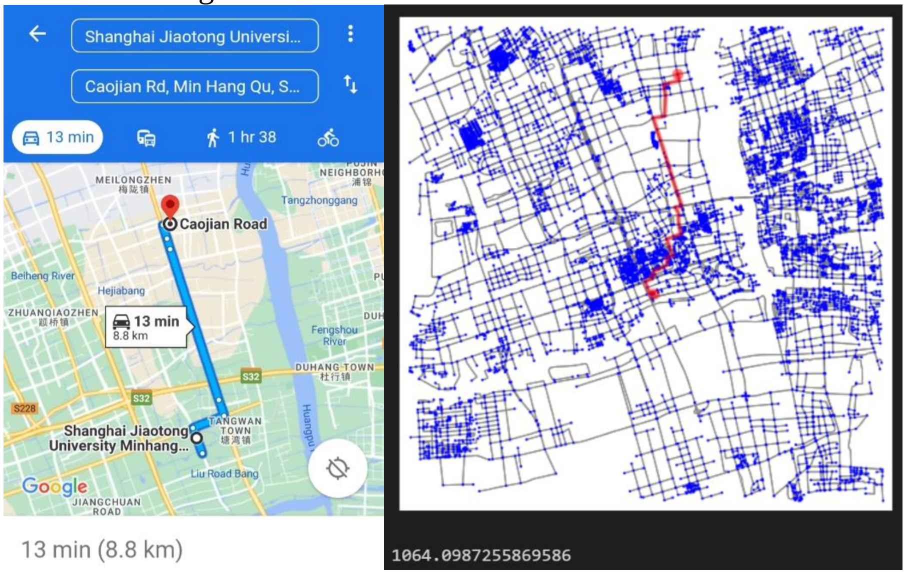
Dijkstra algorism, or greedy algorithm is easy to understand and implement. When dealing with numerous node, greedy algorism only needs to select the nearest node but consider the influence from other aspects. Thus, although it might not return the best answer, it is reliable and costs acceptable time.

4. Result Analysis

Due to the unsatisfiable calculation ability of lab-top, we only cut down a simple part of the road net. Now, we randomly take two attempt from all routing for example

4.1 First Routing

We set the starting time as 7in the morning. Google Map shows that it may cost 13 minutes, slightly shorter than what we have predicted(1064second) two routes are quite alike(despite the part of striking through the campus) . The differences of time are less than 2 minutes, which is a common error. The route is shown in Figure5.



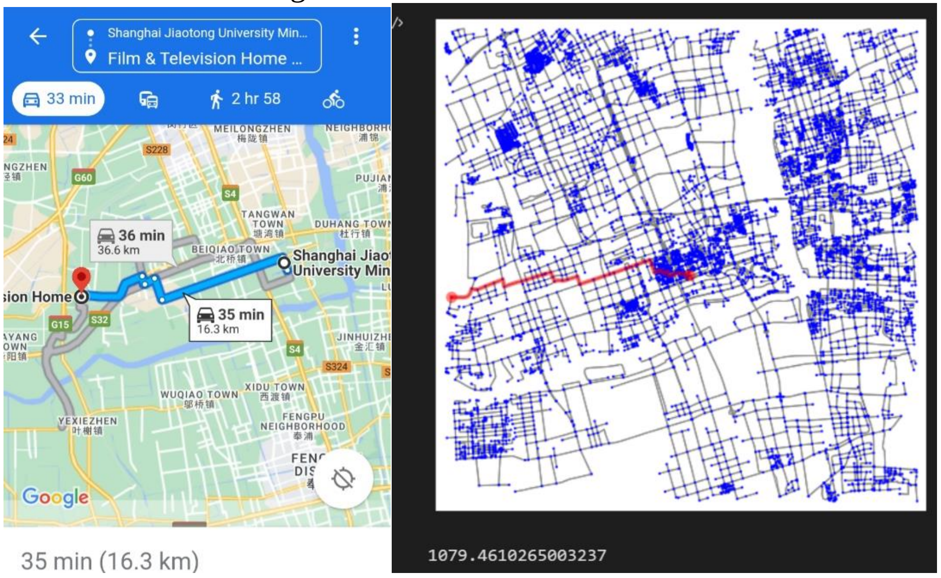
13 min (8.8 km)

```
1064.0987255869586
```

Figure 5: The Route and Expectation of Time Consumption from Navigator App (Left); Path Simulator between Shanghai Jiaotong University and a Movie Theater. (Right)

4.2 Second Routing

Google Map shows that it may cost 35 minutes, slightly longer than what we have predicted(1079second). To figure out the reason why the difference reveal, we compare the road net of each and find google map has some roads not included in their decision progress, also consider the time efficiency of data and the rapid growth of economy in Minhang District, the difference is acceptable. The route is shown in Figure6



35 min (16.3 km)

```
1079.4610265003237
```

Figure 6: The Route and Expectation of Time Consumption from Navigator App (Left); Path Simulator between Shanghai Jiaotong University and a Movie Theater. (Right)

5. Problems and Betterment

5.1 Relate to Reality

During the process we found out while routing the path from campus in Minhang district to campus in Xuhui district, the final path has crossed the campus, which means the simplification program can't tell whether the road is usable for most drivers. For the same time, the data regression reshapes our 3-D world into a 2-D graph, the average speed, and the ability to turn to another road for highway and frontage must be different. So, the final route might not be suitable in the reality.

5.2 Time Consumption

The program is in great needs to be optimized, for it spends too much time. We have already use gc. In data processing, but our laptop continued to break down during the process of running the code.

5.3 Future Plan

First, the running time must be reduced to meet the need of route regulation, because no one wants to make a time table all night. Second, during the process, we noticed that the number of cross roads may affect the accuracy of our prediction. Thus, we are going to do a gradient descent with predicted time, number of crossings and other features to reach a more precise result. Third, take a further look at *Ant Colony algorism*, to replace *Dijkstra algorithm* if the running time is acceptable.

References

[1] https://docs.python.org/zh-cn/3/library/os.html,2022-07-15
[2] https://www.numpy.org.cn/,2022-07-15
[3] https://www.pypandas.cn/,2022-07-15
[4] Boeing G. (2017). OSMNX: New methods for acquiring, constructing, analyzing, and Visualizing Complex Street Networks. Computers, Environment and Urban Systems, 65, 126–139. https:// doi.org/ 10.1016/j. compenvurbsys. 2017. 05. 004
[5] Yang M., Luo W., Ashoori M., Mahmoudi J., Xiong C., Lu J., Zhao G., Saleh Namadi S., Hu S., Kabiri A., & Ji Y. (2023). Big-Data Driven Framework to Estimate Vehicle Volume Based on Mobile Device Location Data. Transportation Research Record, 0(0). https://doi.org/10.1177/03611981231174240
[6] http://www.lvesu.com/blog/php/datetime.examples-arithmetic.php, 2022-07-15
[7] https://www.osgeo.cn/shapely/manual.html, 2022-07-15
[8] https://docs.python.org/3/library/gc.html, 2022-07-15
[9] Blum C. (2005). Ant Colony Optimization: Introduction and recent trends. Physics of Life Reviews, 2(4), 353–373. https://doi.org/10.1016/j.plrev.2005.10.001
[10] https://data.sh.gov.cn/