A helpful GUI for data visualization of Hangzhou Metro Traffic

Summary

The subway is ahead of other cities' public transportation methods in terms of green, environmental protection and traffic volume, and has become a priority transportation mode for major cities. There is a surge in urban metro traffic in China in the past couple years, but with the new lines being put into operation and the expanding of the network scale, the problems are also emerging. It’s unsafe and inconvenient for the children and elderly people to take the subway in rush hours, and numerous subway stations keep outsiders from knowing how to arrive their destinations. In order to solve the problems we’ve mention about, I design a Graphic User Interface to help users have a better understanding of subway inflow and outflow trend, and help outsiders plan feasible routes to their destination.

Firstly, I implement an initialization interface for users to select the dataset they want this program to display and select the adjacency matrix of the stations. File dialogs are used to help users select their csv files.

Then I design a main interface to guide users to the parts they want to use. It gives users four choices, which includes selecting data part, flow view part, plan path part and the reselect dataset part. StackWidget is used to realize the function of page change.

To let users select their interested fields of the dataset, a data filter interface is designed to help them select. After they make sure the fields they are interested in, they can go back to the main interface to use the other parts. Lazy evaluation is used to minimize the unnecessary loading.

After that users can choose the flow view part to see the trend of traffic inflow and outflow over time. They can tune the date, station, start time, end time, time step parameters to see their interested parts. Amortization is used to amortize the expected calculation cost. Interpolation will be used to smooth the trend line when only a few points are available.

Finally, plan path part is designed to enable users quickly acquire a feasible route to their destinations. Transfer cost is taken into consideration while planning the path.

Combining all the mentioned parts, a convenient, good user experienced GUI is created. In the end, according to the peak hour factors: occurrence time of peak, peak hour span and comparison of morning and evening peak traffic, I divided these 81 stations into three types. Each station type has its own characteristics that can be clearly seen from the graph view.

A helpful GUI for data visualization

of Hangzhou Metro Traffic

December 21,2019

Contents

1 Introduction

1.1 Problem Background

There is a surge in urban metro traffic in China in the past couple years, which causes overcrowding in metro systems of many cities. And numerous subway stations keep outsiders from knowing how to arrive their destinations, even the native people want to take subway to places they are not familiar with can take them a long period of time to plan path. What's more, it puts forward higher requirements for the subway operating organization to effectively clear the passenger flow of the subway in time and maintain the facilities in the subway station.

1.2 My Work

A Graphic User Interface is designed to help subway organization have a better understanding of the inflow and out flow trend of each station. According to the data which is displayed visually, the subway organization can take measures to better unblock the crowd and maintain the subway facilities in time. The plan path part of the program enables users to input their start station and end station and it quickly plans a feasible way for them. After that I will analysis the data from the plot and divide the stations into three types according to the peak hour factors. I hope this classification will be helpful for the subway organizers and the passengers.

2 Assumptions

In order to simplify the problem, I make the following assumptions to let the GUI solve specialized problems.

- The dataset folder should include all the data files you want the GUI to show.
- The dataset should satisfy a certain format: all files in the dataset should end with .csv and the date of the data stored in this file should be present in the file name and separated by '_' (For example: record_2019-01-07_part00.csv).
- The subway line message should be stored in a csv file and use "0" to represent there is no edge between two stations and "1" to represent a edge between two stations.
- The time cost between two stations is considered the same, according to the distance consideration of the subway department in planning and construction.
- Users should make sure that the dataset and adjacency file they choose are correct. Otherwise the program will not read the data properly.
- The Items in the dataset should meet a certain format: They should include all the seven fields: Time, LineID, StationID, DeviceID, Status, UserID, PayType. And the sequences of these items shouldn't be changed.

3 Implemented Details

3.1 Initialization Interface

QFileDialog is used to implement the Initialization part. When the user clicked the button: Select Dataset or Select Adjacency, QFileDialog's functions will be called to reap the dataset path. In Select Adjacency case, the program immediately read the data in the file to the memory, so when users use the Plan path part, the program would not have to read again. But in Select Dataset case, considering the great amount of the data in the dataset, I use the lazy evaluation to avoid user waiting too long at once. So after they choose the dataset, we just get all the file names and all the file absolute path into the memory. When we really have to use the data to form the plot, then the program will read the related data through the absolute path we've had before. QFileInfoList is used to get the information (name&&path) of the data. After user select the adjacency matrix, the interface will jump to the main interface automatically.

3.2 Main Interface and Page Change

In the main interface, there are four choice for the users and each choice will bring the user to the corresponding part. StackWidget is used to realize this page change mechanism. Each page of the StackWidget represents an interface of GUI. Using setCurrentIndex() function we can realize the page change.

3.3 Data Filter and Lazy Evaluation

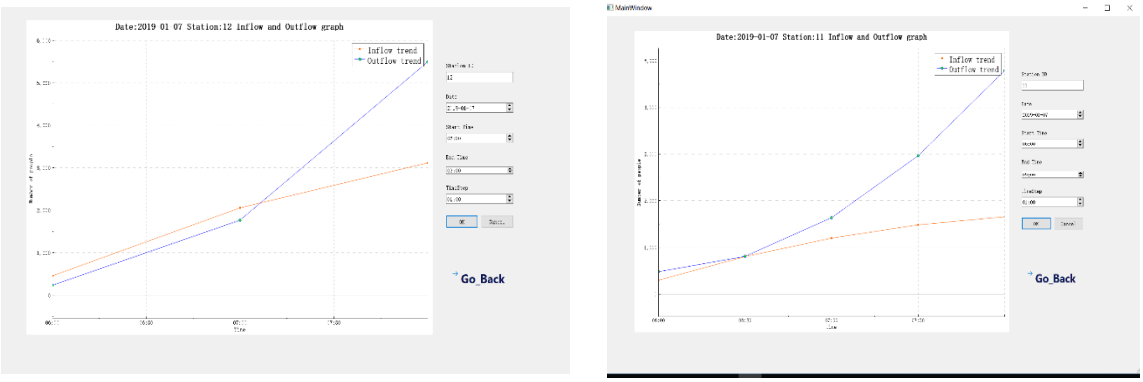
In the Data Filter part, user can choose their interested fields of the dataset. I use a bool array to represent the fields. Array[0] to Array[6] represents Time, LineID…PayType sequentially. If the user choose Time item in the filter interface for instance, Array[0] is set to true. When the user clicked the OK button below the checkboxes, their choice will be saved. Otherwise the Array will remain the same value. When the user clicked Cancel button, the checkboxes will display the user's previous choice. So it is very convenient for the users to use this interface. Lazy evaluation is used to avoid unnecessary calculation. For there exists some cases that after the user clicked OK button, they want to change their interested fields at once.

3.4 Flow View and Amortization

In order to enhance the user's experience and display the information in the plot as clear as possible, I choose QCustomPlot to visualize the data. Firstly, we need to reap the information in the user interface. The information includes: StationID, Date, Start Time, End Time and Time Step. For the reason that the dataset contains a great amount of data, so it will absolutely take a long time to read all the data into the memory. To make things worse, some data we've read in the memory might never be used. So I decided to take the strategy of amortization. Each time the program only read the data of the given date and given interested fields. One advantage of this strategy is that if the users didn't change the date, even if they frequently change the other parameters, the program will form the plot quickly.

Another advantage of amortization is that we can minimize the unnecessary operation of the data that might never be used. The program will first check whether the date is changed or not. *If the date changes, we need to read new files, if not, we don't* need to *do anything, just* use the data in the memory. When the date changes, the program will traverse the FileName Array which is initialized in the initialization interface and find all files of that date. Because the files in the dataset meet a certain format as we've mentioned in the Assumption, the program just need to get start index and end index of the FileName Array. After we get the file range, a while loop is called to read all the data into the memory. I make an improvement here to reduce calculation of sorting by StationID. The amortization strategy is used again. For the fact that the number of station will not change in a short time, so once the data item was read, it will be directly stored in the corresponding memory according to its StationID. By the way, I make an example of using the filter. If *user select the LineID item in the filter, the program will read this part of data and display stations that each line contains in the Plan Path part. If not, the program will not read the LineID data from the dataset.*

QCustomPlot is used to form a clear graph. Firstly, according to the information (Start Time, End Time, Time Step) from the user, we need to decide how many points will be displayed in the plot. Then we read data items that belongs to specified station from the memory. The program will traverse these data items to count the number of inflow and outflow people in Time Step period. Interpolation is used to smooth the trend line when the point's number is less than 6. Real data points will be used as the interpolation points. And midpoints of two adjacent real data points will be used as the input of the interpolation. Then we use the output of the interpolation function and the corresponding inputs to form new points which will let the trend line become smoother. Actually I've try the QSplineseries to form the curve line. But I find it didn't clearly display the information, so I didn't use that. The result of Interpolation is shown in Figure 1(b).



(a) Trend Lines before Interpolation (b) Trend Lines after Interpolation

Figure1: Comparison of using Interpolation

3.5 Plan Path and Improvement of Warshall_Floyed Algorithm

In the part of Plan Path, I use Warshall_Floyed Algorithm to find a feasible path for the user. To make planning more practical, transfer cost is taken into consideration. As we all known, transferring lines may take us a long time and may need us to walk a long distance. Especially when an outsider carries heavy luggage or an office worker who gets tired after the work, they will definitely reluctant to transfer lines. So I add a transfer factor while planning the path. For people in a hurry, the transfer cost factor is set to 1, which represents its cost is equal to the cost between two adjacent stations. For people who want as less as transfer lines as possible, the transfer cost factor is set to 2. So the program will plan more practical path for different people.

4 Data Analysis

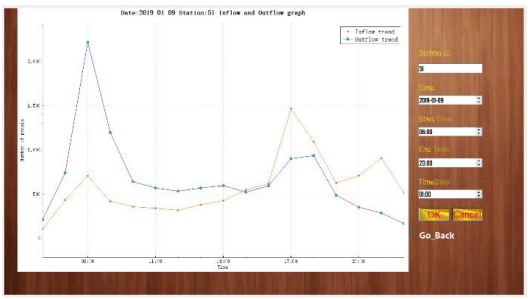
4.1 Peak Hour Factors

According to the graphic view of different stations' inflow and outflow trend, three main characteristics are extracted, which include occurrence time of traffic peak, peak hour span and comparison of morning and evening traffic peak. These three characteristics are all attribute to peak hour factors. Through the peak hour factors, the stations of Hangzhou Metro system can be divided into three types. The first type is residential station type. It is characterized by its peak occurs on weekday mornings and its peak hour inflow number of people exceeding the outflow number of people. On the contrary, stations that form the peak on weekday nights and with their inflow number of people exceeding the outflow number of people are defined as work-type stations. Stations that have a long peak hour span and have a high average of whole day traffic flows are defined as traffic-type stations.

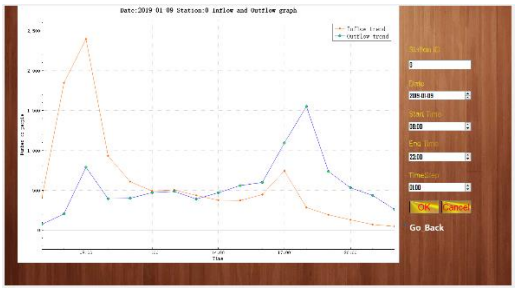
4.2 Typical Representatives of Three Types

According to the classification criteria, the 81 stations are divided into three groups.
Residential station group includes the following stations:
0,1,6,17,19,21,23,26,27,28,29,30,31,32,34,35,36,37,38,39,40,41,49,60,64,65,67,68,69,70,71
72,74,79,80.
Work-type station group includes the following stations:
44,45,47,50,51,52,73,75,76,77
Traffic-type station group includes the following stations:
4,7,9,15

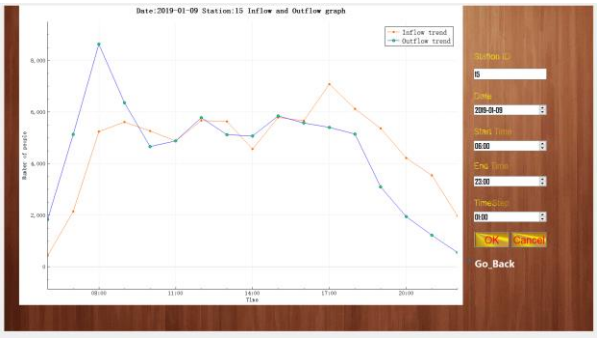
Typical graphic views of these three type are shown in Figure2 (a) and (b) and (c).



(a) Woke-Type



(b) Residential-Type



(c) Traffic-Type

Figure2: Typical views of three types

5 Results

Users can use this GUI conveniently and see the data performed graphically and clearly. Users can also get a quick and feasible route to their destinations after using the Plan Path theory. According to the division of the stations in Hangzhou Metro System, subway organizers can take different measures in different type stations to effectively unblock the crowd and maintain the subway facilities in time. Passengers who have viewed the inflow and outflow trend of their interested stations and time can have a better understanding of the Metro traffic and make more reasonable travel plans.

6 Advantages and Weaknesses

6.1 Advantages

- Users can use this GUI conveniently.
- The data is displayed clearly and truly.
- The GUI can present the results within a reasonable time.
- The interfaces of the GUI is friendly for the users.
- Robustness. The GUI will not crash even if the user's input is incorrect. And the GUI will indicate what users should do to rectify the mistakes.

6.2 Weaknesses

- Do not use dataset processing methods and related technology.
- Do not use multi-threading programming

7 Discussions

Database-related technologies are specifically designed to process and manage large volumes of data. They can help us reach the data and process the data conveniently and quickly. Multi-threading can be very helpful to amortize the necessary calculation so that when users need to view the graph, the program can give faster results. Using these technologies can form an advanced GUI version.