2019

Internship Project Report

ANALYSIS OF THE ROAD NETWORK OF UTRECHT

ROGER RUIZ SALVAT

1. Introduction

MM Guide provides consultancy services to businesses in a data rich environment. MM Guide helps businesses grow from a data rich environment to a decision rich organisation. The company is in Utrecht, which experimenting some issues with traffic congestion that have increased the past years. Because the city area is so small for the number of people that goes through during the day, the roads become often congested. The measurement of the traffic activity around the city area is a way to determine the mobility activity as well. This project analyses the road network context in the area of Utrecht and provides predictive values from simulation inputs. Traffic data is used to perform the analysis. It also provides data visualization tools to make the results accessible. The project orients its methodology in the big data and data analysis fields providing research and experience.

2. Main question and sub questions

Can we predict the impact of an incident on traffic flow on the road network around Utrecht?

From this main question the following sub-questions are derived:

1. How big is the Utrecht road network and which elements compose it?
2. Does exist any previous research applied to other cities?
3. How is the network mathematically represented?
4. How is the network mathematically analysed?
5. What statistics can be applied on the historical data?
6. How can the prediction models be generated?
7. Which software tools are suitable for the analysis and retrieval of data?
8. How can the obtained results be represented?

The goal of this research project is to answer the mentioned questions.

3. Methodology

Some previous knowledge is needed in order to make sure that just the required information is extracted. First, some preliminary research must be done by getting more specific details of the project context. This preliminary research will be mainly focused on the Utrecht road network in order to get some information about it. For example, by knowing its clime to see if this can be a frequent adversity for the traffic flow. Or if there is any event in the city that is regularly going on which could affect to the traffic density.

Another research field of the project will be on previous mathematical modelling done on other cases related to the traffic flow. It can be useful to orient the project to some directions. The idea is not to copy others work (each case for each city might be different), is to implement similar methodologies that are well known to be successful. This research includes theoretical flow concepts as well.

The road network needs a mathematical representation to be able predict certain circumstances by building mathematical models. In order to generate the most accurate representation, research on some mathematical concepts must be done. What is already know is that the representation will be in a graph form. It allows calculate the routes, densities, centralities and the connectivity between the roads. So, the research here will be done on how the graph can be generated including any kind of software or programming needed.

Once the network representation in obtained, is time to analyse it. Research here will be based on network analysis. The most important graph concepts and analysis will be selected to build the models. Mathematical research will take part here to figure out which of these concepts are the most relevant. Besides, the software used to do the analysis and any kind of tool that might help to complete these tasks will be researched.

Part of the analysis is also based on statistics. So, not just the graph concepts will take part in this project. To do statistical analysis some steps must be done before. Starting from obtaining the data, cleaning it, and exploring and analysing the obtained results. Then, conclusions can be taken. Besides, these statistics can be useful to build some predictive models based on historical data. So, finally, these models will be generated. To do all this analysis, some specific software or programming libraries will be needed. This will be researched as well.

To build the prediction models the previous statistical analysis is needed. Based on that statistics the modelling is more specific to the study case. Moreover, the modelling done in other projects which also analyse the traffic flow, can be used as an example to follow. Mathematical research on how to generate those models might also be needed.

To do the analysis and all the processes related to the project, such as data retrieving, specific software is needed. First, the historical data must be loaded. To do that, a technology able to deal with large amounts of data must be chosen. Then, some other tools to do graph analysis might be selected as well. For statistical analysis and modelling, it might be useful to use specific tools as well. A programming language to run these tasks could also be chosen.

Finally, to make the results accessible its fundamental to visualize them. To do that, all the possible options will be considered. Accessibility, a user-friendly interface and updatability will be the main three features to be prioritized. The technologies to be used and how to be used will be researched in order to be sure that the most suitable one is picked.

4. Results

a. Preliminary research on the Utrecht road network

Utrecht, the 4th largest city in the Netherlands, is experimenting a growth in its urban activity. Its central location makes it a clue point as a national logistic point. It is an expanding city. In the past 18 years it gained 96.402 habitants which is the 27,3% of the actual population. Moreover, its economy is constantly developing. The university of Utrecht is one of the most important schools in the country, and many students travel to the city to attend their classes. Due to the relevance of the city, the transport infrastructures must be able to handle a big number of travellers (around 195000 per working day in the railways) and manage possible issues in an optimal way.

Before doing the analysis and modelling tasks, it is important to know some characteristics of the Utrecht road network. An important characteristic is the size of the network in order to know where to look at. After some discussion, is decided that just the city area will be considered. This area is surrounded the main highway roads which facilitates the traffic flow to all the main directions. These roads will delimitate the diameter of the area and will be the focus of analysis.

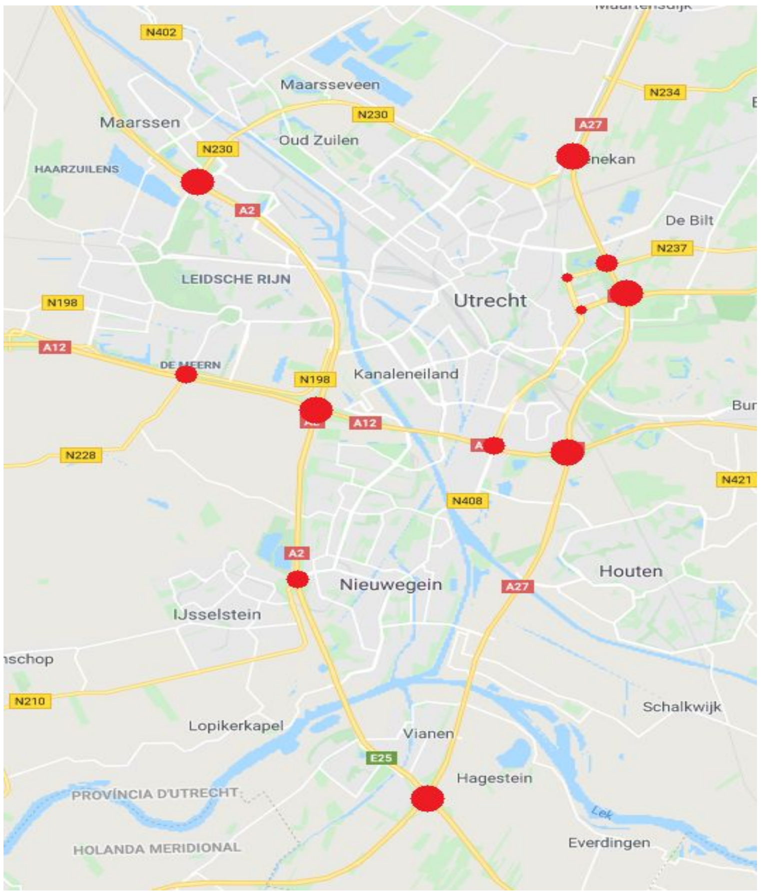


Figure 1.0 Utrecht map sketch

The image shows the main components of the Utrecht road network. There exist connecting roads which connects the distribution traffic points of the city and exit roads which are access ways to the urban area. The traffic flow and density will be analysed on these main elements through the several measurement points that are over network.

An important characteristic to research is the urban activity that may affect the traffic density of Utrecht. Any possible event, the rush hour period or any fact that can increase the number of vehicles on the Utrecht roads, should be known in order to make the modelling easier. Another important fact that can influence the traffic is the weather. The changes on the traffic flow depending on the weather conditions based on historical data, can be used to make predictions once the meteorology in known.

During the week the traffic flow and density is incremented during the rush time. The rush time usually goes from 6:30 to 9:00 and from 16:00 to 18:30. Besides, events such as football games in the stadion Nieuw Galgenwaard, which have a capacity of 23750 spectators, can also affect the traffic fluency. In the city centre there are the most event locations such as Jaarbeurs which can also cause a peak on the traffic density. The central train station can also lead to an increase of the number of travellers especially with international train arrivals.

For the well-known correlation affecting the traffic density, the weather conditions, research is done too. The rain usually causes retentions or even some incidents, especially if the precipitation is heavy. The weather conditions off the city of Utrecht shows that the rain is frequent but not so intense. This implies that the traffic flow is affected when it rains, which occurs often, but it is still fluent.

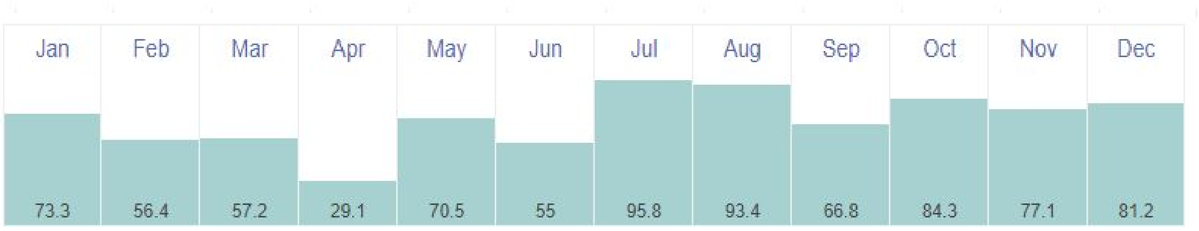


Figure 1.1 average precipitations in mm in Utrecht during the year

b. Preliminary research on previous traffic modelling studies

The traffic flow theories have been developed from two main approaches. Microscopic models look at the vehicles as individuals, and the macroscopic models consider the vehicles as a mass. Some other theories are hybrid due to a mix of microscopic and macroscopic variables in its models. This project will just work with macroscopic approaches as it is fundamentally a big data study.
Macroscopic traffic flow theories are based in three main concepts. The traffic flow, number of cars per unit of time, speed, and density, number of cars per distance. From this last one it can be obtained the spacing between cars as well. The mathematical formulas of these three main concepts are:

$$q = \frac{n}{\Delta t}$$

Where q is the traffic flow, *n* the number of detected vehicles, and t the interval of time of the taken measurements.

The speed is theoretically the distance covered per unit of time. For an individual vehicle it would be described as:

$$v = \frac{\Delta x}{\Delta t}$$

Where v is the speed of the individual x the distance travelled and t the time interval. However, in practice it is not possible to track each speed for each vehicle. Macroscopically is used an average of speeds tracked in a certain time interval.

$$v_t = \frac{1}{m}\sum_{i}^{m} v_i$$

Where m is the number of measured vehicles. There exist space means to macroscopically track the speed, but in this project just the time mean is used.

Finally, the density is described as:

$$k = \frac{n}{\Delta x}$$

Where k is the density. Combining the three variables it is obtained:

$$k = \frac{q}{v}$$

This equality is used to obtain the density from flow and speed data. From density it is calculated the spacing between cars as well.

$$s = k^{-1} = \frac{1}{k}$$

Once the fundamental concepts are known, it is possible to see how are related between them. This is known as the three fundamental traffic flow diagrams, which show the relation between flow and speed, flow and density, and speed and density. From these relations it can be obtained some optimal values which give clue information of our system. First, the flow density relation is described as it follows.

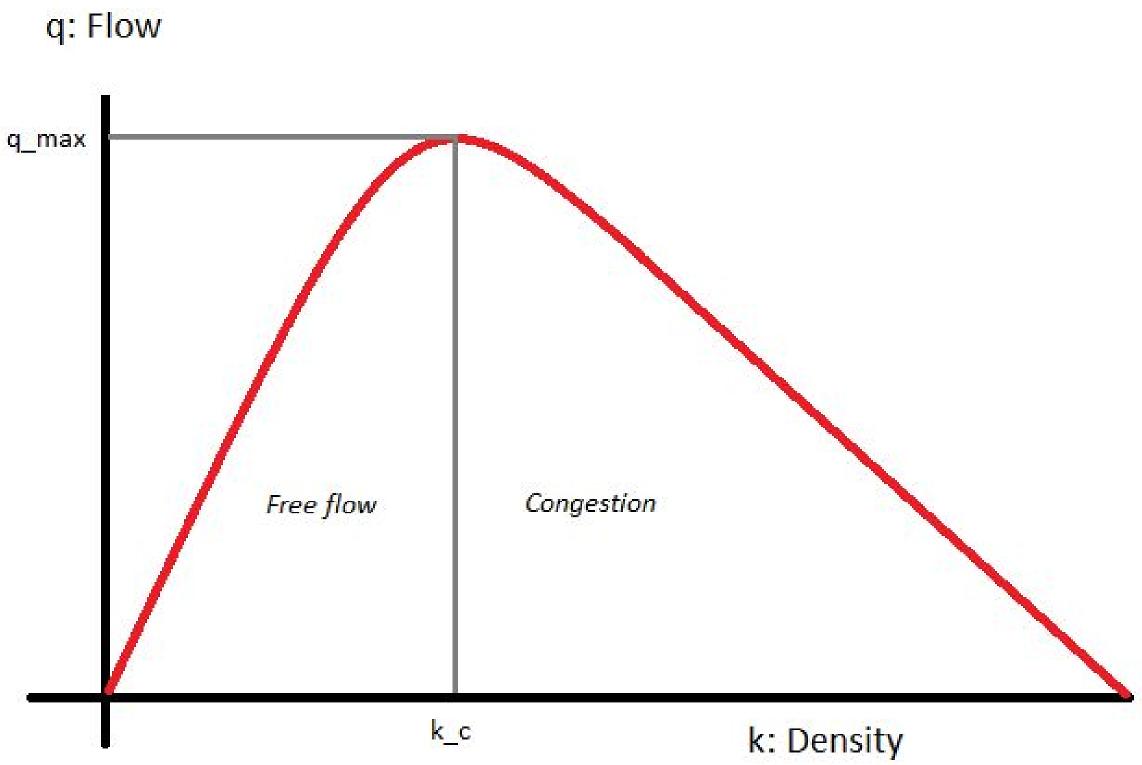


Figure 2.0 flow – density fundamental diagram

From this diagram if can be extracted two clue values. The maximum flow that the road can carry and its corresponding critical density. The critical density is, conceptually, the density in which the road can carry the maximum amount of flow. When this density is reached, the road becomes unstable and the flow starts decreasing. The peak observed in the diagram perfectly shows it. Moreover, the peak splits the diagram into free flow traffic and congested traffic which is found after the critical density is reached. The next fundamental diagram, flow speed, is described as it follows:

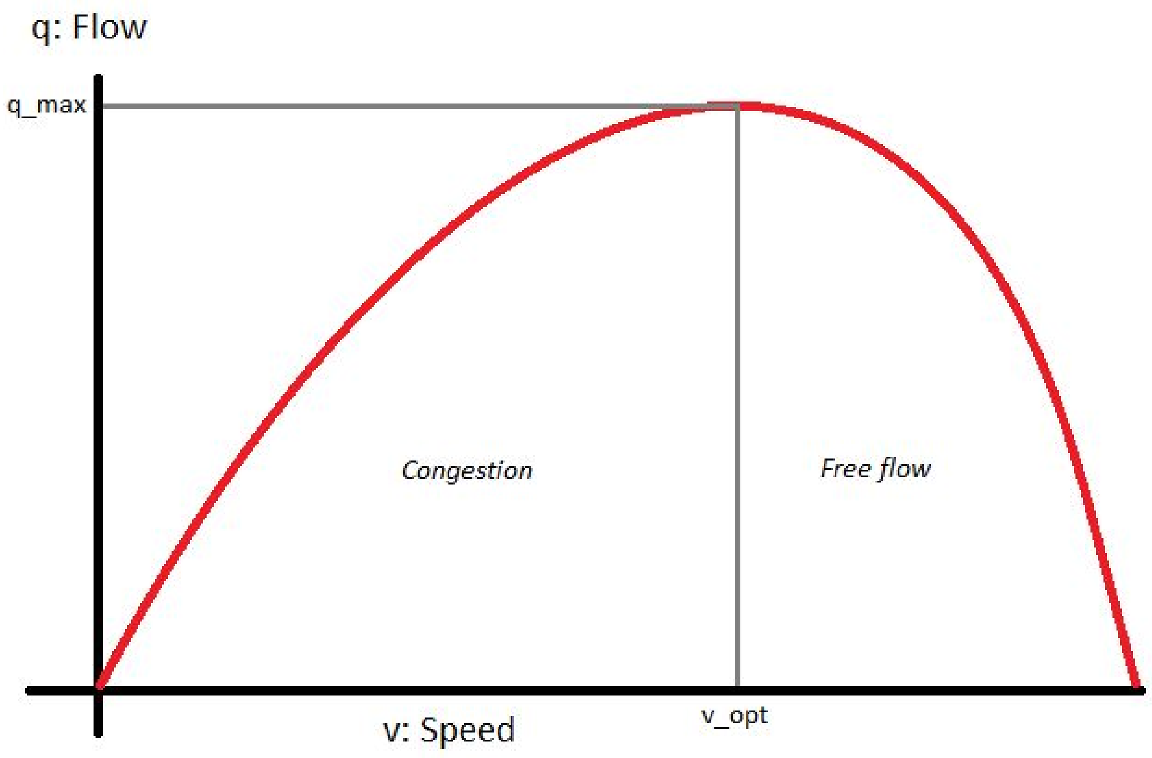


Figure 2.1 flow – speed fundamental diagram

Here, it is again found the maximum flow which coincides with the maximum flow found in the previous diagram. The maximum flow is again found in a peak providing the optimal values. Matching with the maximum flow, in the speed axis can be found the optimal speed which is the speed in which the maximum amount of traffic flow can be carried. The peak splits the diagram again into congested and free flow traffic. Finally, let's look at the last diagram describing the relation between speed and density.

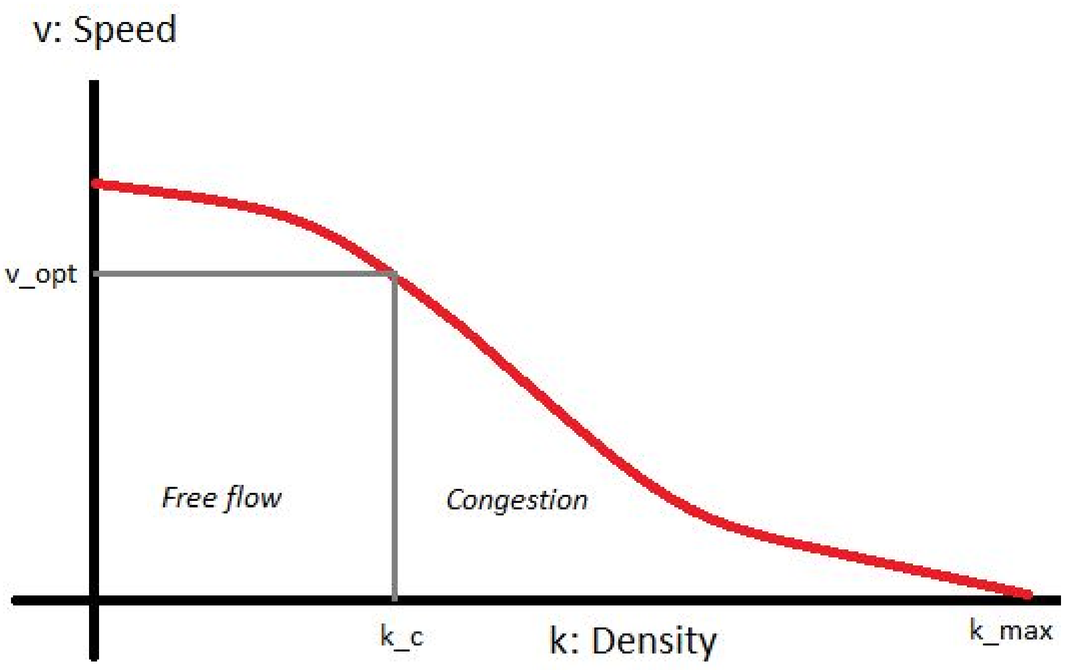


Figure 2.2 speed – density fundamental diagram

In this diagram, no evident peaks can be found. However, it is possible to plot the values previously found. By doing that, the optimal speed and the critical density coincide. Nevertheless, in practice, these two values don't exactly coincide.

c. Mathematical representation of the Utrecht road network

The road network is represented by a graph. This graph is composed by a set of nodes that are the traffic measurement points which can be found in the Dutch roads and a set of edges connecting the nodes.

A Java application, which is run just once, loads the nodes and generates edges for the graph. All the data that this project uses is taken from the same provider (NDW). Check appendix A (data source) for more information about the data source. The measurement point ids and its location coordinates are taken from two different files. One file in an XML format containing references ids, latitude longitude coordinates and configuration details and one other shape file containing spherical coordinates. In the measurement configuration file, there is a relation with the reference id and latitude and longitude coordinates. From this file, the references which coordinates are found in the area of interest (the city of Utrecht) are selected. Then, in the shape file the selection is also made. The reason why this process is done in two steps is because in the XML file the coordinates are in the latitude longitude system. In the other hand, the shape files contain spherical coordinates which are difficult to be processed. Shape files are formatted (.shp) and allow to represent geography objects with a high accuracy. The shape files are loaded using a java library called geotools. For more information about geotools check the appendix B (geotools). To link the nodes QGIS is used. For more information about QGIS check appendix C (QGIS). Now, the roads must be loaded in the QGIS desktop application. The roads are found in a shape file as well. The linking process is done manually. It is the only way to ensure a high accuracy in the node edges. This accuracy is needed later when developing the congestion models. QGIS allows to do these manual tasks easier and faster.

Finally, the loaded nodes and the edges forming the graph are stored in a database (also in the cluster). The database uses a Hadoop technology called Hive. Hive is a non-relational database which distributes the data over the cluster nodes. The reason why hive is chosen is because it is easily supported by the cluster manager. The following picture illustrates the tables which contain the generated graph.

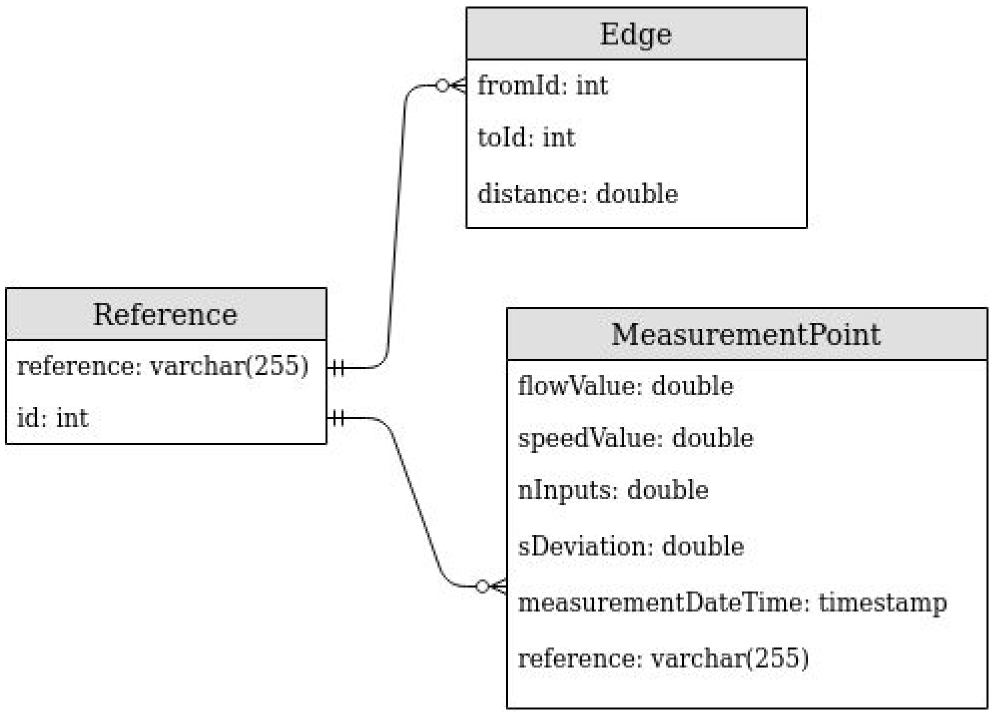


Figure 3.0 database tables containing the road network

The Reference table stores the nodes with its references and ids. The Edge table stores the edges with its node ids and the distance between the nodes forming the links. The MeasurementPoint table stores the traffic data taken from the source and it references to a node in the graph.

To get the traffic data a Java application is run in the cluster updating the database each 5 minutes. The java application runs on spark, another Hadoop technology. Check appendix D for more information about Hive and Spark. The traffic data is taken from another XML file provided by the NDW. The application loads the measurement points from the database and inserts the extracted data using its own data structure.

The traffic data can be either a speed measurement or a traffic flow measurement. On the traffic data file, the measurement points are identified by a reference id which have already been stored in the database. This ids match with a specific location which can also be found in the shapefiles and the configuration XML file. The traffic data from the XML is extracted to the java application data structure. Once the measured data is loaded, it is time to match it with the references found in the database. Finally, the updated data is stored in the database, the MeasurementPoint table, and will be used to do all the later analysis.

The java application which run on the cluster through spark needs to package all the code and its dependencies (managed by maven) into a single jar file. To package the application a maven assembly plugin is used.

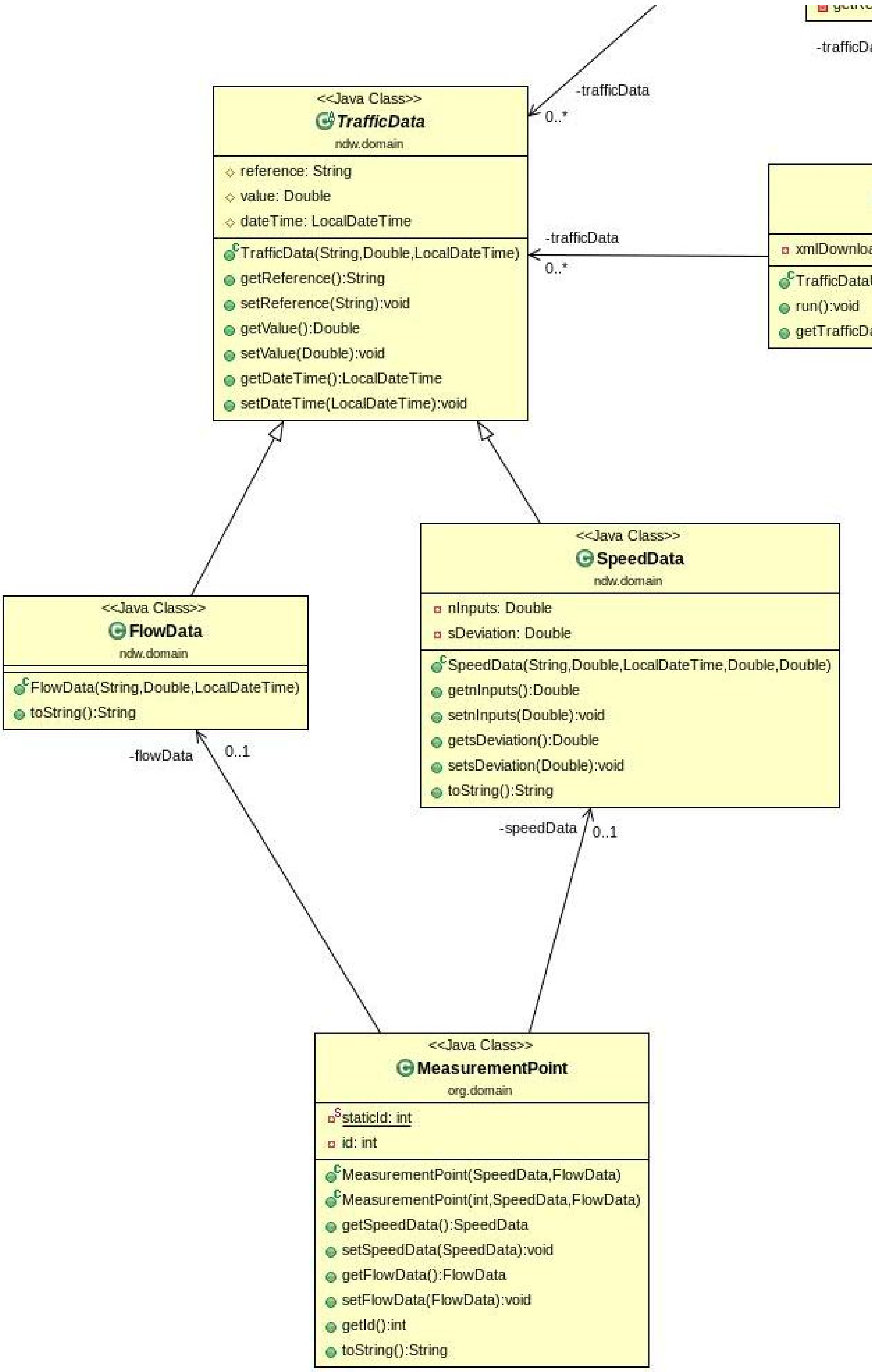


Figure 3.1 class diagram of the data structure used to store the traffic data

As it can be appreciated there exist two kinds of traffic data. One measures the traffic flow and the other the speeds of vehicles. They both have a numeric value associated, but the speed data also includes the number of input vehicles taken to get into that value and may include the standard deviation of that measure. The superclass TrafficData contains also the reference id referencing the measurement point where this data belongs. A TrafficData object is then encapsulated in a MeasurementPoint class.
The data structure used to store several MeasurementPoint objects is a map having the reference id of the measurement point as a key and a the MeasurementPoint object as a value.

To extract the data in these data structure the following classes are used:

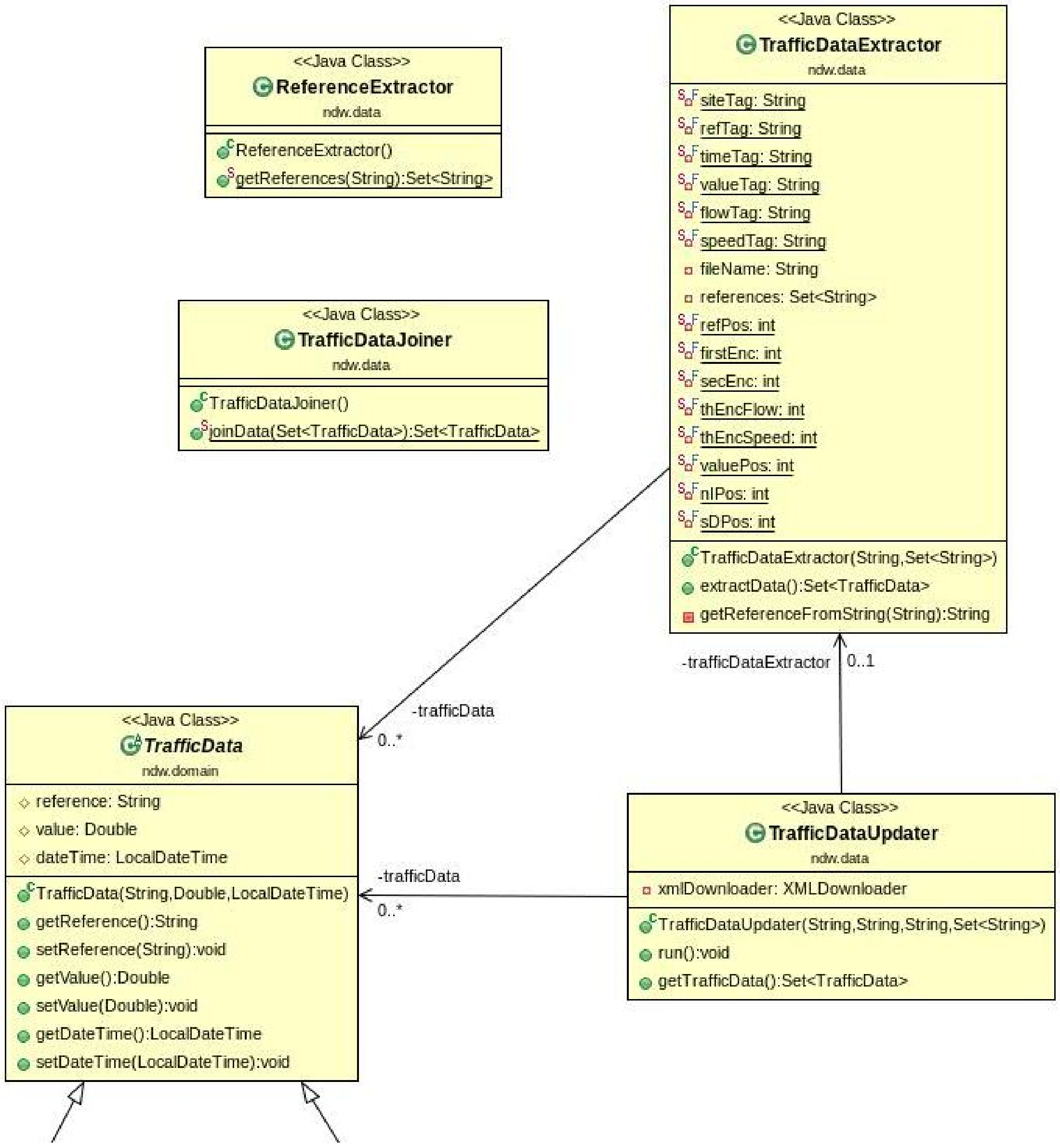


Figure 3.2 data extractor classes

The TrafficDataExtractor class is the one responsible of extracting all the traffic data records from the XML file returning a set of traffic data objects. The ReferenceExtractor does the same job but for the reference ids of the measurement points. The TrafficDataUpdater class is responsible of downloading a new XML file with updated data and making the TrafficDataExtractor run to get all the needed records. Finally, the TrafficDataJoiner class provides a single static method which receives as an input a set of TrafficData objects and makes averages and additions for the repeated measurement points. The reason why these averages and additions are made is because the national data bank provides traffic data for different vehicle types depending on its size. The flow values are added while the speeds are collected in an average. In this study case it is only interesting to look at all kind of vehicle that crosses the measurement point. The macroscopic approach of the analysis considers the vehicles as a mass which behave as a flow.

The Java application which extracts the initial data to generate the graph with nodes and edges have the following structure:

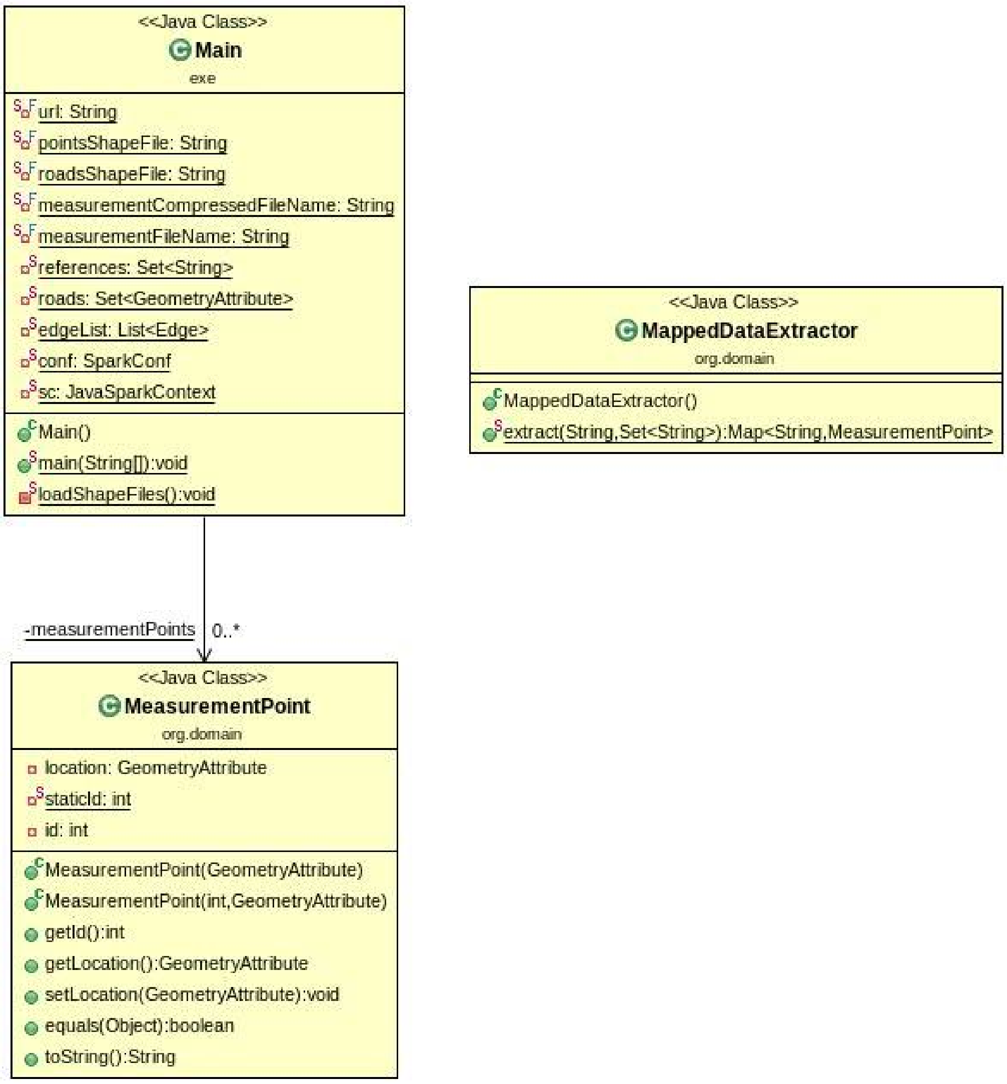


Figure 3.3 classes which extract the initial data from the road network

In this case the MeasurementPoint class stores a GeometryAttribute object which contains the exact location of the point. The MappedDataExtractor is a utility class which maps the references selected from the XML configuration file with the GeometryAttribute objects from the shapefiles. The MeasurementPoint objects are again stored in a map as a value matched with the reference id of the point as a key.

d. Road network analysis

The network analysis allows to have an initial overview of the road network structure. For example, by looking at how many edges has a node, how common is to find a branch in it, which are the most relevant points in the network etc. The generated graph based on the Utrecht road network will be analysed by considering the following concepts.

- Degree distribution (sorted sequence of node degrees).

- Centrality:
    - Degree centrality (fraction of nodes in which a node is connected to).
    - Closeness centrality (reciprocal sum of the shortest path distances from the node to all the other nodes).
    - Betweenness centrality (sum of the fraction of all pair shortest path).

- Density (measures the number of edges per node. A dense graph would have a high number of edges in each node reaching each other node).

- Clustering (measures the number of triangles in the network).

The graph analysis will be conducted using NetworkX a python library for the network analysis. For more information about it check the appendix D (NetworkX). How the analysis will be displayed will be approached later in the visualisation part.

The degree distribution is a simple way to get a first overview of the graph structure. Is a representation of a sorted list of the degrees of the nodes which can be found in the graph. It is usually represented in a form of a scatter plot.

Degree centrality is used to detect central nodes in terms of spreading information and influencing the immediate neighbourhood. It is a measure of centrality given by the node degrees. It is defined by the number of links incident over a node. I allow to see which nodes are more connected to other ones.

Closeness centrality is the average length of all shortest paths from a node to all other nodes, it appreciates the time or speed to reach other nodes from a starting position. It represents an average length between the node and all other nodes in the graph. It shows which nodes are closer in terms of distance to other ones.

Betweenness centrality measures the number of times that a node acts as a bridge for a shortest path. In this study case it will be the most relevant measure because it will show the most concurred points of the network. It helps detecting places where the network concentrates a significant part of its traffic activity.

Clustering measures the number of full connected components formed by three nodes (triangles). It is used to analyse subcomponents in the graph. If there are many subcomponents formed, it indicates that the network is organized by connected groups. These groups may play a certain role in this study case and it will be deduced.

e. Statistical analysis

The statistical analysis will be performed at the individual measurement points or by comparing two of them. Each measurement point will have a statistical analysis associated. The statistical analysis will describe the traffic flow and the vehicle speed average on a certain node over time and will provide a statistical summary as well. Several charts to visualize the results will be displayed. Line charts and box plots are the preferred ones. The goal of this section is to get a general overview of the traffic data. By looking at measures of centrality and dispersion it can be seen how the values oscillate during the day. Besides, by correlating data overtime, patterns in it can be detected and classified. Combining the detected patterns with the measures of dispersion obtained it can be seen at which time of the day the data take extreme values. The noise of the extracted data can be calculated as well. Moreover, it will be possible to compare and correlate two measurement points. This is helpful to see differences in their data and extract conclusions from them. However, the extracted data might contain some wrong values (zeros nulls or minus ones). These wrong values must be cleaned first in order to not make them influence the analysis results. The cleaning process is explained in the appendix F (data cleaning).

The statistical analysis for every single point will consist of:

- Numeric summarization of data
  - Measures of central tendencies
  - Measures of dispersion

- Time series analysis
  - Additive decomposition for traffic flows
  - Additive decomposition for traffic speeds

    - Additive decomposition for densities
- Regression and comparison between measurement points
    - On traffic flows
    - On traffic speeds
    - On densities

All these statistical results will be complemented with different plots which will illustrate the numeric results. The statistical analysis will be conducted using statistical analysis python libraries. For more information check the appendix F (python libraries).

f. Modelling

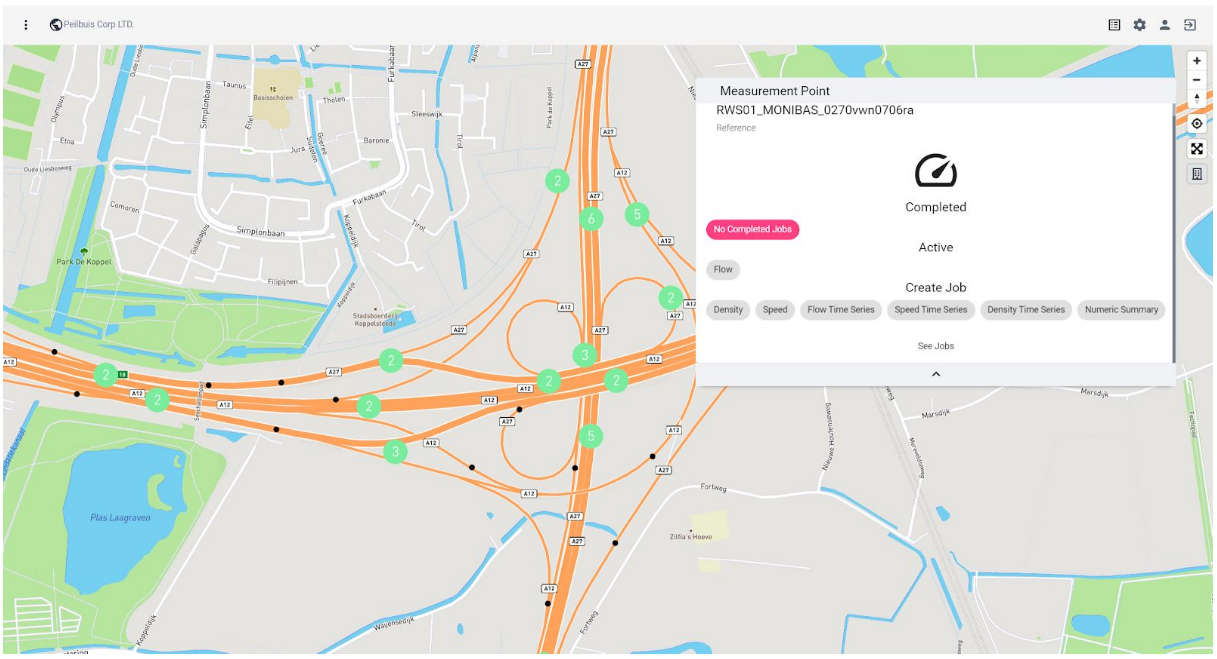
The main goal approached for the modelling in this study case is to predict the impact of a congestion in one or more points by redirecting its traffic flow. For that it is needed to set up a basis. It is needed a prediction model to estimate the traffic flow at certain point at a certain time of the day and a prediction model to estimate the traffic density taking the same parameters. Moreover, a capacity of the road derivative from a calculated critical density and an optimal speed indicating the speed which can carry the maximum amount of traffic. To accomplish that the traffic flow theories must be put in practice. These optimal values derived from the traffic flow theory are calculated by making regressions with the main traffic data components. The regressions are made using specific python libraries explained at appendix E as well.

g. Visualization of results

Finally, the obtained results must be visualized in order to ensure that the analysis is understandable for the public. When doing data visualization, in this project, is assumed that the public have a basic idea on how to interpret charts and plots and have a basic knowledge on statistical analysis. However, is part of this project to make the results understandable with the corresponding explanations.

The methodology used to visualize the results is a subject of research. The tools used for the data visualization must provide a clear graphic layout. For traffic data the chart needed usually are scatter plots, line charts and scatter plots combined with line charts. Then, for statistical analysis box plots are be powerful as well. For network analysis bar charts as histograms indicating frequencies. Despite the traffic data, the network itself which must be represented as well. For that, a map interface might be used with its measurement point around the city area. For a detailed version of it, the shape files might be used and visualized by a *gis* tool.

A goal of this project is to make the visualization interactive. This would be done with a web application which would contain the map and the measurement points in the network. Once a point is clicked, a few options are displayed (each one representing a data visualization service). In this project, the front-end isn't developed (it is done by another development process). Nevertheless, a back end *api* is provided to enable communication from the database to the client.



7.0 web application screenshot

Moreover, as the front-end follows another development line, some other data is needed to visualize results and perform analysis. To do that, *jupyter* notebooks are useful tool for coding and visualizing results. They allow to plot results in charts by using python libraries such as matplotlib. Notebooks are structured by cells where you can place your code and run it individually and separately. Once a cell is run, the run time data is saved in memory. So, it enables to run other parts of the script without running all again. For large calculations, this can save a lot of executing time. The charts provided by this methodology can be easily customized and enables to provide clear results.

To visualize the network without using the web application, and to see more in detail how the nodes are linked, QGIS is used. QGIS is a shape visualization tool which allows to visualize shapefiles and interact with them. It provides extra tools to measure distances and load shapes from databases. The following picture illustrates how the QGIS performs.

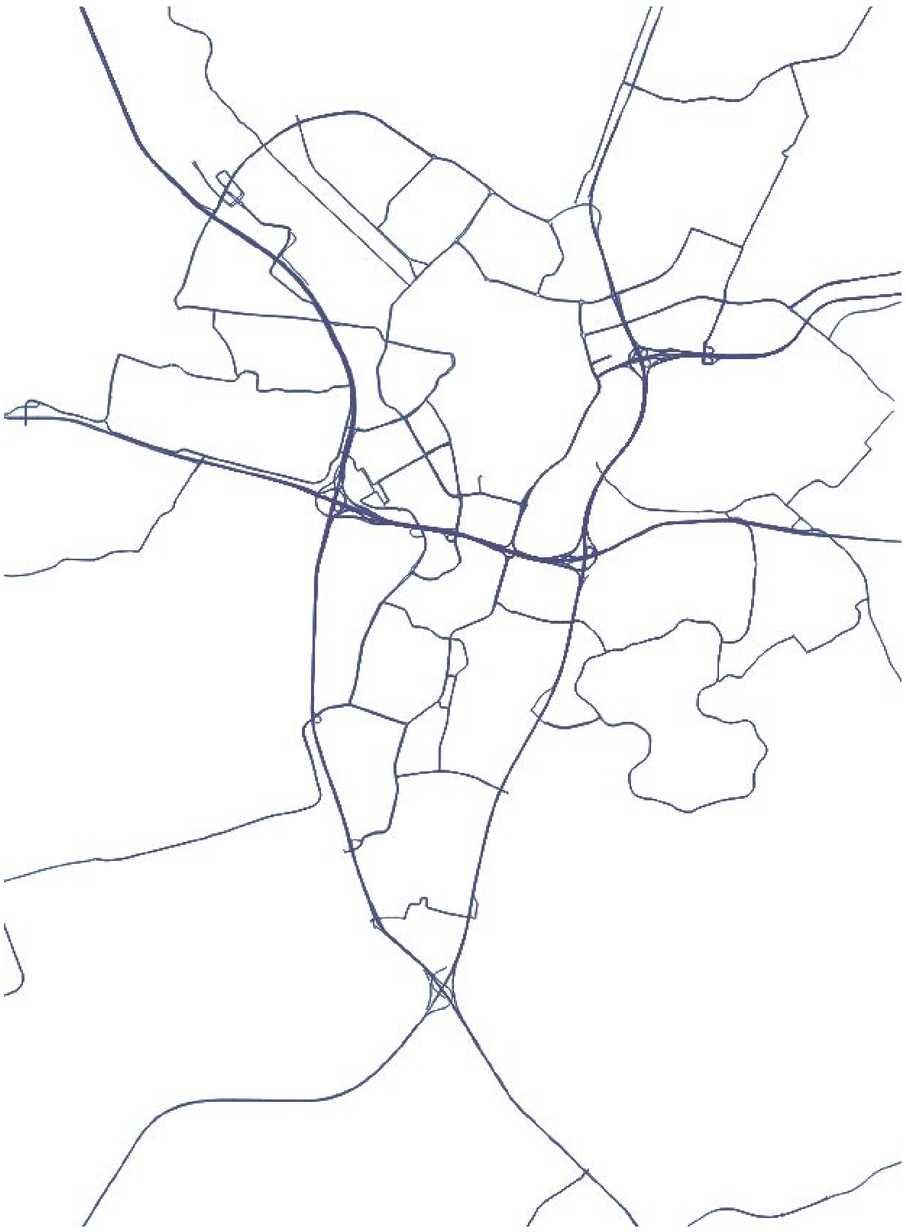


Figure 7.1 roads on the city area of Utrecht

The figure 7.1 shows how QGIS represents the shape files. Now, to see more in detail how the edges of the network are generated let's take a closer look to a section of the network.

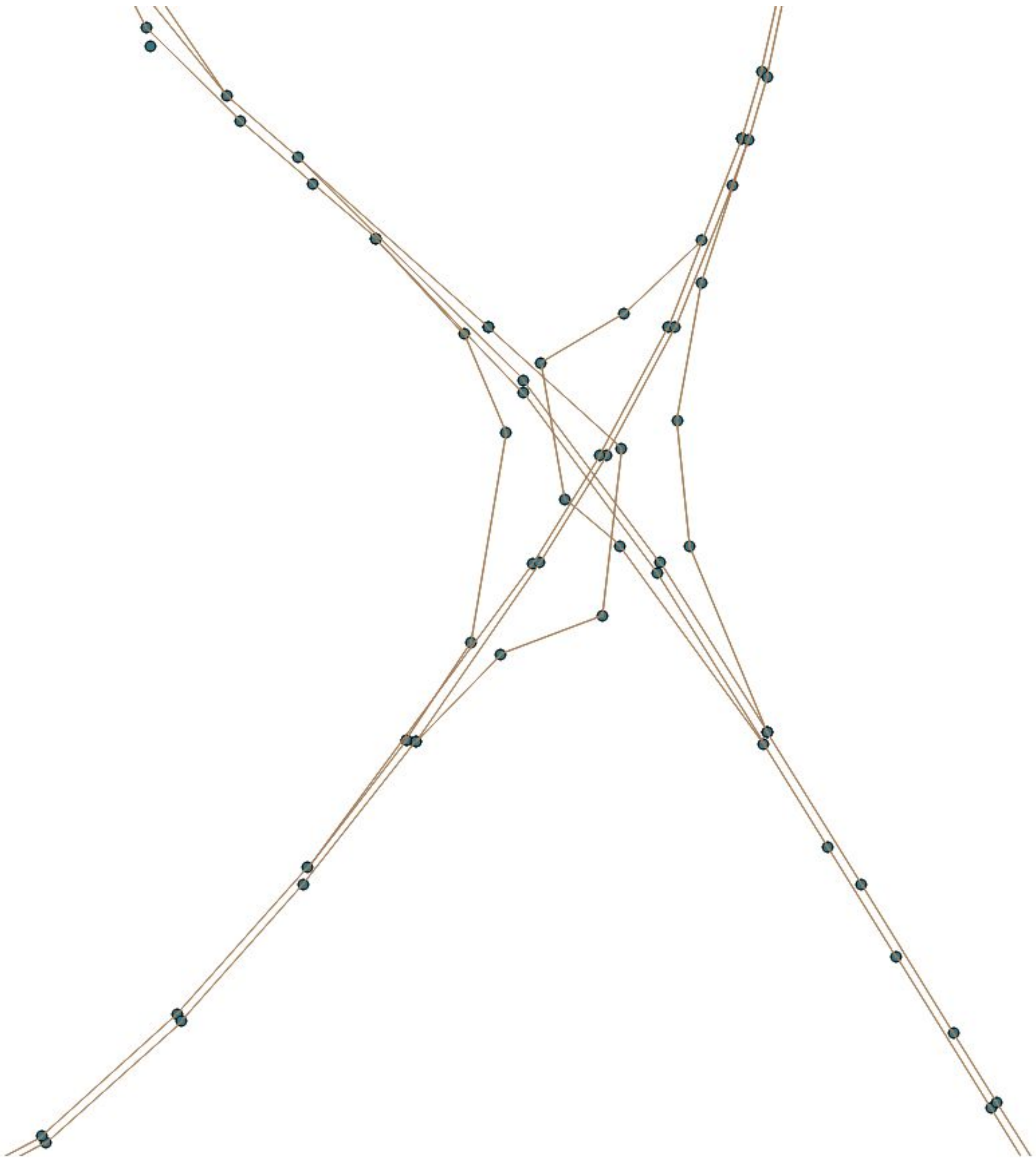


Figure 7.2 network at a section in the city area of Utrecht

At the figure 7.2 it can be seen how the network is generated by linking the measurement points. Note that here the road are straight lines because the simply represent connections between nodes.

5. Analysis

a. Network analysis

The conducted network analysis has determined the following results.
The degree distribution of the network, which is illustrated by the scatter chart, shows that most of the measurement points of the road network have degree 2. These points have no branches and contain an input and an output edge. Usually are middle points of a road segment. Then, there are some nodes with

degree 3. These ones are either creating a new branch in the road or merging one. There are also some nodes with degree 1. These are entrance or exit points in the road network. Finally, there exist a few points with degree 4 and 5. These ones could be considered as distributing points (hubs) in the road network. They used to be central spots of the network.

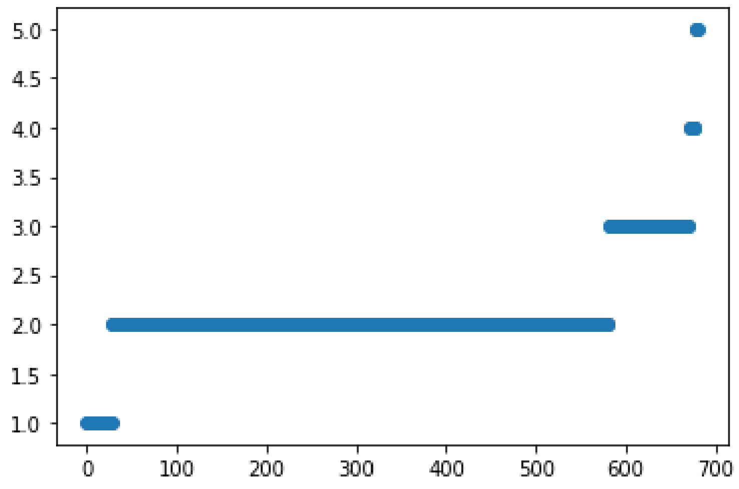


Figure 8.0 degree distribution scatter plot

The centrality analysis was conducted looking at three different centrality types. The degree centrality, which was computed, indicates the node centrality by looking at the degree. It showed the following histogram representing the centrality value frequencies:

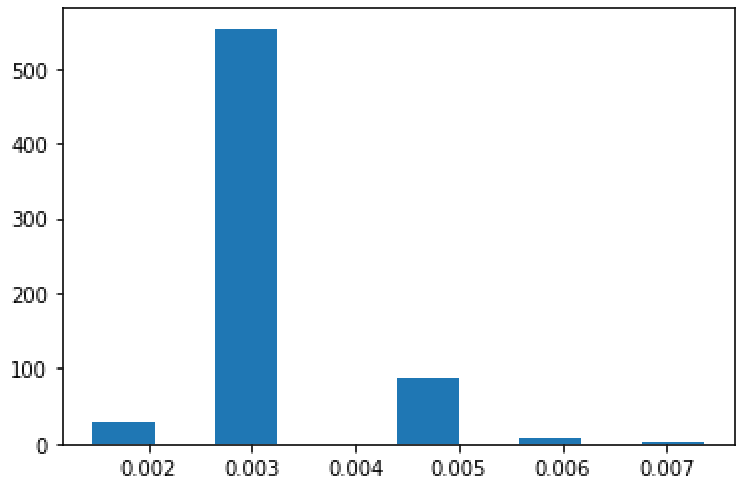


Figure 8.1 degree centrality histogram

This shape shows again that there are just a few nodes which are highly central compared to the others (which are the majority). These central nodes are interesting for later statistical analysis and traffic simulations due to the impact on the traffic flow of the whole network that certain situations directly affecting these nodes might affect also to the rest of the network. There are a few nodes detected which have a higher centrality than most of the nodes, but it isn’t as big as the mentioned ones. These nodes are found in an intermediate level of traffic distribution (between regular nodes and central nodes).

Another centrality which was computed was the closeness centrality. It is a reciprocal of the average shortest path distance to u over all n-1 reachable nodes. The results here differ a bit when are represented in the histogram.

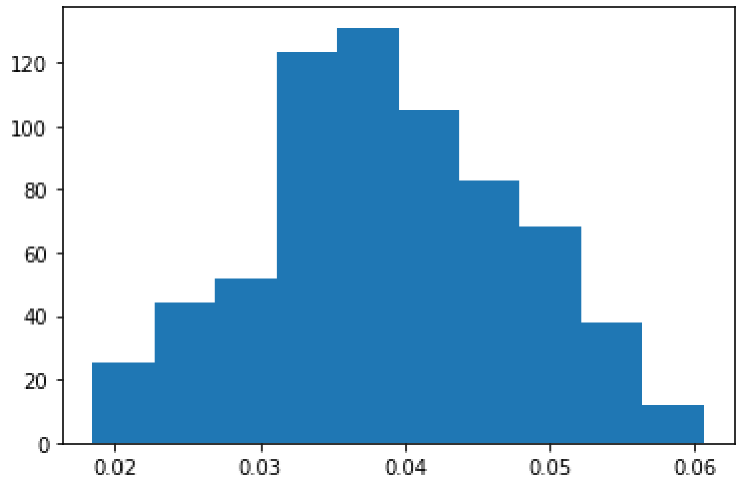


Figure 8.2 closeness centrality histogram

The centralities taken by the nodes, most often are between 0.03 and 0.04 (approximately middle values). However, there are more extreme values taking major centralities. From the definition of closeness centrality previously explained at the section 4.d (results, road network analysis, closeness centrality) this indicates that all the measurement points are close to each other following a normal distribution shaped histogram.

The last centrality analysed, the betweenness centrality, results into an exponentially decreasing form in its histogram. It shows a big difference between the most central nodes and the less central nodes in terms of the frequency in which the centrality values appear. This means that just a few points are really concurred. The road network might have a big number of highways or roads which can carry many vehicles and have no distribution purposes, and a small number of roads which connect to these main roads and have distribution purposes. This last type of road might be the most concurred roads if we later look at the traffic flow at these locations. In this project is mainly analysed the behaviour of those main roads which can carry a lot of traffic and how the distributing roads can redirect big traffic flows or congestions.

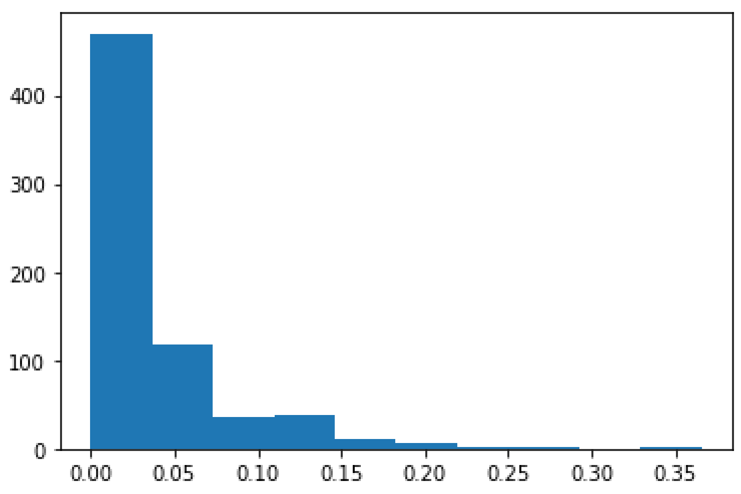


Figure 8.3 betweenness centrality histogram

Finally, the network analysis includes a clustering analysis to find triangles and connected component within the network. This part of the analysis was designed to detect possible connected sections and organized node groups. However, the algorithm used by networkx couldn't find any relevant information from it. This means that this network has no strong connected components or clusters in it. It is reasonable because a road network must carry traffic and not cycle it. That is the reason why no small clusters were found. The network has traffic redirection alternatives using these nodes which have degree 3 and more. The parts of the network which have less connections might have more difficulties to redirect traffic in case of congestion.

b. Statistical analysis

The statistical analysis was done taking a single node as a subject of analysis. Each node can have a dedicated statistical analysis associated. This statistical analysis can be presented as a report when visualizing the results. The report includes a small statistical summary showing the mean and standard deviation of each of the components forming the traffic data. It also includes a time series decomposition for each of the component. The time series allow to visualize the data decomposed in three components. The trend, the seasonality and the residual. This way of representing the data clearly shows the behaviour of the traffic flow during the week and detects possible anomalies. To complement this information, box plots are displayed as well. They are used to check the dispersion of the extracted data. The extreme value on them might have an important meaning. The analysis programming has been made general in order to easily compute any node.

Now let's see the results obtained for the node with reference id: RWS01_MONIBAS_0271hrl0783ra

Let’s take the statistics which apply for a week of measured traffic data. From the day 27-01-2020 at 00:00 to 02-02-2020 at 23:59. The time intervals are 5 minutes for each measurement record. The first information taken from the statistical analysis of that node is a small numeric summary containing the mean and the standard deviation. For the flow data the information obtained is:

flow mean: 4222.509517008501
flow standard deviation: 3076.3937771069377

The standard deviation is high compared with the mean. This means that the data taken is really disperse. It makes sense because the traffic activity is different during the night-time than from the rush hour. This high dispersion indicates that during the day, at the studied point, the number of cars that it goes through is considerably changing. The boxplot is a good tool to visualize the dispersion of the set of data. The following picture represents the boxplot obtained for the flow data of the studied point:

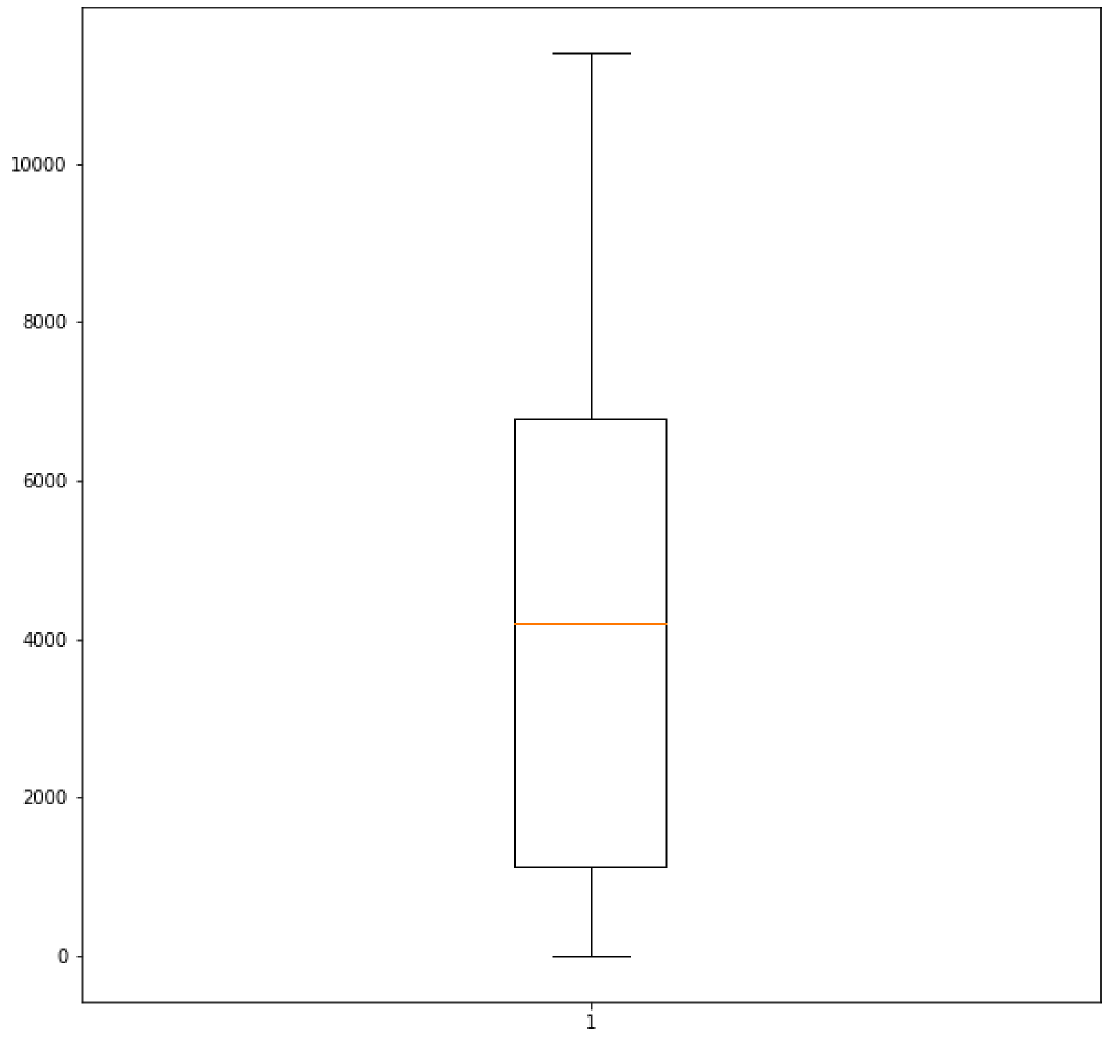


Figure 9.0 boxplot of the flow data

The most disperse values are found above the mean. This means that higher measured flows (rush hour) aren't frequently detected compared with average measured flows. So, it can be said that during a day there are a few traffic flows peaks. The plot tells also that the measured values which are close to the mean, are still disperse. This can be seen by looking at the dimensions of the main box. The measured flows during the night-time makes the average decrease.

After getting some results which gives an overview of the data set, it's time to start building the time series.

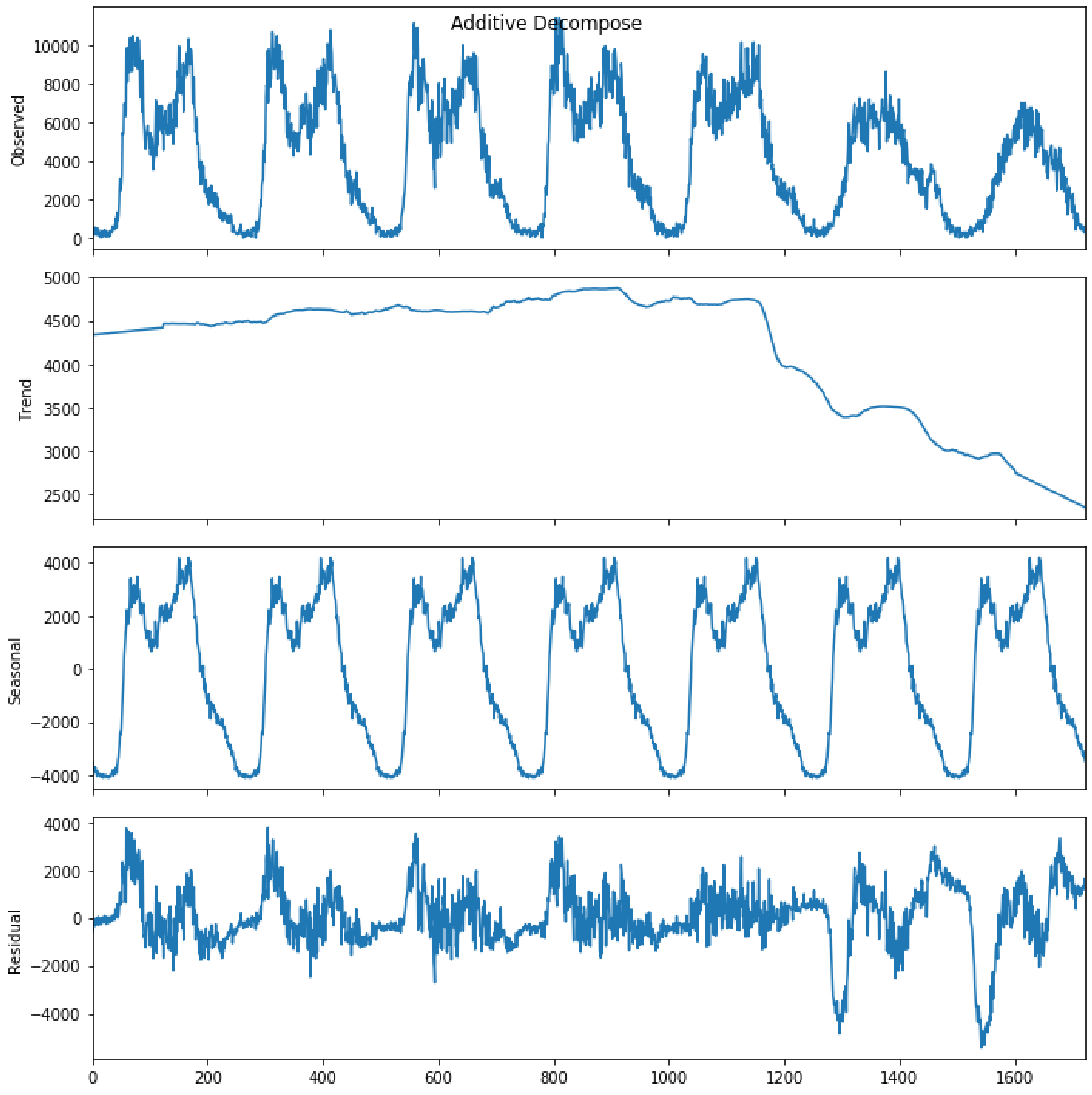


Figure 9.1 time series decomposition for flow data

The figure 5.1 shows the time series decomposition of a week of traffic flow data. The first line chart represents the traffic flow data as it is. Then, it is followed by three other plots which represent each component of the time series. First, the trend components describe the trend of the measured flow over the week. It is calculated from a moving average. As it can be seen, it keeps stable during the working days, and then it suddenly decreases at the weekend. Then, the seasonal component indicates the variation during the time intervals. The intervals are previously fixed and take one day of data. This component indicates the pattern followed by the data. Finally, the residual shows the error of the estimated values compared with the real values. It can be observed that the residual is greater during the weekend when the pattern isn't followed well. The time series are useful to observe any trends and patterns in our data. It can be seen a pattern from the traffic flow evolution during a day. During the night-time there the flow is very low, or it is 0. Then suddenly increases in the morning. Around 6 a.m. There is a peak around 8 a.m., a decrease until 12 a.m., another increase with a peak around 4:30 p.m., and a prolonged decrease until 12 p.m.

If the same technique is applied for the average speed data, it is obtained:

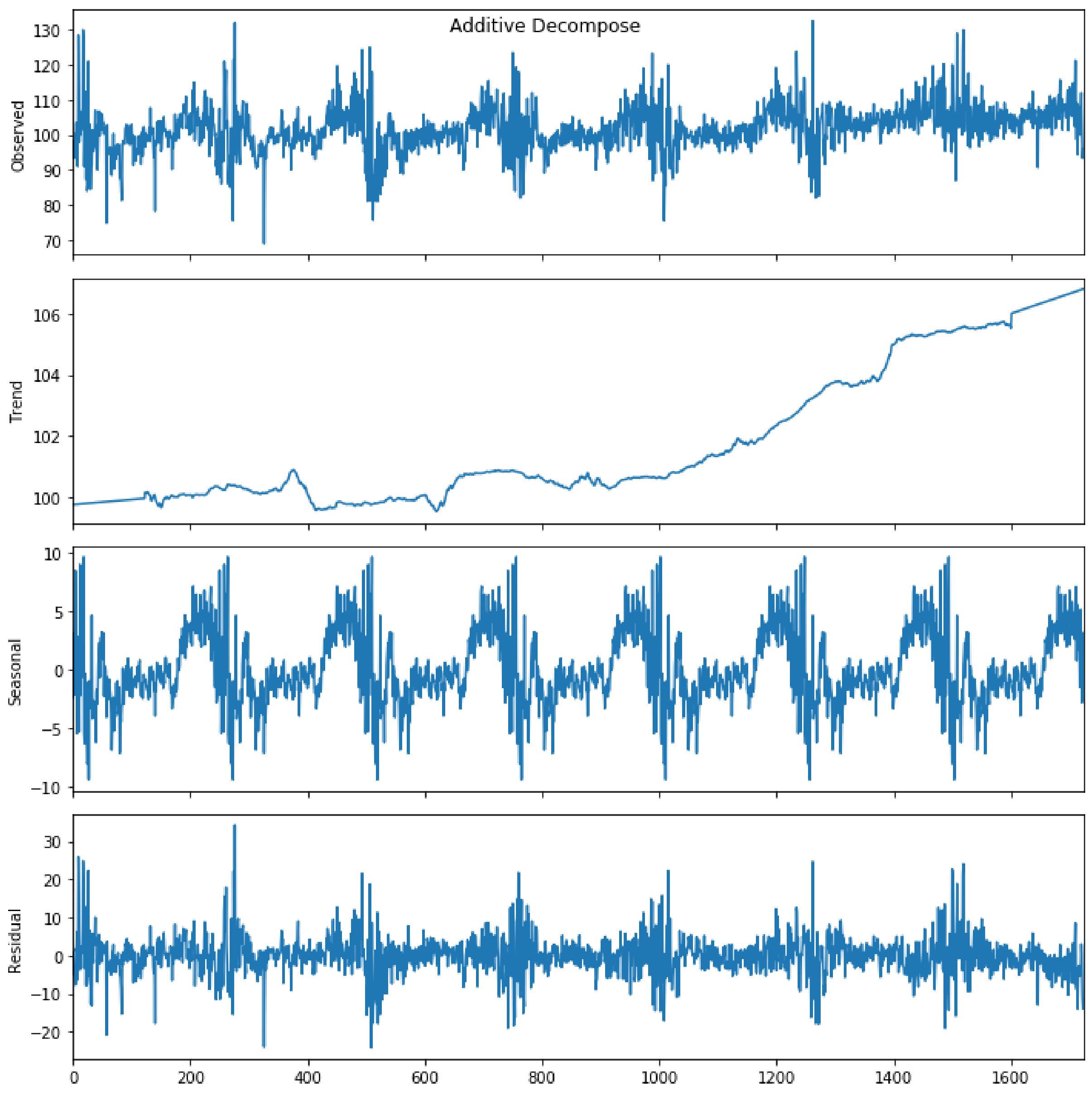


Figure 9.2 time series decomposition for speed data

This time series decomposition shows how the trend increases at the weekend due to less congestion in the roads. In the other hand, there is a visible pattern as well. This pattern indicates the congestion periods (when the speed decreases due to a congestion). The seasonal component clearly indicates this. Finally, looking at the residual component, it is seen when the speed data is noisier. Usually this happens during the nighttime.

To complement the time series analysis for the speed data, its box plot can be observed.

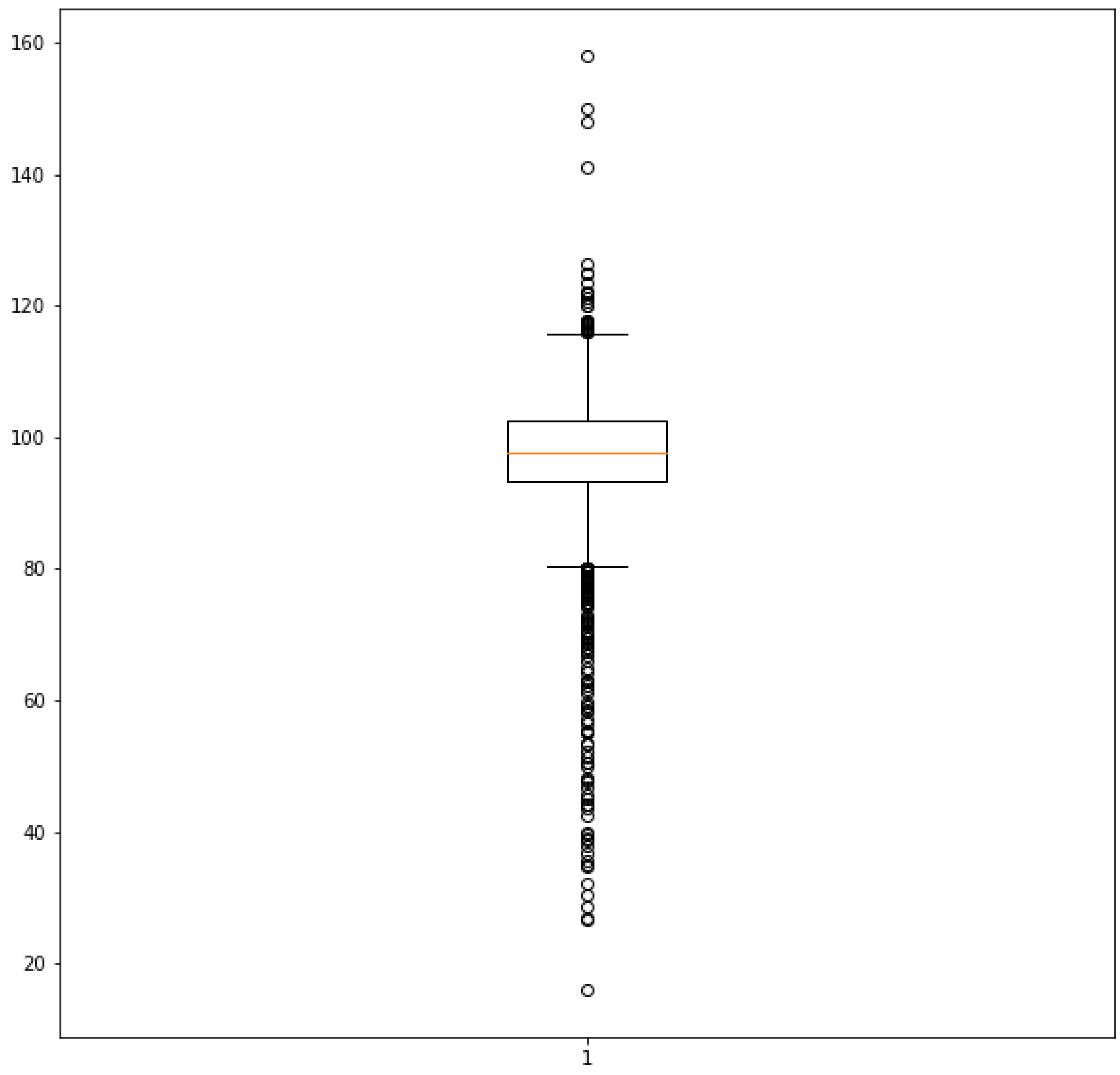


Figure 9.3 boxplot of the speed data

With the statistical values:

mean: 96.39925816023747
sDeviation: 11.9610351062676

This box plot represents a dataset containing a high number of extreme values. The main box with the data belonging to the central quantiles is short compared with the box plot previously commented for the traffic flow. The central data here isn't disperse. Most of the values are close from each other. The small standard deviation shows it as well. In the other hand, there is a high amount of extreme values. These values are dispersed. The extreme values located below the box are associated to traffic congestions while the extreme values located above the box are associated to night-time measurements.

The time series analysis for the density calculated in the road is:

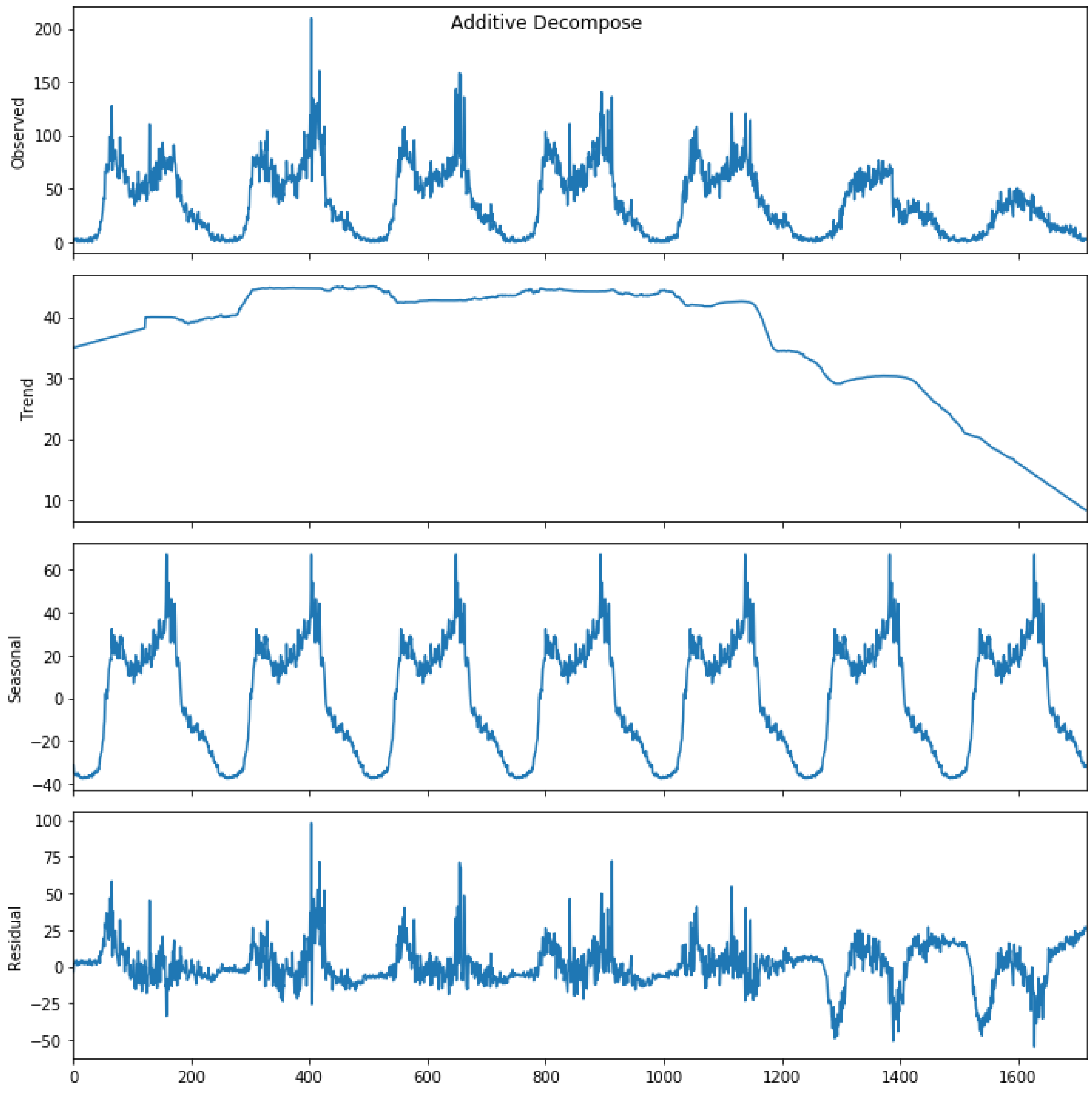


Figure 9.4 time series decomposition for calculated densities

The density time series analysis is like the time series analysis done for the flow. The reason why this happens is because they both measure the number of vehicles. However, the density can better represent a congestion on the road because is a direct way to determine them. Note that the shape of the density line chart during the weekend is different than the shape of the density line chart during on the working days. In the weekend there are no visible peaks. The density data is complemented by the box plot:

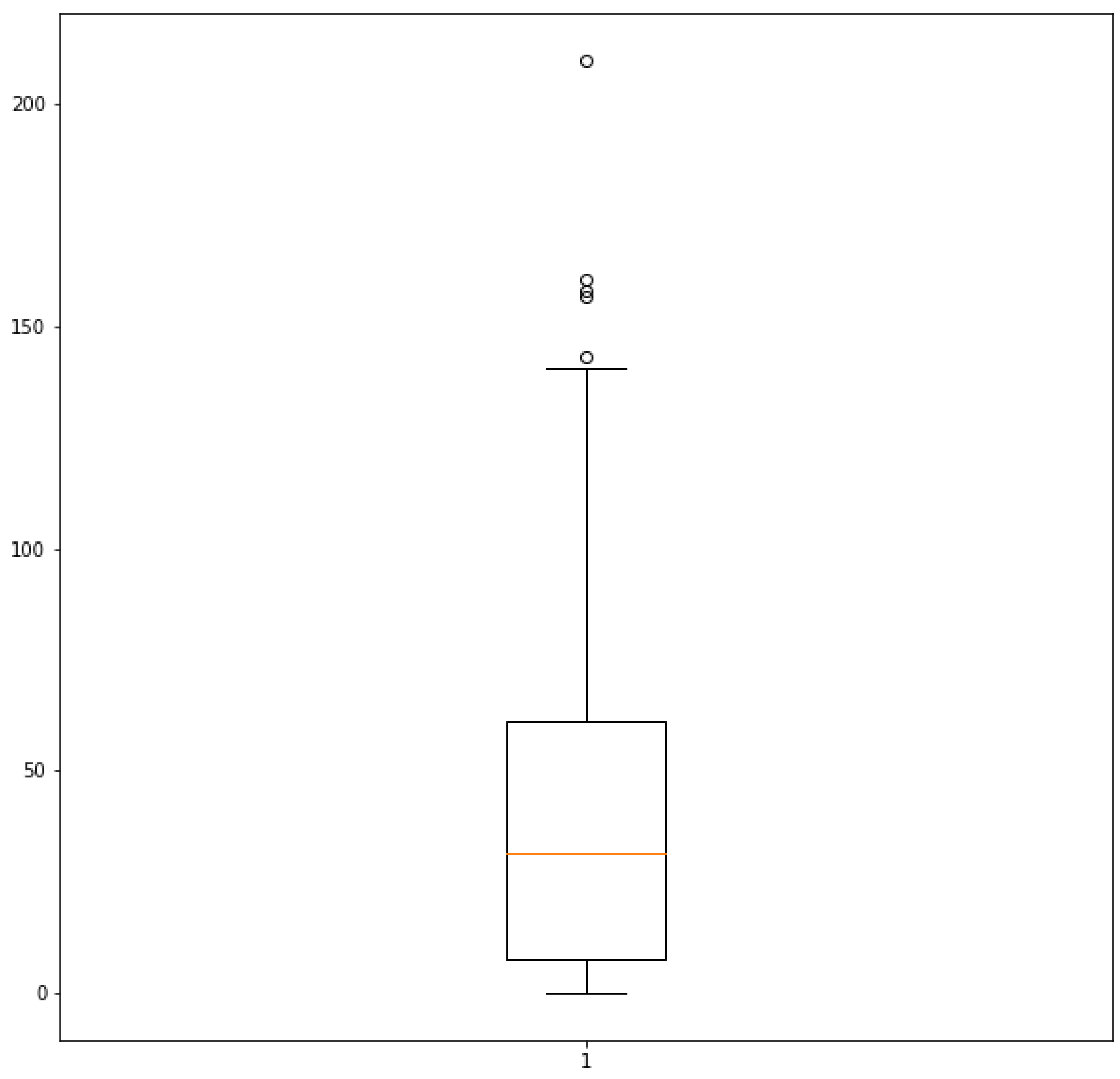


Figure 9.5 boxplot of the calculated densities

With the statistical values:

density mean: 37.263286867124336
density standard deviation: 31.099484473591474

The boxplot for the calculated densities is also similar from the flow data. In this case there are a few more extreme values detected above the mean. These values correspond to high density measurements which could cause congestion.

Statistical comparison between two measurement points

Another statistical application on traffic data is comparing two measurements. This comparison shows the difference on traffic activity between two points. As an example, the point with reference id *RWS01_MONIBAS_0021hrr0535ra* and *RWS01_MONIBAS_0020vwt0651ra.* The displayed data is taken from a day of traffic activity (the 03-02-2020).

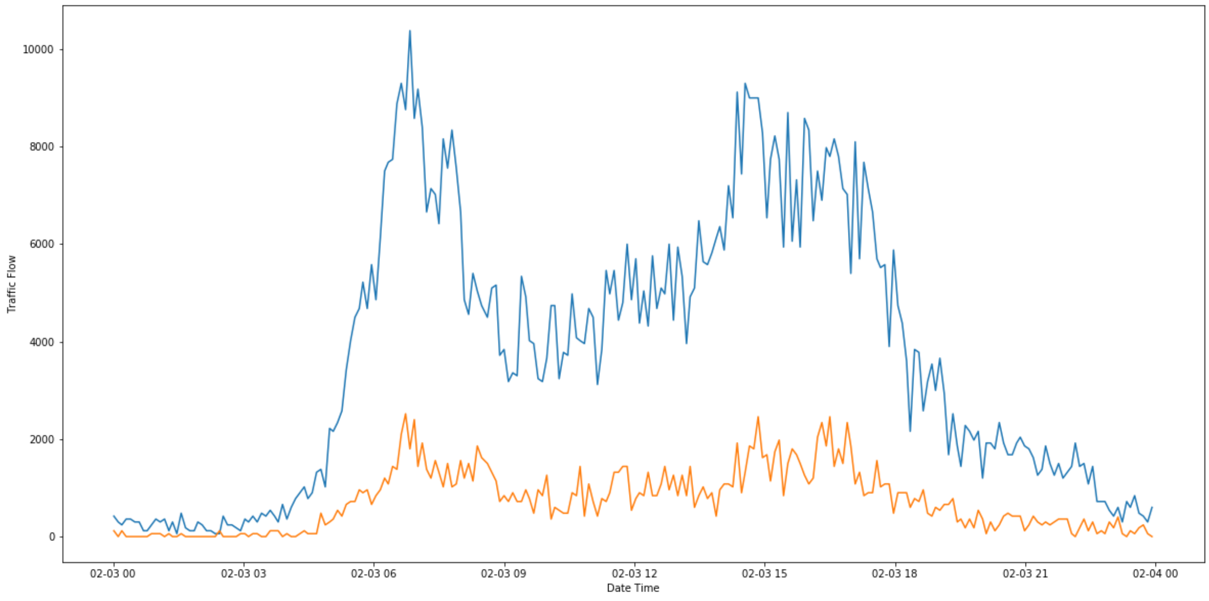


Figure 9.6 flow line charts of two points

The figure 9.6 plots two-line charts each containing the measured traffic flow during the day. It is clearly seen that one measurement point has much more traffic activity than the other one. The same comparison can be done for the traffic density as well.

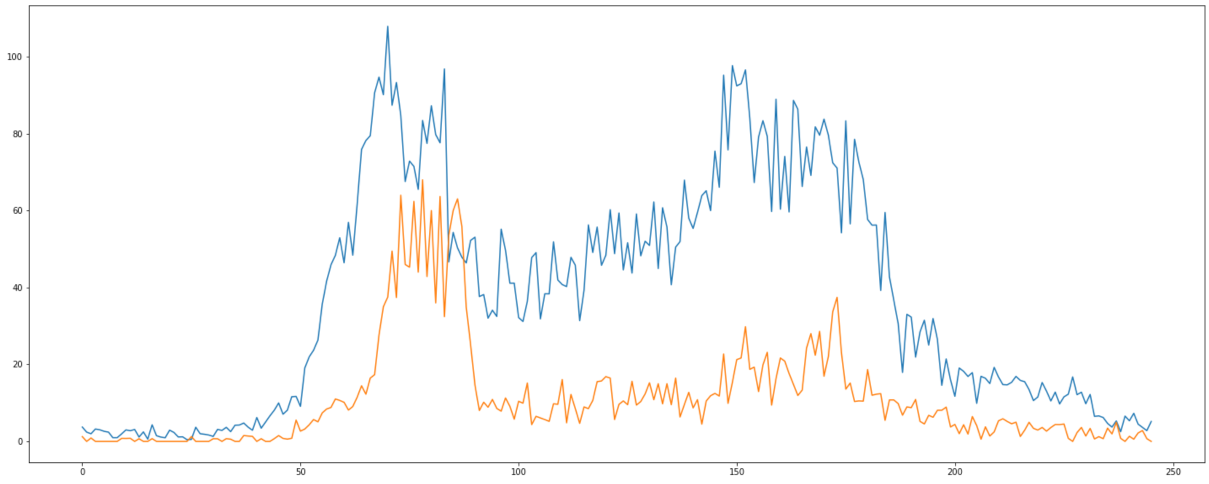


Figure 9.7 density line charts of two points

In this plot is seen again that one measurement point has less traffic activity than the other one. However, let's put attention on the first peak in the morning. There, the difference between both values is not as big as it is on the previous plot (the flow comparison). At a certain moment the density gets even bigger. This means that, even if there are more vehicles going through the first measurement point (blue line), the second measurement point (orange line) gets denser at the morning peak.

To corroborate this analysis of the situation, the speed line charts show the congestion a decrease on the speed.

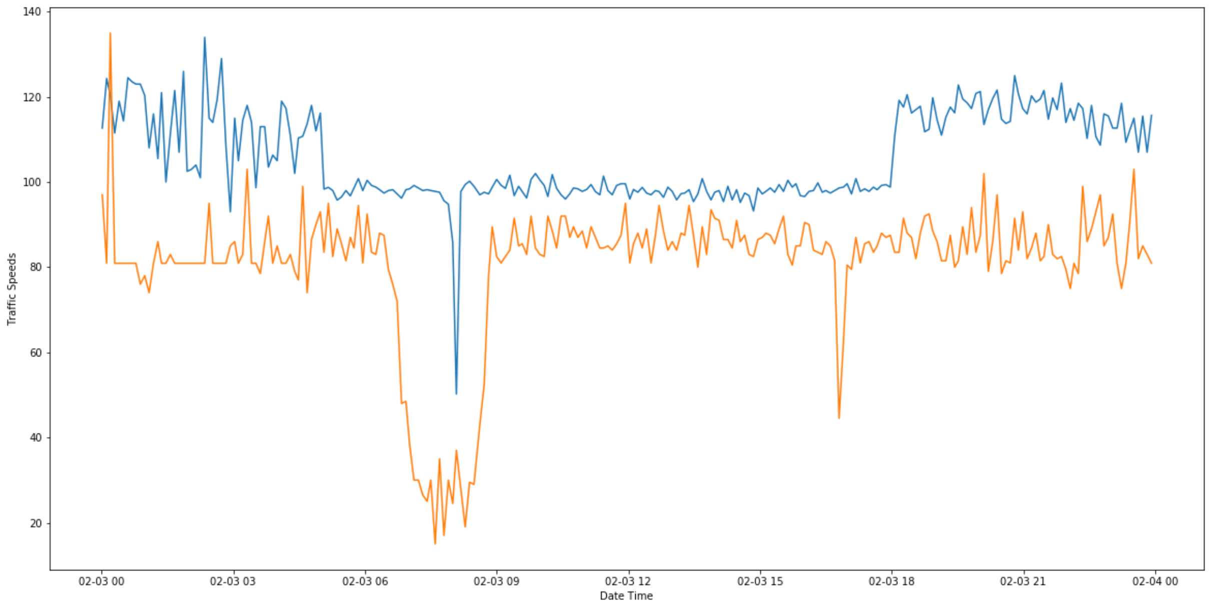


Figure 9.8 speed line charts of two points

The observed speed decreases are due to a congestion because they’re generated during a high-density period. The orange line represents a measurement point where the average speed is lower than the other. The orange line corresponds to a measurement point located in a road where the speed limit is less than the measurement point represented by the blue line. However, if just the congestion period is strictly analysed, it is seen that it is longer for the orange line that for the blue line.

Another way to compare these two measurement points is by looking at the box plots. The following box plots represent the traffic flow data.

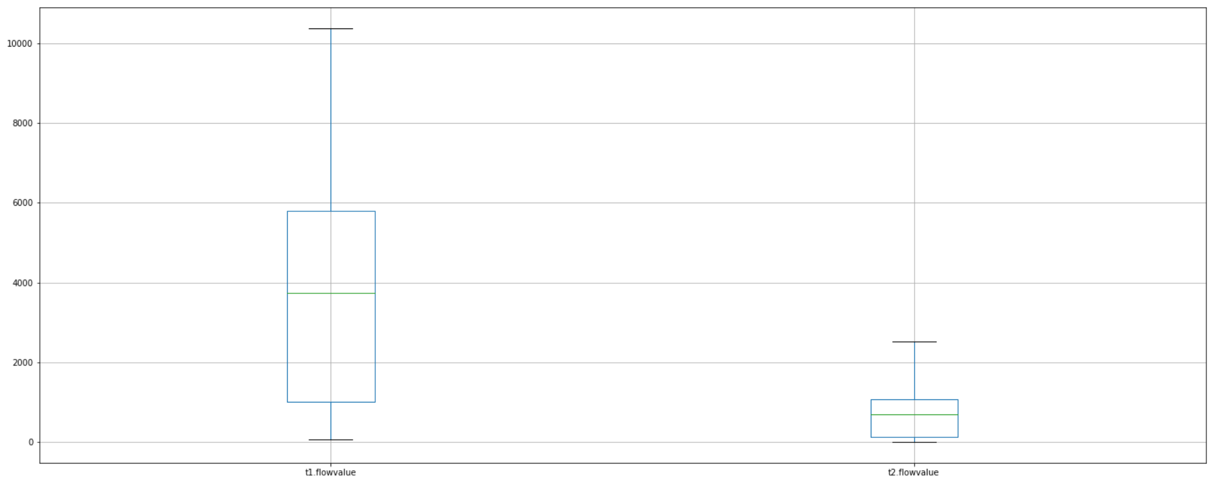


Figure 9.9 flow comparison by box plot

By looking at these two boxes plots it is clear which one is the point which carries more traffic flow. The one in the left (blue line for the line charts), has bigger dimensions which implies bigger flow average but bigger dispersion in its data as well. The box plots for the calculated densities are:

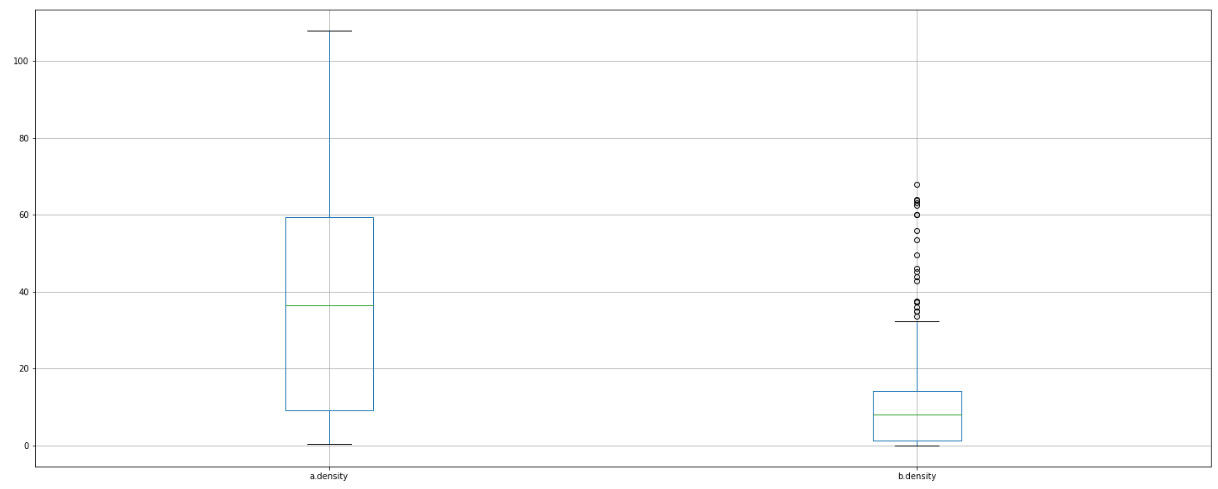


Figure 9.10 density comparison by box plot

The box plots here are like the ones done for the traffic flows. In this case it is seen that the right box plot has more disperse data on top of the box. This indicates that the high densities there are more dispersed than in the other measurement point. If they are more disperse it means that they are far from the mean as well. The box plots for the average speeds are:

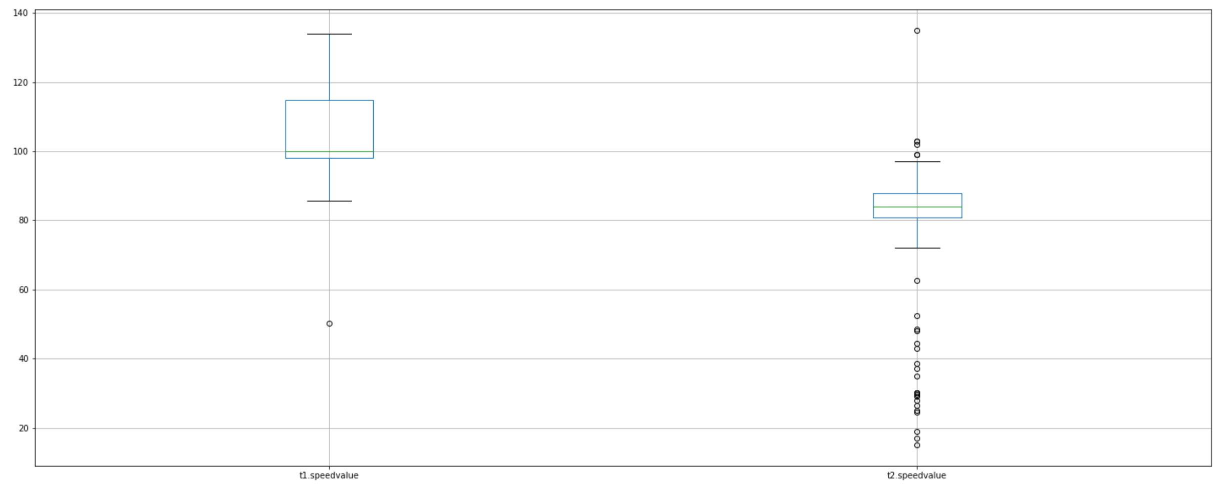


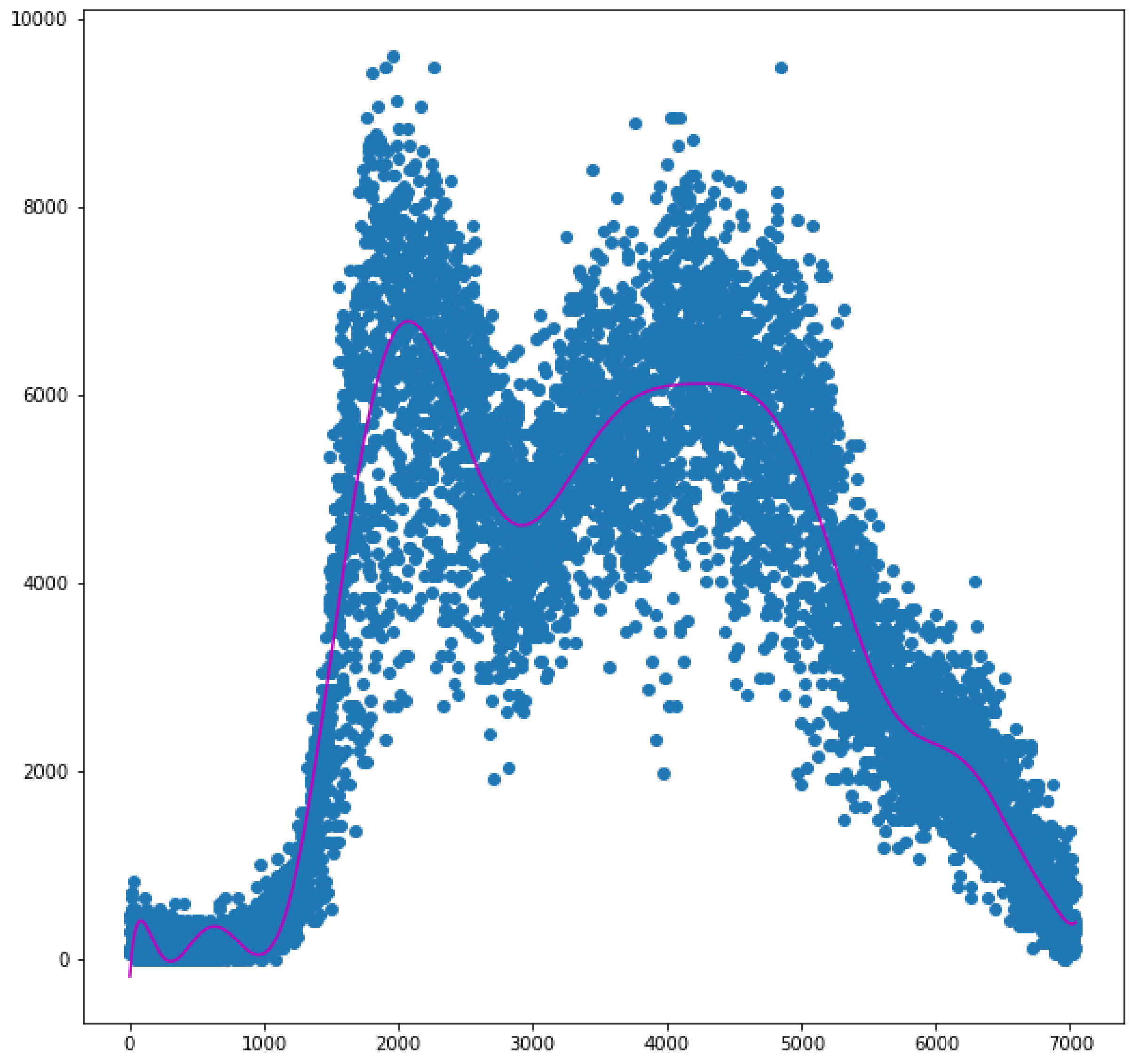
Figure 9.11 speed comparison by box plot

As mentioned before, the box plot in the right (orange line), has more low extreme values corresponding to the congestion periods. It means that the road of this measurement point gets significantly congested while the road from the other measurement point gets rarely congested.

c. Modelling

The modelling is the subject of main interest of this project. The model provided by this research project predicts the impact of an incident in the area of Utrecht over the network. To be able to generate this model it is needed some user inputs, some predefined elements and an algorithm to compute the simulations. The predefined elements consist of some values and other models that provide the fundamentals to run the algorithm. First, let's look at these basic models and optimal values and the process to obtain them.

The whole process will go through the three main variables mentioned before. The traffic flow, the average speed and the traffic density. First, from the traffic flow and density a regression between the time as an integer value. The obtained models predict the flow and density at any time of the day. The following picture represents an example of the flow regression.



Figures 10.0 flow model regression

The figure 10.0 shows the polynomial fit used to obtain the regression between the traffic flow and the time of the day. Note that the x axis are integer values which represent a time of the day in which the y axis value corresponds. It is necessary to transform the time to an integer value in order to make the regression. It is also seen that in the nighttime the flow value is overfitted. This is because the flow data at that period is very noisy. Despite the overfitting, the traffic flow at that time can be considered zero or almost zero because there are just sporadic vehicles going through the measurement point. For the rest of the day, the model clearly shows the evolution of the traffic flow. The same kind of regression can be applied for the traffic density.

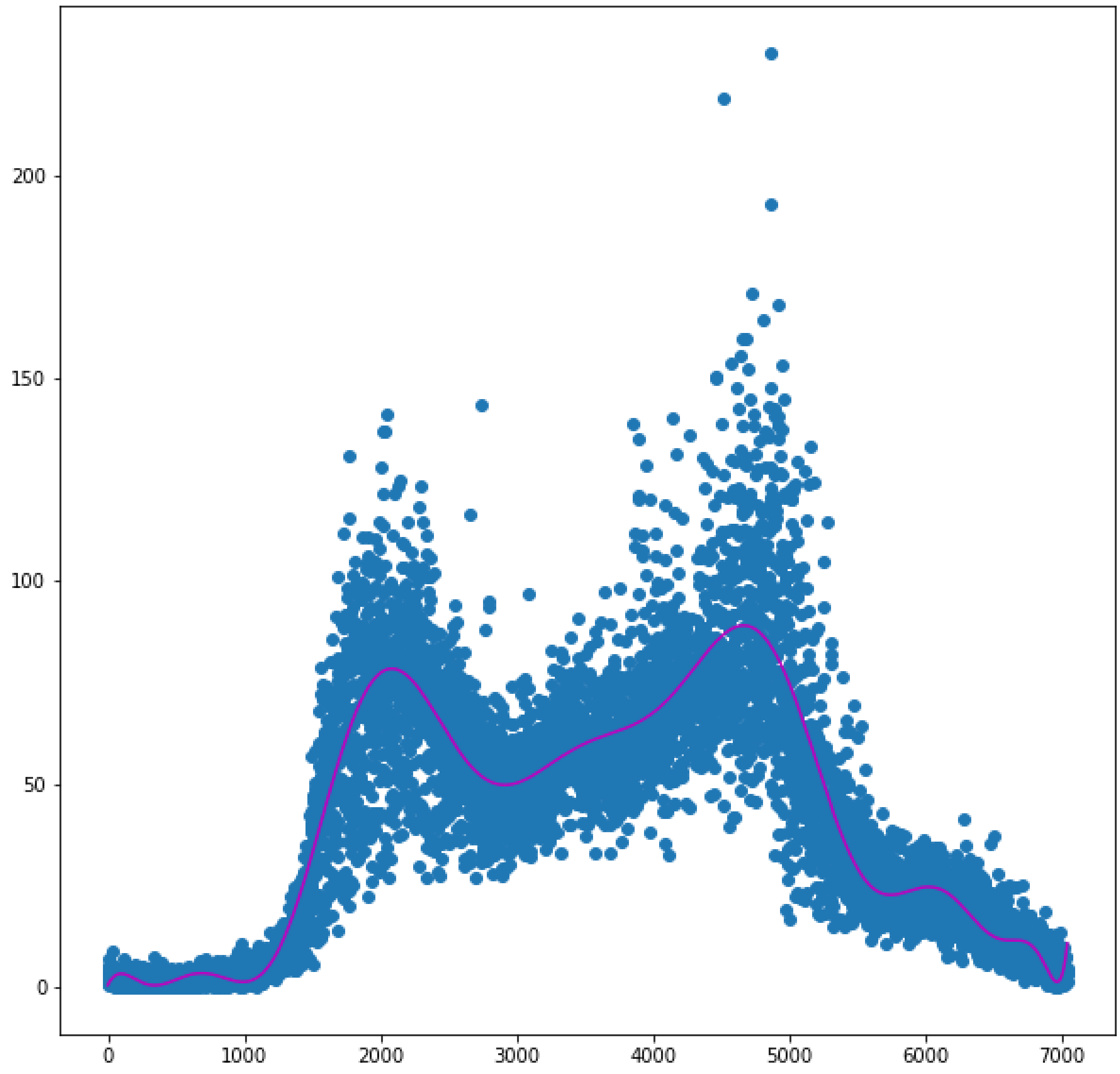


Figure 10.1 density model regression

It can be observed some overfitting in the extremes of the line chart as well. These overfitted values can be considered zero as well.

Even if both charts look similar, the line shapes don't follow the same evolution. In both line charts there are peaks on traffic activity. One peak in the morning, and the other one later in the afternoon. However, the peak observed in the afternoon, is more peak shaped for the density model than for the flow model. This is due to an excess on the traffic density which causes congestion and stops the traffic flow increase. This phenomenon will be later explained at the optimal values section.

The observed peak can take different shapes depending on the measurement point which they belong. For example, in an exit point of the city area, the afternoon peak would be greater than the morning peak while in an access point it would be the other way around. The following chart shows a comparison between two density models

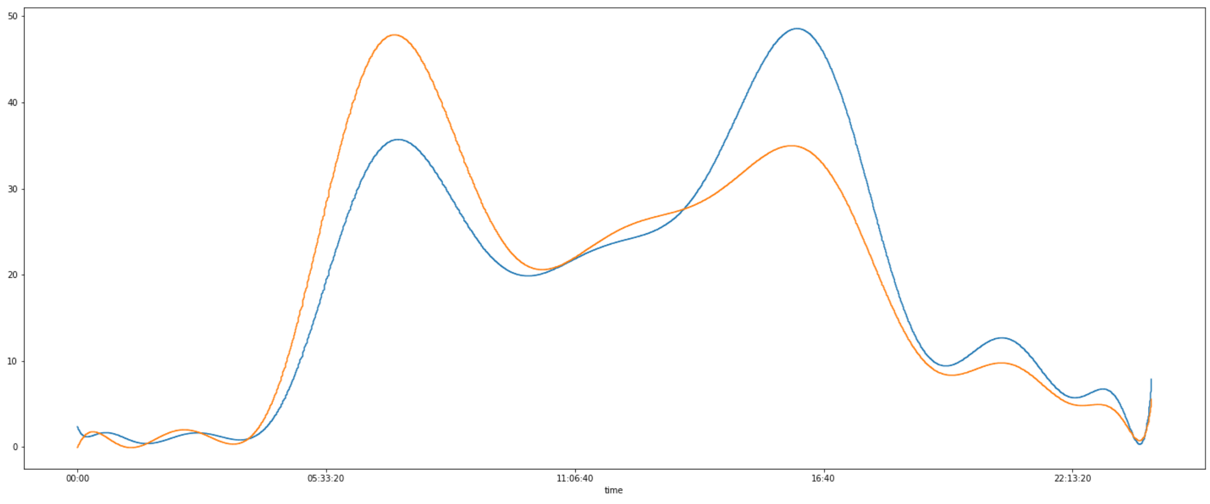


Figure 10.2 density model comparison

The orange line has the morning peak higher than the blue line. It can be concluded that the corresponding measurement point of the orange line is an access point to the city area while the corresponding measurement point of the blue line is an exit point of the city area.

From these first extracted models, it can be seen a component to classify the measurement point by looking at the line shapes. Later, when looking at the congestions originated, more classification components will be observed.

Once the flow and density models are extracted, it is time to find regressions between the main variables and the derived optimal values. Let's first start with the relation between the speed and flow data. To represent this, a scatter plot is made.

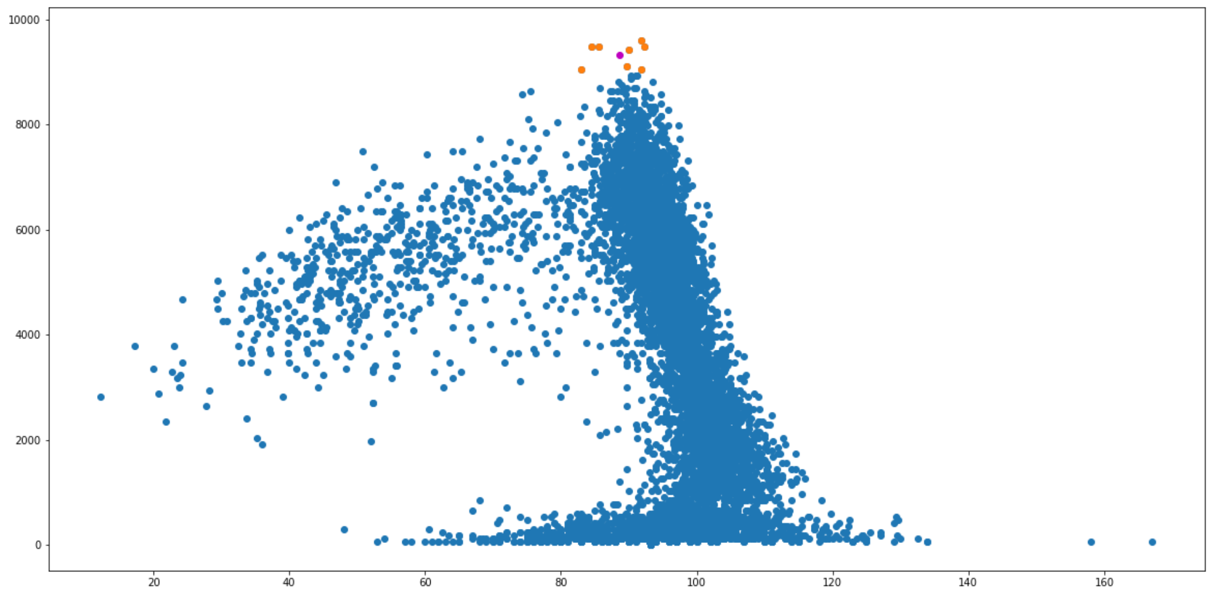


Figure 10.3 flow vs speed scatter plot

On the x axis the measured average speeds and in the y axis the measured traffic flows. Note that it exists some low flow values with disperse speed values. These values are measured during the night-time when there are less inputs and the data is noisier. These values aren’t considered when finding the optimal values because they can blurry the results.

From these relations it is obtained the optimal speed which is the speed of the road which can carry the maximum amount of traffic flow. To obtain these values the top flows are selected and a centroid value from that set of flows is calculated. Its corresponding speed will be the optimal speed of the road. In this case, the obtained optimal speed is 88.5625.

The next relation will be between the flow and the density of the road. Now, a polynomial fit is done to extract the optimal values. First, however, it is interesting to see the scatter plot of the relation. In this scatter plot, the spots will be coloured by its speeds.

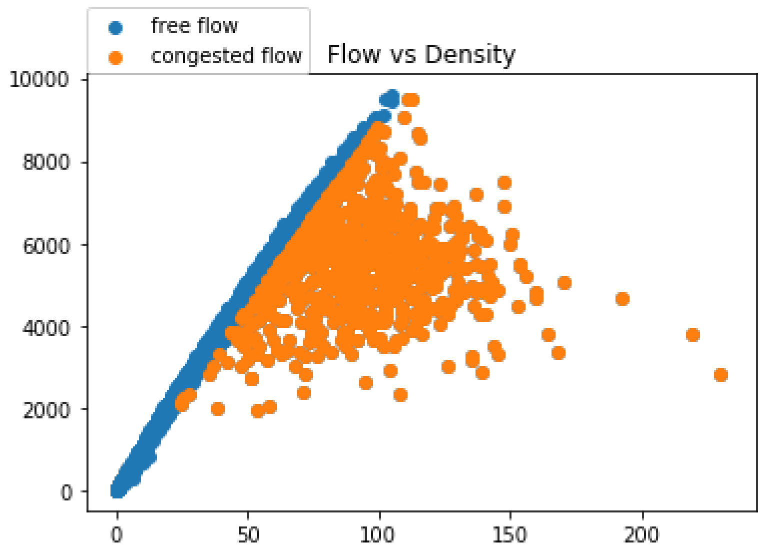


Figure 10.4 coloured scatter plot flow vs density

The orange spots are measurements with an average speed below the optimal speed. The reason why the segregation by the speed is done in this flow density scatter plot is to see how data behaves when the road average speed is below the optimal speed and when not. It is seen that is more ordered, following a linear increase, when the speed is above the optimal values (blue spots). The values below the optimal speed (orange spots), are more disordered and disperse and exit the linear relation. Let's see now the polynomial fit between flow and densities.

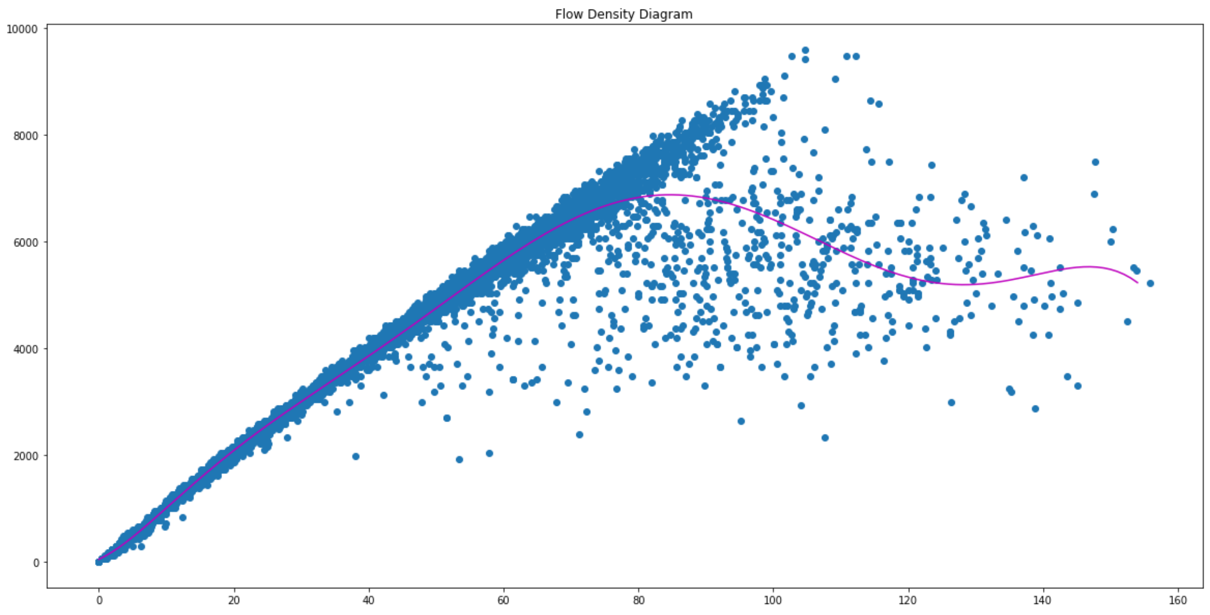


Figure 10.5 flow vs density regression

From this regression it is seen that an evolution on the traffic flow values depending on their density. The higher the density is, the bigger is the chance of obtaining a flow value out of the linear increase. From the regression, then a critical density is obtained. This critical density is an optimal value found at the peak of the curve when its slope is 0. The critical density is the maximum density which can carry free flow. If the critical density is reached, there is a high chance that the road gets congested. This optimal value can be used later as measure of the capacity of the road.

Using the density model and the critical density found, it is possible to see if the road, as a normal behaviour, reaches the critical density. In that case, it does.

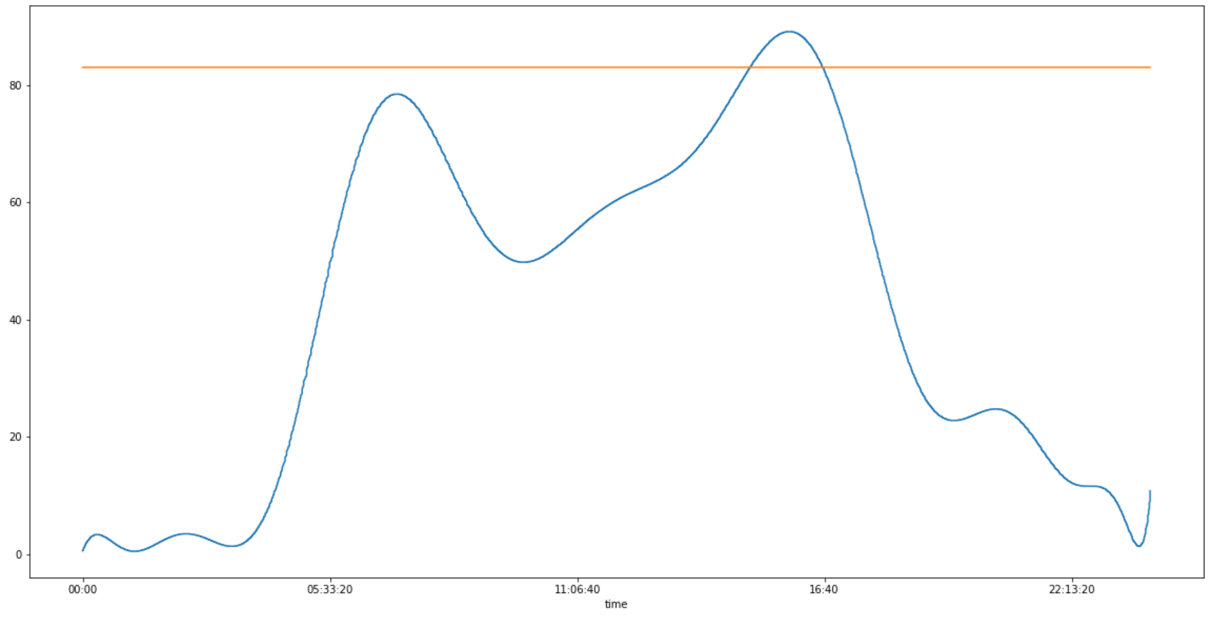


Figure 10.6 density model with critical density

It is observed that at this measurement point, the density during the afternoon peak is above the critical density. This means that at that time, the road is congested. By calculating the area in between the density model and the critical density, it is obtained the number of vehicles per kilometre which should be removed to avoid the congestion. At this point the calculated value is: 1907.5290715931114 ≈ 1908.

One practical utility of the density model is to detect unusual congestions. From the study previously done, suppose now that a certain point in the road network it is known to be congested at a certain period. By plotting the average of the speeds coloured by speed intervals into the density model, it can be seen at which time of the studied day the speeds got low. Note that the density model is a general description of the road while the plotted speeds are single measures of a day of data.

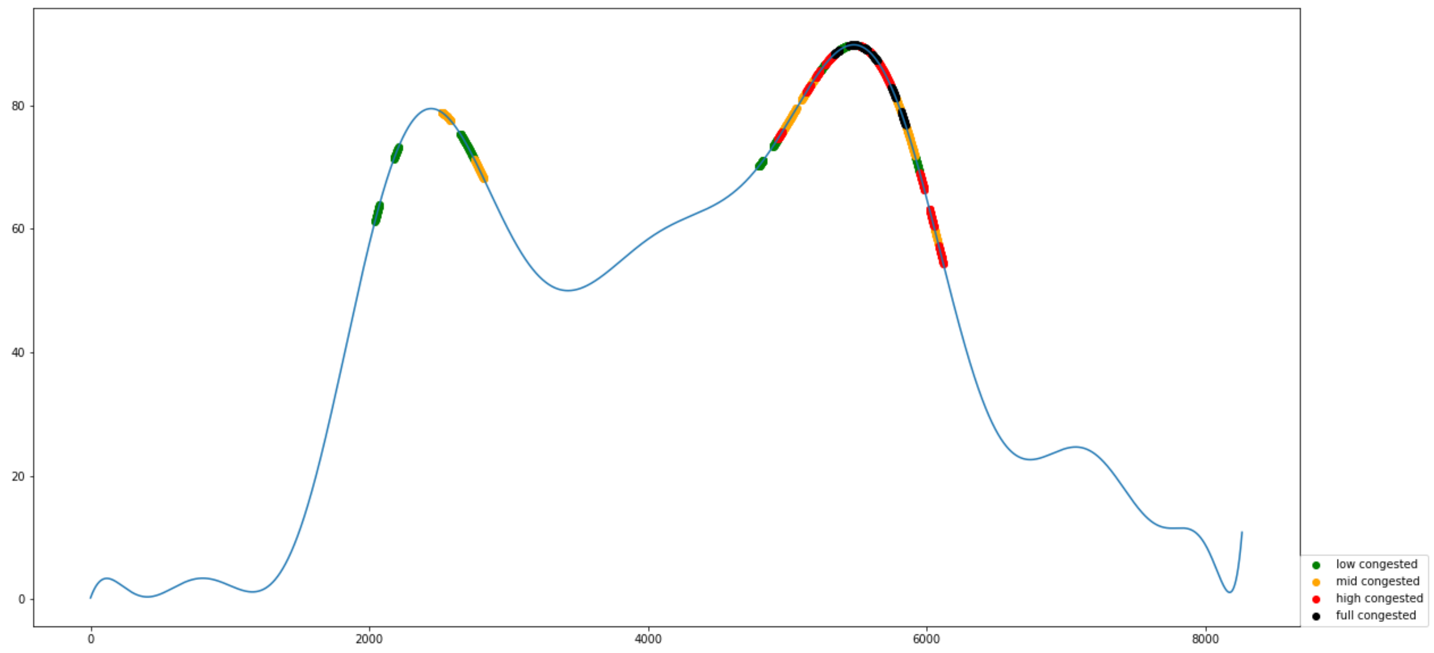


Figure 10.7 density model with coloured speeds (04-02-2020)

The figure 10.7 shows a regular day of traffic activity. In both peaks there are some speeds detected below the optimal speed. In the morning peak the plotted speed is slightly below the optimal speed and for a short period of time, while in the afternoon peak the plotted speeds are significantly low and indicates a congestion. This would confirm that the studied point is an exit gate of the city area. Now, let's compare the chart with the one from the Christmas day.

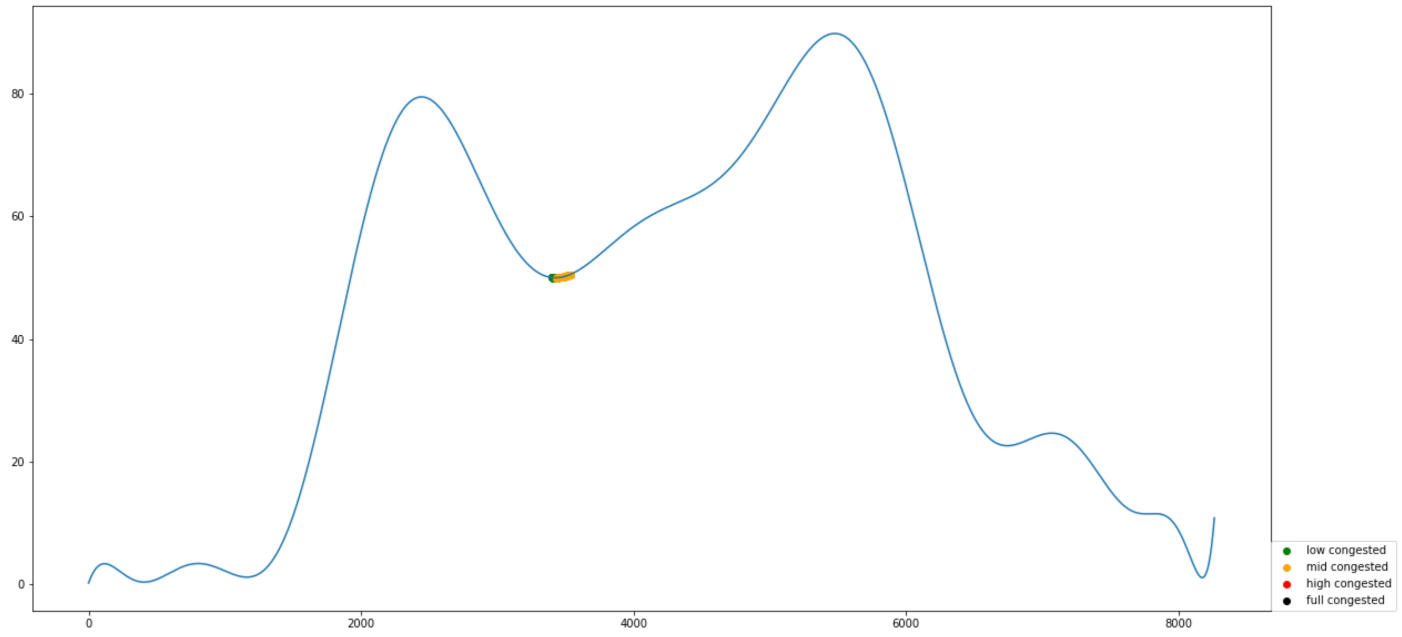


Figure 10.8 density model with coloured speeds (24-12-2019)

In this case, during almost the whole day there aren't low detected speeds. Just a small speed decrease at the mid-day. This is because the traffic activity wasn't so high. To contrast this, the following chart shows an unusual behaviour on the traffic road activity.

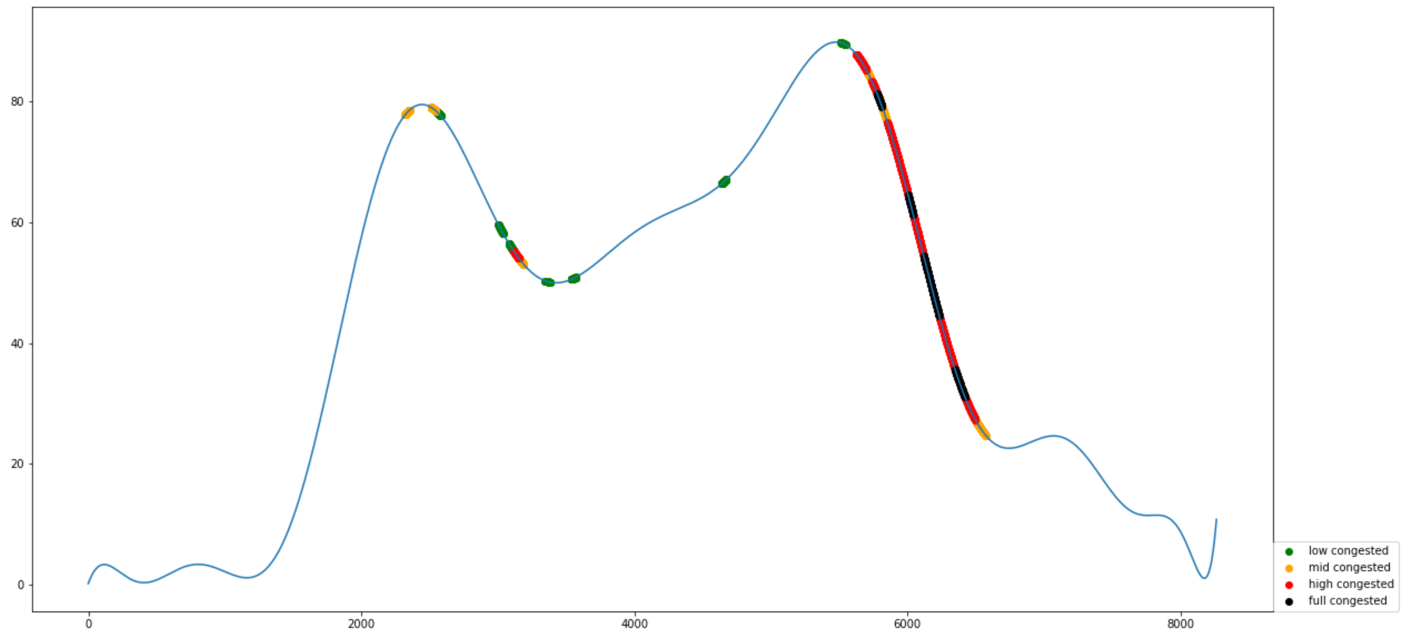


Figure 10.9 density model with coloured speeds (26-11-2019)

This line chart shows an unusual distribution of the congestion as they're aren't located in the density peaks. Besides, in the afternoon is seen a long congestion which is elongated until almost the night-time. The cause of this unusual situation is unknown, but the rush hour seems to be displaced a few hours later in both periods, morning and afternoon. Finally, another unusual congestion which had less impact is also visible in the following chart.

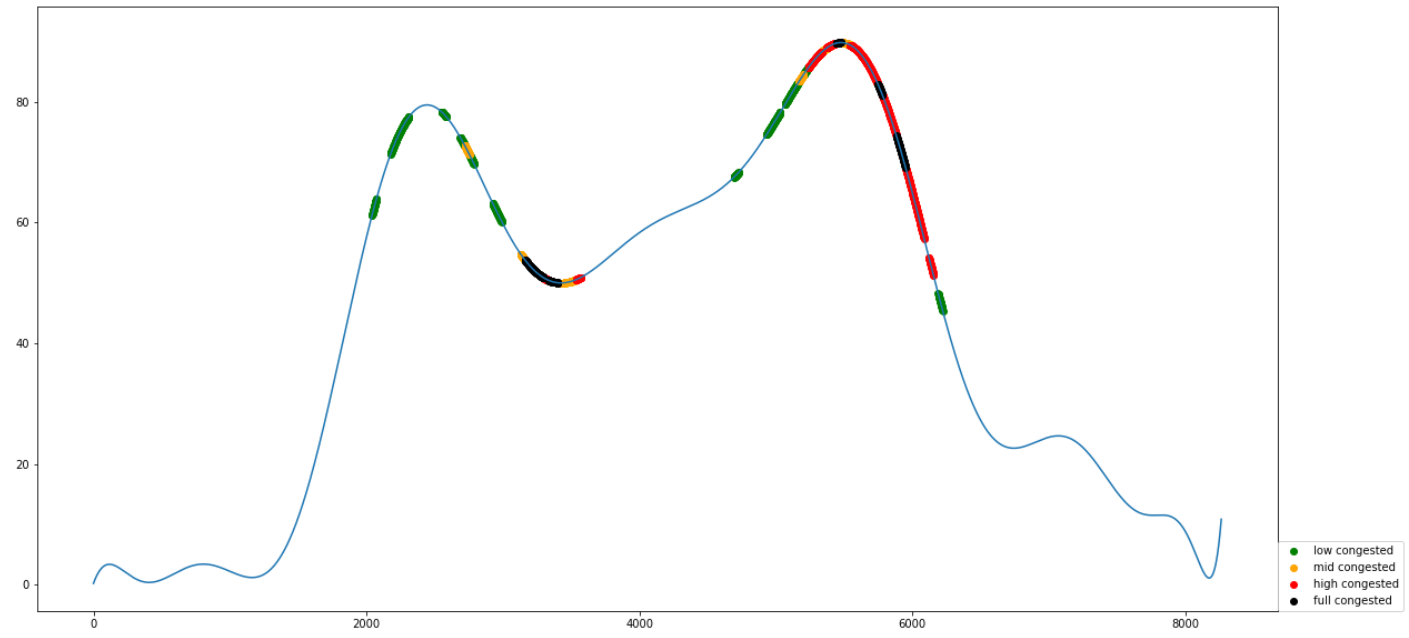


Figure 10.10 density model with coloured speeds (04-12-2019)

The congestions detected in the peaks are a usual situation. However, there are some black spots visible in the mid-day representing a traffic congestion. This congestion is in an unusual time of the day, that's why it can be said that it is an anomaly. The short period of time for in which the congestion takes to spread, indicates that the possible issue could be solved relatively quick.

To corroborate that when the critical density is reached a congestion occurs, let's look at the historical data of the whole day of the calculated density data. The following chart must be compared with the figure 10.10.

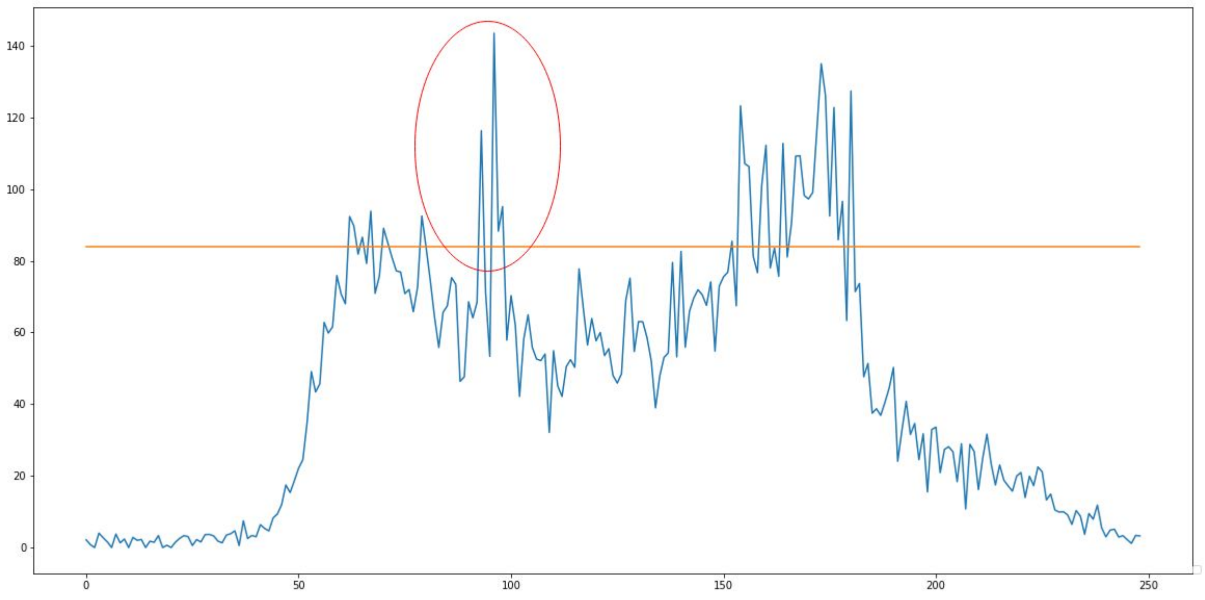


Figure 10.11 critical density reached causing congestion

As it is seen, when the density during the studied day reaches the critical density, the road gets congested (comparison between two charts). The figure 10.11 shows that in both peaks this critical density is reached. However, in the afternoon peak, the density is much above the limit than in the morning peak. This is reflected as well in the figure 10.10 when the speeds don’t indicate a high congestion in the morning peak while the afternoon peak is highly congested. The highlighted peak in the figure 10.11 corresponds to the anomaly in the figure 10.10. So, by looking at these two figures it can be confirmed that when the critical density is reached, a congestion is originated.

Before jumping into the congestion model and its algorithm, it is also interesting to look at the relation between the average speeds and the traffic density. The following regression represents this relation.

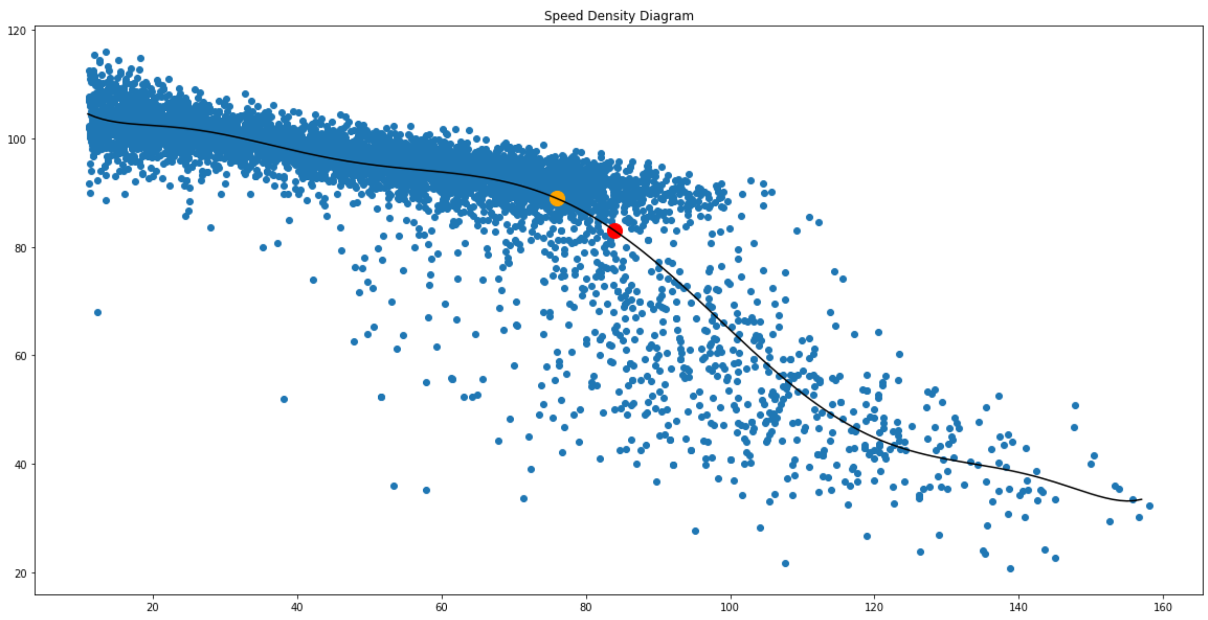


Figure 10.11 speed vs density regression

In this regression is also visible a significant difference between the data corresponding to a free flow state of the road and the data corresponding to a congested state of the road. By looking at density of the spots it is seen that the least dense spots are congested state data. It is because the congested situations aren't often detected. The regression line also shows how the average speed decreases when the density increases (and its decrease slope). The red dot shows the speed at the critical density and the orange spot shows the optimal speed. These two dots should be theoretically at the same position, nut in practice they're a little bit separated. The optimal speed is reached before the speed at the critical density, so it could be concluded that the first symptom of a road getting congested is a decrease below the optimal speed. Then, when the critical density is reached, the road gets congested.

d. Congestion Model

The congestion model is responsible to estimate the density over several points within the road network from a simulated congestion. The model is generated from some predefined models and clue values, some assumptions and the parameters which the user inserts as an input.

Assumptions

1. The sum of the flow that inside a measurement point is, at any time, 0.
2. When the congestion occurs, the traffic flow is uniformly redirected.
3. A road section which is behind the simulated congestion will be potentially congested, while a road section which is in front the simulated congestion will carry free flow.
4. A road branch originated in a congested section pointing forward, will carry free flow.

* Check the flow redirection, the bottleneck and the thermal conductivity sections for a better overview of the mentioned assumptions

Predefined elements

1. Flow model: estimates the traffic flow at a point at a certain moment of the day.
2. Density model: estimates the traffic density at a point at a certain moment of the day.
3. Optimal speed: the calculated speed which can carry the maximum amount of traffic flow.
4. Critical density: the critical density value from which the road starts to get congested if it is reached.
5. Density-flow model for congested situations: this model is a mathematical equation which describes the traffic density from a new calculated flow assuming that the road is congested.

Parameters

1. Time of the day when the congestion is simulated.
2. The simulated congestion including:
    a. The percentage of the available capacity from the usual capacity of the road where the congestion occurs.
    b. The edge where the congestion occurs (two points in the graph).
3. The number of layers taken in the subgraph which will be run by the algorithm.

Output

The output of the model is the set of the estimated densities for each point involved in the calculation compared with the corresponding critical density and the expected density with no congestion.

Traffic flow redirection algorithm

The estimated values are product of an algorithm which iterates over all the points which are part of the selected context. The critical density is used as a measure of capacity. The following list shows how the algorithm performs using the mentioned inputs and predefined elements.

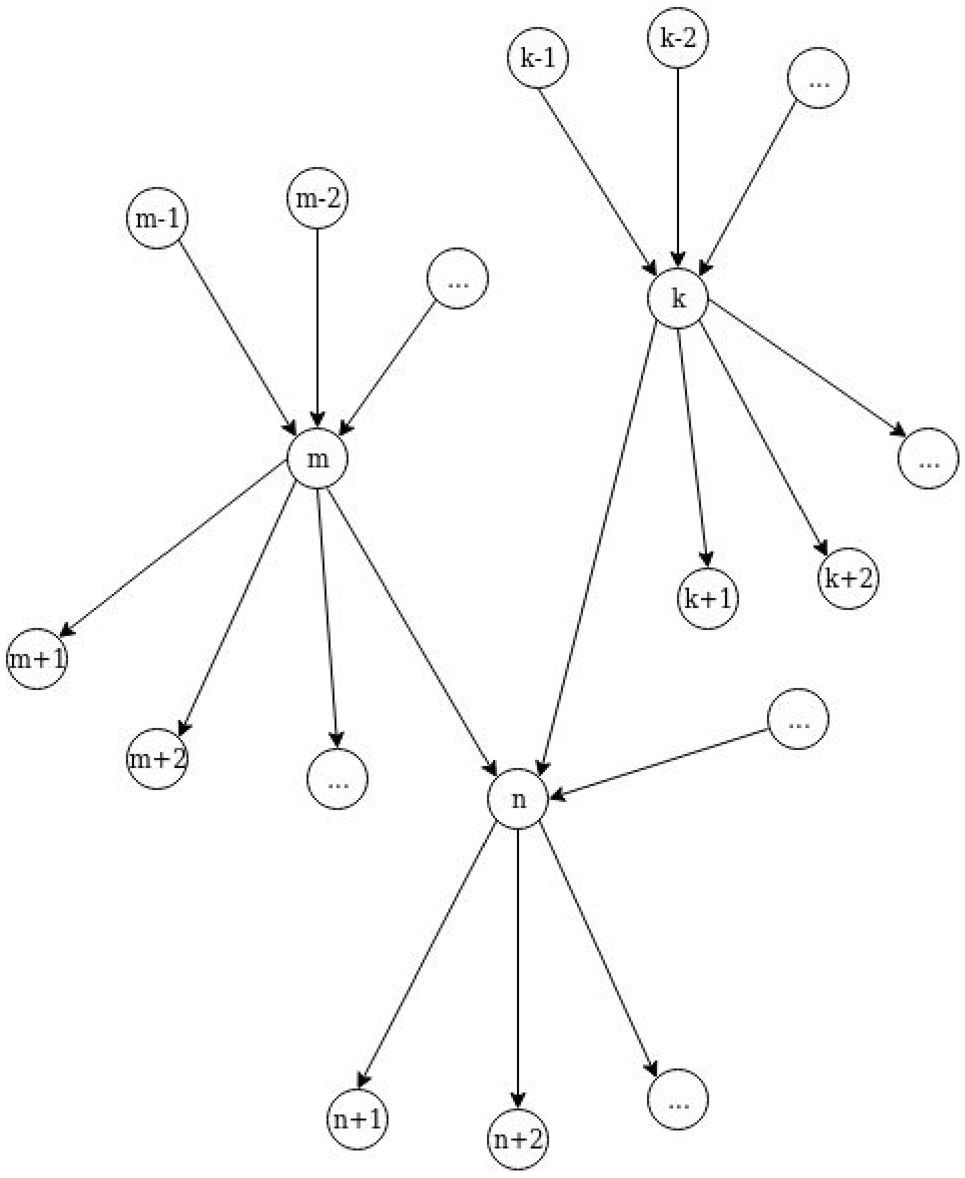


Figure 11.0 general network for better understanding of the algorithm process

1. The program loads the inputs and all the nodes specified by the user and calculates the predefined models and clue values for all them. This process is done by a recursive algorithm which labels the nodes depending on its origin (input or output of the origin node).

2. Taking the inserted congestion, the algorithm splits into two processes. One starting from the tail node of the congested edge (backwards) and another one starting from the head node of the congested edge (forward). The backward partition will simulate congestion an increase while the forward partition will simulate traffic density decrease (assumptions 4 and 5).

3. (Backward partition):

Calculation of a new critical density and a new flow for the starting node:

$$q_n = q_n \cdot p$$

$$KC_n = KC_n \cdot p$$

(Forward partition):

Calculation of a new flow for the starting node:

$$q_n = q_n \cdot p$$

Note that the flow will become similar as the backward node, but the capacity won't decrease which makes the density decrease.

Where q_n is the estimated flow at the specified time of the day of the *n* node, KC_n its critical density*,* and *p* the available percentage of the road capacity.

4. (Both partitions):

For the iterated node, the neighbours which aren't yet calculated by the algorithm, are taken. For each of the taken nodes, a new partition is made (step number 5). The new partition will calculate a new flow value for its node. The formula used for that will depend on the position in the edge of the node. If the reached node is a tail, it will calculate the new flow as a backward partition unless the origin node belongs to a forward partition. Then it would be calculated as forward partition as well. If the node is a head, it will calculate the new flow as a forward partition.

5. (Backward partition):

According to the figure X.X and supposing that from the node *n* the node *m* is reached: the backward partition will take *m* as its node. The new flow considering the new values that might be previously calculated, is estimated by the formula:

$$q_m = (q_n - \frac{KC_n}{KC_n + KC_{k+1} + ... + KC_{k_{max}}} \cdot q_k - ...) \cdot \frac{KC_n + KC_{m+1} + ... + KC_{m_{max}}}{KC_n}$$

(Forward partition):

Supposing that n is taken as the partition node, the new flow considering the new values that might be previously calculated, is estimated by the formula:

$$q_n = \frac{KC_n}{KC_n + KC_{m+1} + ... + KC_{m_{max}}} \cdot q_m + \frac{KC_n}{KC_n + KC_{k+1} + ... + KC_{k_{max}}} q_k + ...$$

6. (Backward partition):

For backward partition nodes, the new estimated density will be calculated considering them as potentially congested. In this case, the formula to be used is:

$$k_{new} = k + Kc_0 \cdot (q_{n_{new}}/3600 - q_n/3600)$$

* Check the thermal conductivity proof for the proof of this formula

(Forward partition):

For forward partition nodes, the new estimated density will be calculated considering the partition node traffic flow as free flow. In this case the formula to be used is:

$$k_{new} = \frac{q_{new}}{v_{opt}}$$

7. Check if all the nodes specified by the user are done. If not, go back to step 4.

Observations

1. The new density for forward partition nodes is calculated from the estimated flow and the optimal speed. The reason why the optimal speed is taken is because, in the worst case, it could carry the maximum flow.

2. If the road is potentially congested, the system is unstable. When the system becomes unstable it becomes congested due to a succession of circumstances. The excess of flow causes an increase on the density which causes a decrease of the speed. Then, the decrease of the speed causes an increment of the traffic density which causes a decrease of the flow. And so on. This process cannot be calculated in detail, and other ways to estimate the new density must be taken.

3. The reason why the formula for congested situations to calculate the new density is not taken to calculate densities for uncongested situations is because the concept of capacity/permeability is just ignored.

4. If the flow decreases in a forward partition node is because a decrease on the number of cars at that point, while if the flow decreases in a backward partition node, is because an increase of the traffic density which generates a new density above critical density (the system is unstable).

5. If the flow in a forward partition node is increased is due to a redirection of the flow. This could cause another congestion. If the estimated density calculated from the new estimated flow is above the critical density of the point, the road will get congested at that point.

Example

In order to show how the algorithm performs, a single congestion will be simulated on the edge which is formed by the nodes identified as:

- Head: <id: 903, reference: PUT01_N230.03L_1>
- Tail: <id: 841, reference: GEO1A_A_RWS_359507>

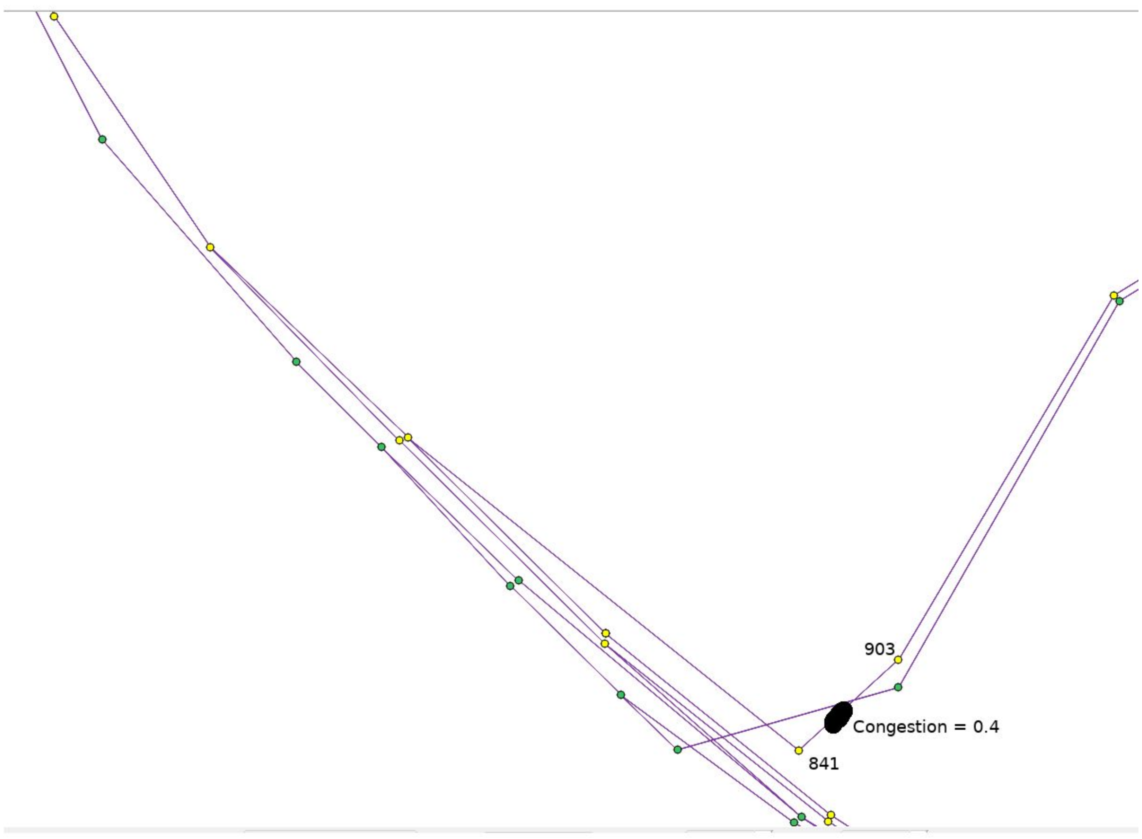


Figure 11.1 subgraph taken for the example simulation (yellow spots)

The simulated congestion will decrease the road capacity a 60%. *(p = 0,4)*.

The other inputs inserted by the user will be:

- Time: 12:30
- Depth: 5

First, the program loads the needed data to all the points involved in the simulated case. If the initialization is completed successfully, as a confirmation is obtained:

```
903 <0> <critical density: 145, optimal speed: 71.12530864197531, maximum flow: 9960.0> :

[841 <1> <critical density: 235, optimal speed: 24.08333333333333, maximum flow: 6333.333333333333> :
```

```
[631 <1> <critical density: 37, optimal speed: 89.125, maximum flow: 3367.5> :

[634 <1> <critical density: 102, optimal speed: 93.95555555555556, maximum flow: 10720.0> :

[529 <1> <critical density: 119, optimal speed: 93.5625, maximum flow: 11617.5> :

[315 <1> <critical density: 117, optimal speed: 90.62962962962962, maximum flow: 11886.666666666666> :

[113 <1> <critical density: 109, optimal speed: 92.8, maximum flow: 11647.5> :

[]]], 542 <-1> <critical density: 51, optimal speed: 96.875, maximum flow: 5767.5> :

[270 <-1> <critical density: 66, optimal speed: 93.03703703703704, maximum flow: 7393.333333333333> :

[]]], 188 <-1> <critical density: 32, optimal speed: 95.94444444444444, maximum flow: 3246.6666666666665> :

[646 <-1> <critical density: 34.47562776957164, optimal speed: 96.71428571428571, maximum flow: 3334.285714285714> :

[461 <-1> <critical density: 51, optimal speed: 79.86111111111111, maximum flow: 5560.0> :

[]]]]], 923 <-1> <critical density: 96, optimal speed: 83.58253968253968, maximum flow: 8666.666666666666> :

[917 <-1> <critical density: 96.71740935510657, optimal speed: 86.02380952380952, maximum flow: 8320.0> :

[922 <-1> <critical density: 89.81407913554743, optimal speed: 89.9, maximum flow: 8074.285714285715> :

[]]]]
```

The obtained output is a list of the loaded nodes with its clue values. Even if it is not seen, one of the nodes have no critical density (id 922). This is because the road never gets congested and it cannot be determined by the historical data. Usually these points are found on the exit gates of the road. When doing the calculations, a default critical density will be assigned. It will be the maximum flow divided by the optimal speed of the node.

Once all the nodes are loaded to the program, the algorithm starts. It first makes two partitions. The initial forward (node 903), and the initial backward (node 841). The first new flow values are calculated as it is previously mentioned. A new critical density is calculated as well for the backward partition node. It will be later used for other nodes as a measure of permeability of the road. The code responsible of that calculation is:

(First backward partition):

```
def __first_backward(self, root, p):  # root is the node object and p the percentage of the available capacity
  kc = root.mb.criticalDensity  # critical density without the congestion
  kc_new = kc * p  # critical density with the congestion
  flow_model = root.mb.flow_model  # flow model describes the flow during the day
  q = flow_model(self.timeToInt(root))  # flow without congestion, timeToInt converts a string time to an integer
  q_new = q * p  # flow with congestion
  MB_root.kc_new = kc_new  # set new critical density of the first backward node
  root.q_new = q_new  # set new flow for the node
  for i in root.adjacent_roots:  # split again into partitions
    if (i.master < 0):
      self.__backward(i)
    if (i.master > 0):
      self.__forward(i)
```

(First forward partition):

```
def __first_forward(self, root, p):
  flow_model = root.mb.flow_model  # flow model describes the flow during the day
  q = flow_model(self.timeToInt(root))  # flow without congestion
  q_new = q * p  # flow with congestion
  root.q_new = q_new  # set new flow for the node
  for i in root.adjacent_roots:  # make other forward partitions to estimate new densities
    self.__forward(i)
  k_new = self.__calculateNewKForward(root)  # calculates the estimated density
  root.k_new = k_new  # saving the estimated density to the node object
  root.v_new = q_new / k_new  # calculating estimated speed and saving it into the node object
```

After completing the first initial calculations, the algorithm generates more partitions to estimate the values for the rest of the nodes. It will jump to a backward partition if from a backward node the node adjacent node is backward too, while it will jump to a forward partition if the adjacent node is forward. From a forward partition it will always jump to a forward partition. The new partitions made use different formulas mentioned on the previous section as well. These formulas need some inputs that will first be taken from the node in which new values are about to be estimated.

(Backward partition):

```
def __backward(self, root):
```

```
root_origin = root.origin  # getting the origin node from this partition


q_dict = {}  # dictionaries containing values later used in the formula

kc_dict = {}

 for i  in root_origin.adjacent_roots:  # getting values for the backward neighbours which aren't
the current node

    if (

             i.master < 0 and i != root):  # getMaster provide information about the partition type of
the current node

        if (i.q_new > 0):  # checking if a new flow is already set

            q_dict[i.node_id] = i.q_new  # setting flow value from adjacent node to the dictionary

        else:

            q_dict[i.node_id] = i.mb.flow_model(

                self.timeToInt(i))  # setting flow value from adjacent node to the dictionary

        kc_i_list = []

        for j in i.adjacent_roots:  # getting a list of the critical densities

            if (j.master > 0 and j != i):

                kc_i_list.append(j.mb.criticalDensity)

        kc_dict[i.node_id] = kc_i_list

kc_i_list = []

 for i  in root.adjacent_roots:  # getting a list of the critical densities of the forward nodes from
the current nodes

    if (i != root_origin and i.master > 0):

        kc_i_list.append(i.mb.criticalDensity)

kc_dict[root.node_id] = kc_i_list  # setting the list to the dictionary

# the flow of the origin node must a new one

# calculating new flow using the formula and its extracted inputs

      q_new   =   self.__backward_function(root.node_id,   q_dict,   kc_dict,   root_origin.q_new,
MB_root.kc_new)

root.q_new = q_new  # setting the new flow to the node object

for i in root.adjacent_roots:  # generating new partitions

    if (i.master < 0):

        self.__backward(i)

    if (i.master > 0):

        self.__forward(i)

k_new = self.__calculateNewKBackward(root)  # calculates the estimated density

root.k_new = k_new  # saving the estimated density to the node object
```

```
  root.v_new = q_new / k_new  # calculating estimated speed and saving it into the node object
```

(Forward partition):

```
def __forward(self, root):
  mb = root.mb  # getting the origin node from this partition
  kc = mb.criticalDensity
  kc_dict = {}  # dictionaries containing values later used in the formula
  q_dict = {}
  for i in root.adjacent_roots:  # collecting values for the backward nodes
    if (i.master < 0):  # checking the backward adjacent nodes
      kc_i_list = []
      for j in i.adjacent_roots:
        if (j.master > 0 and i != j):
          kc_i_list.append(
                j.criticalDensity)  # collecting the critical densities from the forward adjacent
nodes
      kc_dict[i.node_id] = kc_i_list
      if (i.q_new > 0):  # setting the flows from the backward adjacent nodes to the dictionary
        q_dict[i.node_id] = i.q_new
      else:  # if the new flow is not set to the node object
        q_dict[i.node_id] = i.mb.flow_model(self.timeToInt(i))  # new flow
  kc_i_list = []
  for i in root.origin.adjacent_roots:  # collecting details for the origin node
    if (i.master > 0 and i.node_id != root.node_id):
      kc_i_list.append(i.mb.criticalDensity)
  kc_dict[root.origin.node_id] = kc_i_list
  if (root.origin.q_new > 0):
    q_dict[root.origin.node_id] = root.origin.q_new
  else:
    mb_origin = root.origin.mb
    flow_model_origin = mb_origin.q_new
    q_origin = flow_model_origin(self.timeToInt(root.origin))
    q_dict[root.origin.node_id] = q_origin
  # setting the new flow to the object nodes
  if (kc <= 0):
    kc = root.origin.mb.criticalDensity
    root.mb.criticalDensity = kc
  q_new = self.__forward_function(q_dict, kc_dict, kc)
  root.q_new = q_new
  for i in root.adjacent_roots:  # generate new partitions
      self.__forward(i)
  # setting the new speed and density to the object nodes
  k_new = self.__calculateNewKForward(root)
  root.k_new = k_new
  root.v_new = q_new / k_new
```

In both sections of code, the function representing the formula is called. It performs as it follows:

(Backward function):

```
def __backward_function(self, root_id, q_dict, kc_dict, q_origin, kc_origin):
  q = q_origin
  for key in q_dict:  # iterating through the dictionary (each adjacent node)
      kc_total = 0
      if (key != root_id):
            for i in kc_dict[key]:  # adding the critical densities (denominator) for each array in the dictionary
             kc_total = kc_total + i
          kc_total = kc_total + kc_origin  # adding the critical density from origin as well
            q = q - (kc_origin / kc_total) * q_dict[key]  # qn - KCn / (KCn + KCk+1 + ... + KCkmax * qk)
  kc_total = 0
   for i in kc_dict[root_id]:  # adding the critical densities of the adjacent nodes from the current node
      kc_total = kc_total + i
  kc_total = kc_total + kc_origin  # adding the critical density from origin as well
  q = q * (kc_total / kc_origin)  # calculating new flow
  return q
```

(Forward function):

```
def __forward_function(self, q_dict, kc_dict, kc):
  q = 0
  for key in q_dict:  # iterating through the dictionary (each adjacent node)
      kc_total = 0
          for i in kc_dict[key]:  # adding the critical densities (denominator) for each array in the dictionary
          kc_total = kc_total + i
```

```
        kc_total = kc_total + kc  # adding the critical density from origin as well
        try:
            q = q + (kc / kc_total) * q_dict[key]  # calculation new flow
        except:
            print("ERROR! two nodes with no critical density")
    return q
```

Finally the backward and forward functions use the following other functions to estimate the new density value.

```
def __calculateNewKForward(self, root):  # calculate new density for forward nodes
    k_new = root.q_new / root.mb.optSpeed  # using optimal speed -> worst case
    return k_new

def __calculateNewKBackward(self, root):  # calculate new density for backward nodes
    q = root.mb.flow_model(self.timeToInt(root))
    k = root.mb.density_model(self.timeToInt(root))
    k_new = k + self.__densityForCongestedState(q, root.q_new, MB_root.kc_new)
    return k_new

def __densityForCongestedState(self, q, q_new, kc):  # density by flow difference
    k = -kc * (q_new / (60 * 60) - q / (60 * 60))  # transform to seconds
    if (k < 0):  # absolute value
        k = k * -1
    return k
```

The programs iterate recursively calling the backward and the forward function. When all nodes have the estimated values set, it stops and returns the predicted values. As an example, the following string shows the output of the congestion model.

Id	**Optimal value**	**Expected value**	**Estimated values**
903	Maximum flow: 9960 Critical density: 145 Optimal speed: 71.12	Flow: 42020.77 Density: 56.46	Flow: 1681.11 Density: 97.09 Speed: 17.31
841	Maximum flow: 6333.33 Critical density: 235	Flow: 1593.19 Density: 66.76	Flow: 637.27 Density: 26.46 Speed: 24.08

	Optimal speed: 24.08		
631	Maximum flow: 3367.5 Critical density: 37 Optimal speed: 89.125	Flow: 1417.22 Density: 15.22	Flow: 2220.42 Density: 24.91 Speed: 89.125
634	Maximum flow: 10720 Critical density: 102 Optimal speed: 93.95	Flow: 4162.78 Density: 44.4	Flow: 6715.96 Density: 71.48 Speed: 93.95
529	Maximum flow: 11617.5 Critical density: 119 Optimal speed: 93.56	Flow: 4814.96 Density: 52.53	Flow: 6715.96 Density: 71.78 Speed: 93.56
315	Maximum flow: 11886.66 Critical density: 117 Optimal speed: 90.62	Flow: 4874.44 Density: 53.15	Flow: 6715.96 Density: 74.1 Speed: 90.62
113	Maximum flow: 11647.5 Critical density: 109 Optimal speed: 92.8	Flow: 4883.81 Density: 52.46	Flow: 6715.96 Density: 72.37 Speed: 92.8
542	Maximum flow: 5767.5 Critical density: 51 Optimal speed: 96.87	Flow: 1979.75 Density: 19.77	Flow: 6510.32 Density: 67.2 Speed: 96.87
270	Maximum flow: 7393.33 Critical density: 66 Optimal speed: 93.03	Flow: 3112.89 Density: 31.80	Flow: 6510.32 Density: 69.97 Speed: 93.03
188	Maximum flow: 3246.66 Critical density: 32 Optimal speed: 95.94	Flow: 947.94 Density: 9.53	Flow: 4191.77 Density: 43.68 Speed: 95.94
646	Maximum flow: 3334.28	Flow: 955.31 Density: 9.4	Flow: 7853.29 Density: 81.2

	Critical density: 34.47 Optimal speed: 96.71		Speed: 96.71
461	Maximum flow: 5560 Critical density: 51 Optimal speed: 79.86	Flow: 2078.37 Density: 20.81	Flow: 7853.29 Density: 98.33 Speed: 79.86
923	Maximum flow: 8666.66 Critical density: 96 Optimal speed: 83.58	Flow: 3723.34 Density: 42.36	Flow: 1681.11 Density: 75.26 Speed: 22.33
917	Maximum flow: 8320 Critical density: 96.71 Optimal speed: 86.02	Flow: 3038.62 Density: 32.97	Flow: 1681.11 Density: 54.84 Speed: 30.65
922	Maximum flow: 8074.28 Critical density: 89.81 Optimal speed: 89.9	Flow: 2832.64 Density: 32.16	Flow: 1681.11 Density: 52.33 Speed: 32.12

Flow redirection

The assumptions 1 and 2 are based on the flow dynamics physical theories. The first assumption (the sum of the flow that inside a measurement point is, at any time, 0) is a fundamental step when solving electronic circuits. In that case the voltage is taken instead. What this assumption is saying is that all the flow influencing one node must be equal to the flow exiting the node. Later, is also assumed that the traffic flow redirects uniformly. This assumption is perfectly proven in other physical dynamic contexts as the objects which are redirected are particle. However, in traffic flow analysis there exist a human component which might make this assumption to fail in some cases. Nevertheless, it was necessary to have a criterion when predicting the traffic flow.

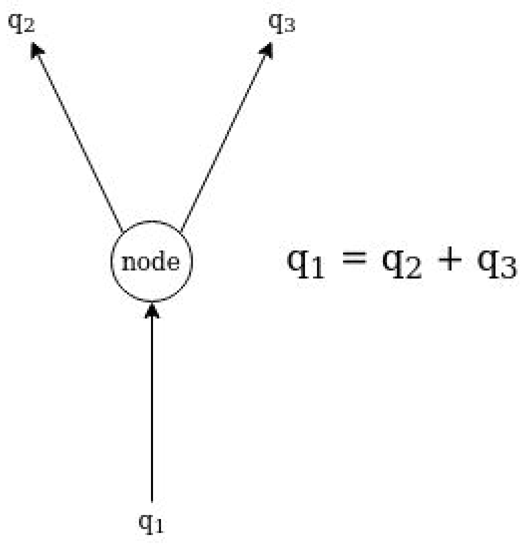


Figure 11.2 conservation of flow

Bottleneck

In traffic flow theory, there is a similitude used to explain the causes and the effects of a congestion. This similitude is done comparing the traffic flow with the fluid flow. To represent it is usually used a bottleneck because the phenomena is the same. The decrease at certain path of the capacity of the road makes the vehicles to stack if the density of the road is above the decreased capacity.

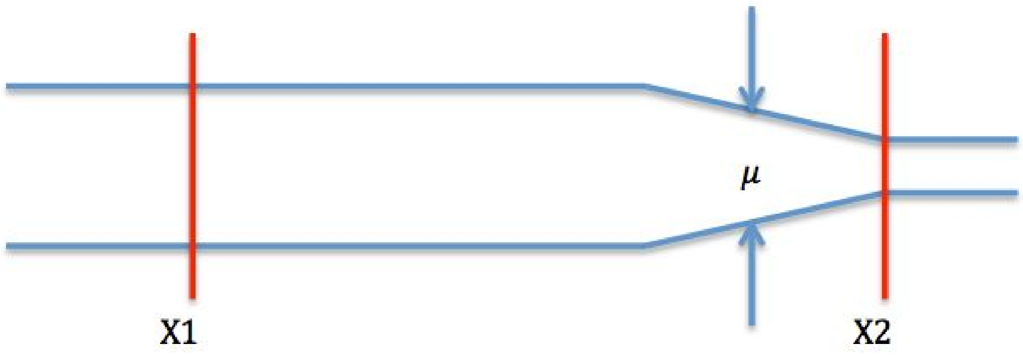


Figure 11.3 bottle neck representation

The figure 11.3 represents the road capacity decrease as reduction of the road size. The μ is the percentage of the available capacity of the road. The X2 is a point situated after the congestion while the X1 is a point situated before. Note that X2 will carry free flow according the flow dynamics theories.

In the other hand X1 will be found to be congested if the vehicles get indeed stack because of the congestion.

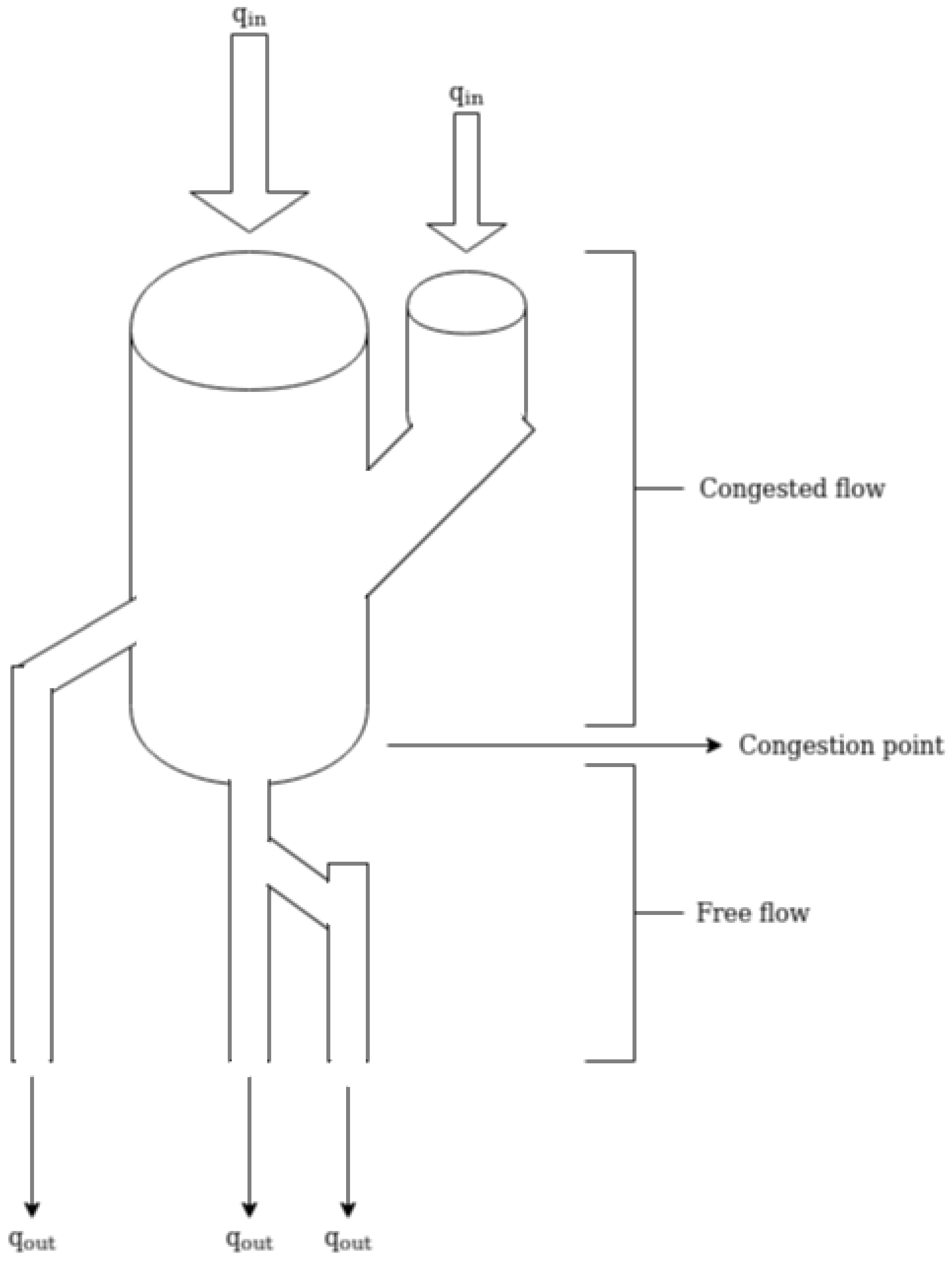


Figure 11.4 full representation of the bottleneck (all possibilities considered)

Now, at the figure 11.4 the bottleneck is extended to any possible case which can be found in a road network. All the out gates carry traffic flow even if they are branched from a congested section. The sections which are found before the congestion point gets congested if the number of vehicles which they carry don't fit the new road capacity.

Thermal conductivity

The estimated density from a node which is potentially congested is calculated by the formula:

$$k_{new} = KC_0 \cdot (q_{n_{new}}/3600 - q_n/3600)$$

Algorithm step 6 (backward partition)

The formula is inspired from the thermal conductivity theory. This theory can be adapted to other contexts by doing some modifications. The reason why it works well with traffic flow is because in both contexts flows and densities are modelled. The formula is obtained from the following process:

The thermal conductivity equation, which describes the heat flux density is:

$$q = -k \nabla T$$

Where *q* is the heat flux, *k* a conductivity constant of the material in which the heat is transmitted and ∇T the gradient of the temperature.

By bringing the equation to a one-dimensional equation (*x* component of *T*), it is obtained:

$$q_x = -k \frac{\partial T}{\partial x}$$

Transforming this equation to the traffic flow analysis study case it is obtained:

$$k = -Kc \frac{\partial N}{\partial x}$$

Where *k* is the traffic density, *Kc* the critical density as a road capacity and *N* the number of counted vehicles.

The increment of the number of detected cars respect the increment of distance can be also expressed as:

$$\frac{\Delta N}{\Delta x}$$

If the increment of distance is from the node where the congestion is originated to the node which is being analysed.

Let's label now this increment at time 0, and when the congestion reaches the analysed node time 1.

When the congestion reaches the analysed node, the new estimated flow is in its final state. Note that the new calculated flow of the node where the congestion is originated is in its final state as soon as the algorithm is initialized. So, now the increment of the number of vehicles detected at the analysed node respect time (0 and 1), will be the same as the increment of the number of vehicles at time 0 respect the distance between nodes. However, just the flow is known (not a count of vehicles). To get a figure representing a count of vehicles it is needed to consider a time interval as an instant. One second would be an appropriate time interval value. So, the increment of flow is reduced to an interval of one second and the obtained value is used to finally estimate the new density of the node.

$$k = -Kc\frac{\Delta N}{\Delta x} \rightarrow k = -Kc\frac{\Delta q}{\Delta t}$$

$$\frac{\Delta q}{\Delta t} = q_1 - q_0 = \frac{\Delta N_1}{\Delta t_1} - \frac{\Delta N_0}{\Delta t_0}; \Delta t_1 = \Delta t_2 \rightarrow \frac{\Delta N_1 - \Delta N_0}{\Delta t_0} \; or \; \frac{\Delta N_1 - \Delta N_0}{\Delta t_1}$$

The expression $\frac{\Delta N_1 - \Delta N_0}{\Delta t_0}$ becomes $\Delta N_1 - \Delta N_0$ by reducing the value to an instant (1 sec).

$$k = -Kc(\Delta N_1 - \Delta N_0)$$

or also written as:

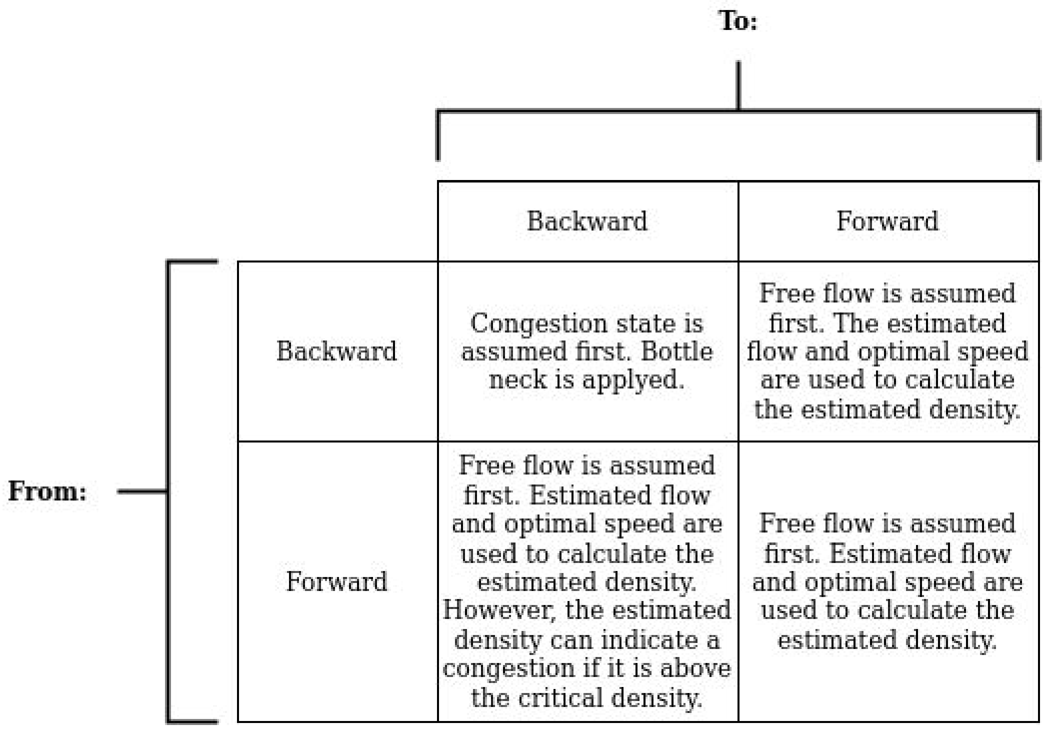
$$k = -Kc(\Delta q_{1_{1\,sec}} - \Delta q_{0_{1\,sec}})$$

In this project, the measured flows take intervals of one hour. For that reason, the estimated flow value and the initial flow value is divided by 3600.

$$k_{new} = k + Kc_0 \cdot (q_{n_{new}}/3600 - q_n/3600)$$

Finally, note that the last formula adapted to the study project takes the critical density at the initial node where the congestion is originated (n=0). Besides, the calculated density is added to the expected density *k in the last formula* because otherwise it would just represent a difference between the first congested node and the iterated node.

For better understanding of the iteration process:



From: \ To:	Backward	Forward
Backward	Congestion state is assumed first. Bottle neck is applyed.	Free flow is assumed first. The estimated flow and optimal speed are used to calculate the estimated density.
Forward	Free flow is assumed first. Estimated flow and optimal speed are used to calculate the estimated density. However, the estimated density can indicate a congestion if it is above the critical density.	Free flow is assumed first. Estimated flow and optimal speed are used to calculate the estimated density.

Figure 11.5 backward-forward iteration process

Software architecture

The program which makes the estimations of the new values is organized in several classes which have a responsibility assigned. The idea behind the architecture is a graph data structure. The data structure is based on the node objects which have some values assigned and contain other objects from the same class type which represent the adjacent nodes. Each node is labelled with its ID and an Integer value which indicates if the node is a forward adjacency or a backward adjacency. This process is run by another object responsible of the performance of the algorithm. Besides, the class ModellingBasis is responsible for the calculation for the initial flow and density models and optimal values. This class uses a utility class which cleans the data by removing null or wrong values. The simulation object encapsulates this process. Finally, a test case was written to run the code associated to this architecture and check the results obtained. The following diagram shows the described architecture.

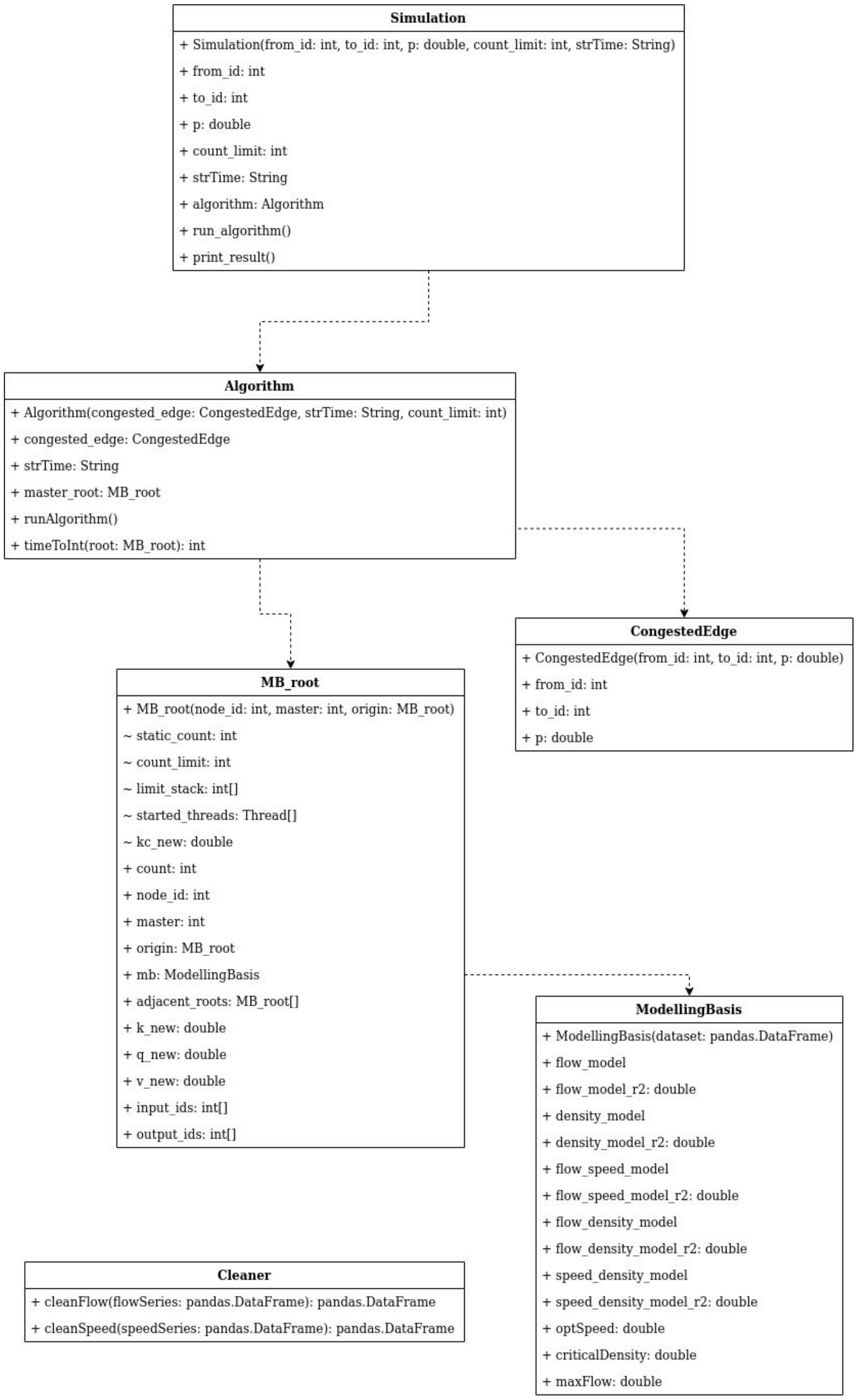

Simulation
+ Simulation(from_id: int, to_id: int, p: double, count_limit: int, strTime: String)
+ from_id: int
+ to_id: int
+ p: double
+ count_limit: int
+ strTime: String
+ algorithm: Algorithm
+ run_algorithm()
+ print_result()
Algorithm
+ Algorithm(congested_edge: CongestedEdge, strTime: String, count_limit: int)
+ congested_edge: CongestedEdge
+ strTime: String
+ master_root: MB_root
+ runAlgorithm()
+ timeToInt(root: MB_root): int
CongestedEdge
+ CongestedEdge(from_id: int, to_id: int, p: double)
+ from_id: int
+ to_id: int
+ p: double
MB_root
+ MB_root(node_id: int, master: int, origin: MB_root)
~ static_count: int
~ count_limit: int
~ limit_stack: int[]
~ started_threads: Thread[]
~ kc_new: double
+ count: int
+ node_id: int
+ master: int
+ origin: MB_root
+ mb: ModellingBasis
+ adjacent_roots: MB_root[]
+ k_new: double
+ q_new: double
+ v_new: double
+ input_ids: int[]
+ output_ids: int[]
ModellingBasis
+ ModellingBasis(dataset: pandas.DataFrame)
+ flow_model
+ flow_model_r2: double
+ density_model
+ density_model_r2: double
+ flow_speed_model
+ flow_speed_model_r2: double
+ flow_density_model
+ flow_density_model_r2: double
+ speed_density_model
+ speed_density_model_r2: double
+ optSpeed: double
+ criticalDensity: double
+ maxFlow: double
Cleaner
+ cleanFlow(flowSeries: pandas.DataFrame): pandas.DataFrame
+ cleanSpeed(speedSeries: pandas.DataFrame): pandas.DataFrame

Figure 11.6 class diagram of the program used in the congestion model

This architecture is then tested using a unit test case which instantiates an Algorithm object, it runs it, and shows the results. It checks if the results are as they're expected.

When using the model just the following code is needed.

```
# values for the simulation
from_id = 903
to_id = 841
p = 0.40
count_limit = 5
strTime = "12:30:00"

# initialize object
simulation = Simulation(from_id, to_id, p, count_limit, strTime)
# run algorithm
simulation.run_algorithm()

# print simulation result
simulation.print_result(simulation.root)
```

e. Back-end API

In order to provide the front-end communication with the client, a back-end API has been built. The API is responsible to provide data from the database. The provided data is a json string object containing the data requested by the client. The API uses a python event-driven networking engine called twisted to enable the communication as a server. Besides, a web framework called *klein* is implementing the engine. The reason why these technologies where chosen is because they allow to implement the API easily and fast and is a good way to build small web servers. Moreover, it enables multiple requests. The problem is that if the API would significantly scale, probable it would need a rebuilt. However, the deadline of this project forced to implement something quick but good enough to handle the requests and supporting the encrypted https protocol.

At first, the database requests were consuming lot of time to retrieve the data. The large amount of data in the Hive database is the cause of the issue. When the client requests the needed data, in order to not wait for to request to be finished, the request is inserted in an *active jobs* list which would contain the request which are being processed. The client as a response to that would get a link which would provide the data once the job is completed. The completed jobs are also stored in a *complete jobs* list which contain all the requests which are already processed. If another request is made by the client, then the API checks if there is a job already completed with the same parameters, and if it's so, it will directly provide the link without waiting for the request to be processed. Note that once a job is inserted in the *active jobs* list, a new thread is started to process the request. This is a way to process multiple requests at the same time.

Check the appendix F for a detailed documentation for the built API.

6. Conclusions

The project has gone through a large analysis process. First understanding the context of the assignment and making a planification with its milestones. Then, researching about the needed data and understanding its structure. Getting adapted with the technologies to deal with big data. Building a data retriever application to get the data from the data source and store it to a database. Investigate about methodologies to make the analysis. Link the nodes of the road network and correlate their data. Perform statistical analysis to the node data. Research on modelling techniques for traffic data. Generate models based on the historical data. Design an algorithm to put everything together and be able to predict from user inputs. Test the models and improve them. Research on ways to visualize data and the obtained results. Research on how to create an api to communicate from the database to a client. Design and implement the api. Finally, summarize all the process and extract conclusions from the results.

The literature research performed in this project is involved in both conceptual and technical areas. Conceptually, a learning process for the understanding of the traffic behaviour was done. This context consists on the macroscopic approach of the traffic analysis done during the project. Working with traffic data as a flow and apply the dynamics theories. Working with one dimensional model. Understanding of the components which form the traffic behaviour and how are they correlated. Looking at the possible optimal values which can be found when making the analysis. Finally, researching on how a congestion originates. For the technical issues the research was done in order to complete the planned milestones and goals. Some methodologies and techniques used are specific for some contexts and research was needed. First to work with geographical data files and to work with xml structured files. Then to acquire skills on the big data field and be able to use the cluster of computers and the software installed. Research was also done to discover which python libraries were suitable for the analysis process.

The data retrieval process was based on building an application to do this task. When designing the application, first was important to understand the structure of the provided data. In one hand, the project went through a learning process of the shape files which contained the measurement locations, and in the other hand it went through an understanding process of the traffic data and the structure of the xml files. It is concluded from the data source, that the shape files provided precise static data for geographical locations. Precise for the exact coordinate system, and static because the data remains always the same. The data accuracy, as an inconvenience, come in high size files and are memory consuming when they are processed.

However, as they are just processed once, it doesn't affect to the performance of the whole process. The traffic data files, as they contain data for all the measurement points in the Netherlands, are big too. The xml format lets them to be processed in a reasonable time. The java spark library provides a good service to accomplish that. The traffic data structure provided by the national databank is designed to let all the developers work with it, independently of the approach of the project. This is seen, for example, looking at the way in which the data is provided. Data for the different kind of vehicles depending on its size is provided separately. This enables to have a flexible data source but making the data retrieval process a bit more complex. The updating system works perfectly, and the time intervals are accurate enough. The documentation provided by the national databank is appreciated for the development process, but the English version of their web site could be improved too. Before performing an analysis, the extracted data had to be cleaned. It contained some wrong values caused by error in the measurements or missing inputs. The data cleaning allowed to work with better data in terms of reliability and accuracy.

The analysis process had two main phases. First a statistical overview of the traffic data and its context and later the modelling. The first statistical exploration is basically used to discover some aspects of the traffic data and its behaviour. At first small statistical summaries were made to check the distribution of each of the components which are used for the traffic data analysis. Then, a time series was made to detect patterns in the data and check how noisy it is. Compare between working days and weekends. The flow components show, in most of the measurement points used, two peaks of traffic activity during the working days. In the other hand, on weekends, these two peaks aren't visible. The time series showed that the traffic activity decreases during the weekend as well. The average speed data showed a noisy period during the night-time and more stable values during the rest of the day. These stable values showed, in a few measurement points, some speed decreases which are cause of congestion. However, the research done tells that to confirm these speed decreases as congestions the traffic density must be check as well. The traffic density must be high enough to confirm that the speed decrease is caused by a congestion. As part of this statistical analysis in can be included the network analysis as well. The network analysis was done to comprehend the road network itself by checking central points of the city area. Moreover, the statistical analysis can be used to compare different measurement points and deduce a type for each one. The second part of the analysis process consisted on the modelling of the traffic data. The modelling provided the optimal values needed later for redirecting the traffic flow from input simulations. Besides, these optimal values give valuable information of the road section as well. From the modelling, several information could be deduced as well. Information such as congestion periods of the road, anomaly detection, number of cars exceeding the capacity of the road, the relation between flow and density, etc. All this information can be used to detect traffic situations like incidents or special events. After getting these optimal values and some predefined models, a general model called congestion model could be made. This model used the previous information to redirect traffic flow and estimate new traffic density values. From the estimated traffic density values, it can be seen if at which points of the road network a congestion will be originated. The model also allows to predict which average speeds are estimated as well from the congestion situation. This last model, however, needs more testing and from now is just kept as something more theoretical. The reason why the model couldn't be tested, is because for that is needed a bunch of confirmed incidents and their impact. In this project it was no time that.

7. Improvements and future analysis

The project has achieved the main goals planned when it was first plan. First, gain experience in the big data field and the technologies to be used. Then, retrieve data from the national databank and store it in a local server for later analysis. Perform analysis on traffic data and build predictive models. Finally, visualize the obtained results. However, some issues occurred during the development process which and some results couldn't be fully obtained. In this section, all the possible improvements and some other future analysis which could be done as a continuation of this internship project will be exposed.

Retrieval of data
The retrieval of data has caused most of the issues, specially at the beginning of this project. The technologies used to run the retriever application and store the results are quite specific. Besides they require a configuration process which sometimes can be a bit confusing. Another problem originated was on the XML structure. Spark provides a java library to read these types of files, but the learning process was a bit slow. However, all the mentioned issues could be solved. Even making the application stable took some time. It used to break down for running out of memory or due to modifications on the XML files structure. The national data bank provides traffic data from different kind of vehicles. As a future improvement, and for other project scopes, the data could be saved considering this classification. The implementation for that could take some time due to the configuration information which must be extracted from another source.

Modelling
This project focussed basically in macroscopic modelling (considering the vehicles as a flow). From this approach many conclusions could be extracted. Optimal speeds in roads, maximum amount of cars per minute the road can carry, maximum number of vehicles per kilometre which the road can carry as free flow, the number of vehicles which should be removed to make the roads less congested, the most and the least congested points in the road network, anomalies in traffic data, identifying points by congestion peaks, etc. However other approaches could bring even more information to the study case. As an improvement of this project, modelling on specific road sections could be applied, and by checking delays, the time a congestion takes to end could be estimated.

Congestion model

The congestion model is an implementation of the modelling previous done. It is based on traffic redirection and it assumes that the traffic redirects uniformly. This assumption could sometimes differ from the reality because there exists an unpredictable human component which can decide not to take certain directions when there is a traffic congestion. Nevertheless, the model estimates the situation in the worst case so the predictions can be made knowing that the congestion won't be worse. From the improvement which could be done in the modelling by estimating traffic delays, another improvement could be made in the congestion model. By substituting the parameters inserted by the user which sets the graph depth by the time after the congestion was originated. From that time the number of affected nodes could be calculated first and then the algorithm would run as it is right now.

Database API
The visualization of the results was planned to be in a web application format. This couldn't be fully developed during this project, so the database api just followed the same development process. As a continuation of this project, more visualization features could be provided by the web app and more developed could the API be.

8. References

Weather conditions source (time and date):
https://www.timeanddate.com/weather/netherlands/utrecht/climate

Short term traffic prediction models (C.P.IJ. van Hinsbergen, J.W.C. van Lint, F.M. Sanders)

Traffic flow theory and modelling (Serge Hoogendoorn and Victor Knoop)

Macroscopic traffic flow modelling (Victor L. Knoop)

Study of the US Road Network based on Social Network Analysis (Elie Ngomseu Mambou, Samuel Nlend and Harold Liu)

Dutch railway handles 1.3 million passengers per working day. By Janene Pieters on July 4, 2019 on NLTIMES.nl

Traffic Flow analysis Wikipedia: https://en.wikipedia.org/wiki/Traffic_flow

Traffic flow fundamental diagrams Wikipedia:
https://en.wikipedia.org/wiki/Fundamental_diagram_of_traffic_flow

NDW official site: https://www.ndw.nu/en/

NDW data documentation: http://docs.ndwcloud.nu/en/avg/reistijden/reistijden.html

Apache Spark official site: https://spark.apache.org/

Apache Official site: https://hive.apache.org/

QGIS documentation: https://www.qgis.org/en/docs/index.html

Learning Spark, LIGHTNING-FAST DATA ANALYSIS (Holden Karau, Andy Konwinski, Patrick Wendell & Matei Zaharia)

Geotools documentation: https://docs.geotools.org/

NetworkX documentation: https://networkx.github.io/documentation/stable/

Matplotlib official site: https://matplotlib.org/

Sklearn official site: https://scikit-learn.org/stable/

Pandas official site: https://pandas.pydata.org/pandas-docs/stable/

Numpy official site: https://numpy.org/

Appendix A - Data source

The traffic data source is the Nationale Databank Wegverkeersgegevens (NDW). It is an organisation that maintains a database of both real-time and historic traffic data. It allows to control better the CO2 emissions, traffic congestion and the security over the Dutch roads.

The road authorities use this data to manage better the traffic flow. In this project, the source of data is used to generate the appropriate graph of the Utrecht road network and to do the necessary statistics to build models.

The data comes in a xml format. There are two kinds of files. One containing configuration information for each of the measurement points, and another one containing traffic data. The one containing traffic data provides flow data and speed data per measurement point. The data comes divided by vehicle type. However, in this project, as a macroscopic analysis is performed, the flow data taken is the sum of all the input types.

To build the graph, just the information about location is taken (latitude and longitude). Each location taken from the xml represents a measurement point in a Dutch road. Then, the locations extracted which are in the area of Utrecht are selected.

NDW also provides real-time situational data. Any breakdown or accident or any kind of issue is reported through a messaging system. The exchange of the situational messages is based on Datex II. DATEX II is the electronic language used in Europe for the exchange of traffic information and traffic data.

Appendix B - Geotools and the shape files

Geotools is a java library made to work with geometry issues and is oriented for geography utilities. It allows to load shape files containing coordinates in a specific coordinate system. It provides an accurate service for calculations related with geographic space. According to the official web site:

GeoTools is an open source (LGPL) Java code library which provides standards compliant methods for the manipulation of geospatial data, for example to implement Geographic Information Systems. The GeoTools library data structures are based on Open Geospatial Consortium (OGC) specifications.

Appendix C - QGIS

QGIS is a desktop application designed to create, edit, visualize and analyse spacing data. Uses the geographic information system (GIS) to provide the necessary tools to work with geographical data. It enables to load shapefiles and inspectorate them. In this project it was necessary to be used to work with the spacing documents. The shape files are formed by geometry object which can be either polygons, points or lines. It is an open-source cross platform and allows to include Python and C++ plugins.

Appendix D – Hadoop technologies

In order to process and store large amounts of data, clustering computing is needed. Clustering allows to process the data partitioned and distributed over different nodes. Each node is a computer that simultaneously processes its data partition with the other nodes. This parallel computing system can be implemented by a software tool called Apache Spark. Spark is a lightning-fast cluster computing technology designed for big data analysis. It provides an interface for programming clusters with fault tolerance. Moreover, Spark facilitates the implementation of certain algorithms used in data analysis. It includes a stack of libraries which can be combined. This allows to do different kind of analysis using one single software system. This is one reason why apache spark is chosen. Another reason is its speed processing for large datasets. Besides, it offers APIs in different kind of programming languages. In this project, the program used to retrieve data from the national data bank will be developed using the Java org.apache.spark library. So, the coding needed with be based on Java.

The data retrieved is stored in another hadoop based technology called Hive. Hive is a database which provides data warehouse facilities such as reading, writing, and managing large datasets. With Hive, the querying can be done using SQL, and the connection to the database using JDBC drivers.

Appendix E – Python libraries

In this project, for the data analysis many python libraries are used. In this section they will mentioned explaining their main utility.

NetworkX is a python library made for network analysis. It provides a large list of algorithms which can be applied to any generated graph. It has its own data structure for storing graphs and it can provide different representations of it as well. Moreover, it provides some visualization features for the graphs. In this project networkX is used for the network analysis of the rod network.

Pandas is a python library made for easily use data structures with high-performance and efficiency. It includes many functionalities to load data and structure it. I enable to apply some statistical calculations on the data structures as well. In this project it is used to store structured data, concatenate data frames and process them.

Numpy is a fundamental package for scientific computing. It provides N-dimensional array data structure. Besides it allows some algebraic calculations. In this project it is used to stored data in the numpy array to later process it by other libraries.

Matplotlib is a 2D plotting library which provides graphic functionalities to represent data plotted in charts. It accepts flexible data inputs and allows to customize the visualization by colouring and labelling the output. In this project it is used to represents data and visualize the analysis.

Scipy.interpolate, scipy.optimize, scipy.interpolate and scipy.misc are libraries which provide mathematical functionalities such as function optimization, integration derivation and interpolation. In this project they are used modelling purposes.

Sklearn is a python library which provides extended predictive tools for data analysis such as classification algorithms, regression performance, clustering etc. It focuses in supervised learning. In this project is mainly used for regression purposes.

Appendix F – Data cleaning

The data obtained from the national databank may contain some null or zero values which must be cleaned in order to work with reliable data. This process is fundamental because if it is not done all the analysis will be influenced by the wrong values and some wrong conclusions could be taken. The cleaning process is implemented on the flow and speed values. However, both processes are a bit different from each other.

Flow data cleaning

The flow data values which are wrong are those which take a zero value at a time of the day when is supposed to be higher. It is important to detect when those values should be higher or not because during the night-time the flow often becomes zero and it is not because the values are wrong. The flow data might have wrong values due to many circumstances. Error in the measurement, bad internet connection of the server retrieving the data, bad structure of the provided data files etc. Nevertheless, it is not often to find many of them on the studied domain. If it is the case, then it should be considered that the whole domain is corrupted or not valid for analysis. So, to do the cleaning process is fundamental to look at the previous values of the corrupted one. If those values are high enough to assume that the analysed values should be high as well, then an average value is assigned to it. The average value assigned will depend on how high the value of the previous measurement is. Depending on that the assigned average will be done from different quantiles. There are for quantiles for that, and just the three highest are used for the data cleaning.

Here it follows an example of the cleaning result.

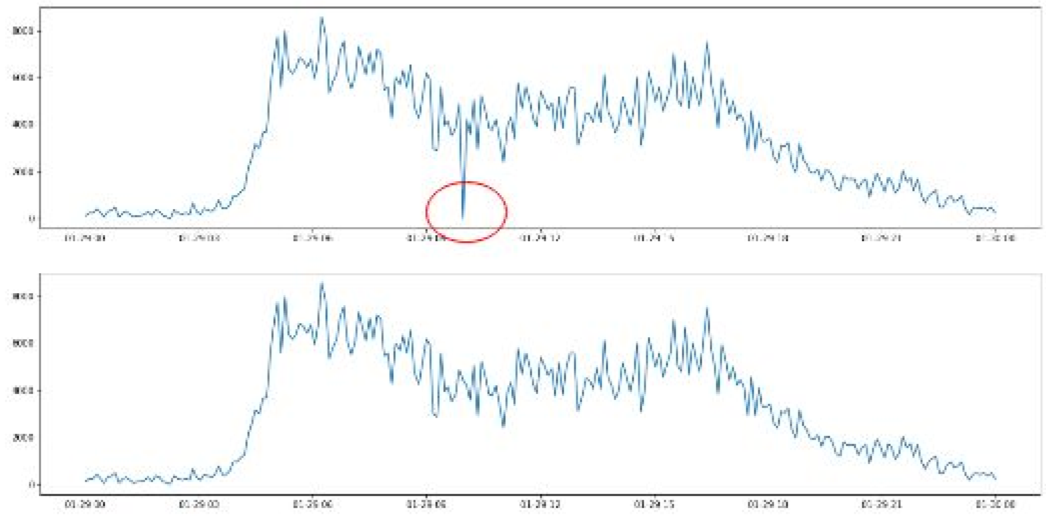


Figure E.0 flow data cleaning

On the top line chart, the highlighted value is clearly wrong so it is corrected as it can be seen on the bottom chart.

Speed data cleaning

The measured speed data is an average value of an interval of one minute. If there are no inputs during that interval, the average cannot be making. In that case, the national databank sets the speed value at that time to minus one. The cleaning process will detect these values and will assign an average value for that measurement. The average value assigned is an average of all the speed data without considering the minus one values.

Here it follows an example of the cleaning result.

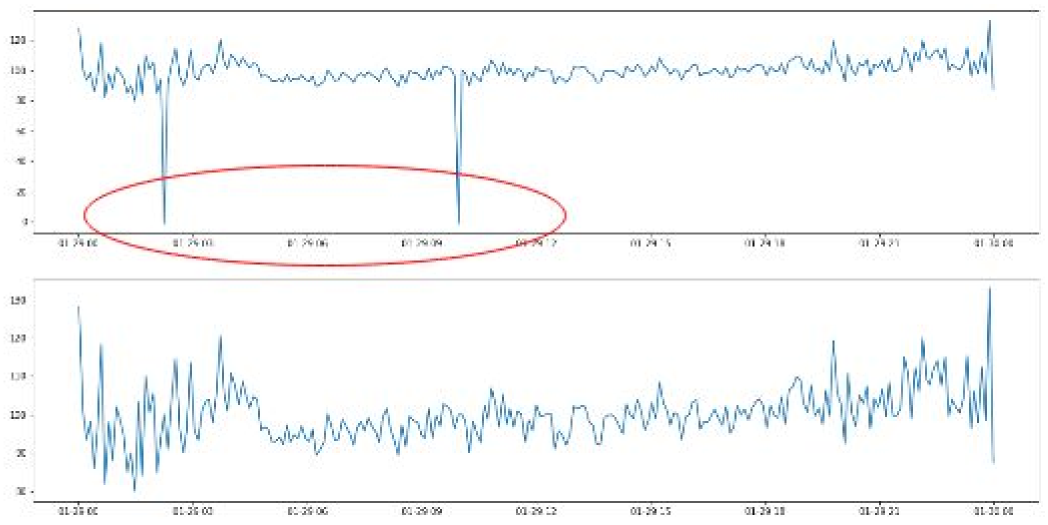


Figure E.1 speed data cleaning

On the top line chart, the highlighted value is clearly wrong so it is corrected as it can be seen on the bottom chart.

Appendix G - Back-end API documentation

getActiveJobs()

• Returns a list of the jobs which are being processed.

Each element in the list contains:

reference: String
type: String
link: String

getCompleteJobs()

• Returns a list of the jobs which are already processed.

Each element in the list contains:

reference: String
type: String
link: String

getJsonResult()

• Returns the json string of the referenced request.

• The structure of the json will depend on the request.

getReferences()

• Returns a json object which represents the nodes in the road network containing a list of references and ids.

Each element in the list contains:

reference: String
id: int

getEdges()

• Returns a json object which represents an edge list containing two ids (head and tail nodes) and the distance between the nodes.

Each element in the list contains:

fromId: int
toId: int
distance: double

The following requests will be inserted in the Active Jobs list and will remain there until they are fully processed. Once a job is completed, it is stored in the Complete Jobs list. Then, the request getJsonResult will provide the json string.
All of the following request will return a json string containing the reference of the requested point, the type of data requested and the link to the json.

getMeasurementPoint(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object which represents the traffic data of the referenced measurement point during the given time period containing a list of the dates and times when the measures were taken, the flow values and the speed values.

Each element in the list contains:

measurementdatetime: String
flowvalue: double
speedvalue: double

getFlow(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node

• Json object which represents the flow values of the referenced measurement point containing a list of the dates and times when the measurements were taken and the flow values.

Each element in the list contains:

measurementdatetime: String
flowvalue: double

getSpeed(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object which represents the speed values of the referenced measurement point containing the list of the dates and times when the measurements were taken and the speed values.

Each element in the list contains:

measurementdatetime: String
speedvalue: double

getDensity(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object which represents the density values of the referenced measurement point containing the list of the dates and times when the measurements were taken and the density values.

Each element in the list contains:

measurementdatetime: String
densityvalue: double

getNumericSummary(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object containing the mean and the standard deviation of the flow, speed and density of the given measurement point within the given time interval.

flowMean: double
flowStd: double
speedMean: double
speedStd: double
densityMean: double
densityStd: double

getFlowTimeSeries(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object containing an array of arrays with the date and time when the measurement was taken, and the four components of the flow time series. The observed data, the trend component, the seasonality and the residual of the series model.

Each element in the array contains:

measurementdatetime: timestamp
flowvalue: double
trend: double
seasonal: double
resid: double

getSpeedTimeSeries(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object containing an array of arrays with the date and time when the measurement was taken, and the four components of the speed time series. The observed data, the trend component, the seasonality and the residual of the series model.

Each element in the array contains:
measurementdatetime: timestamp
speedvalue: double
trend: double
seasonal: double
resid: double

getDensityTimeSeries(reference: String, start_date: String, end_date: String)

• Starts a new job to generate a json string as it follows.

• As parameters needs the reference of the node, and the start and the end date of the set of measurements taken at that node.

• Json object containing an array of arrays with the date and time when the measurement was taken, and the four components of the density time series. The observed data, the trend component, the seasonality and the residual of the series model.

Each element in the array contains:

measurementdatetime: timestamp
densityvalue: double
trend: double
seasonal: double
resid: double