SAPIENZA
UNIVERSITÀ DI ROMA

Real-Time Predictive Traffic Signal Synchronization through VANET Communication

Faculty of Civil and Industrial Engineering

Master's in Transport Systems Engineering

Marreddy Vivek Reddy

Matricola 1709256

Supervisor

Prof. Gaetano Fusco

Assistant supervisor

Chiara Cholombaroni

A/A 2017/2018

Acknowledgements

I feel like it is difficult to address all the people who have helped me these past two years in Italy, the rewarding experience it has been. First, I thank prof Fusco Gaetano with who it has been my privilege to work with. It is my honour to have him as my mentor during my research. It was his lessons of traffic engineering during second semester I had grown passion about traffic engineering

All of this would not be possible if it was not for Stefano Ricci. I sincerely thank him for believing in me and giving me an opportunity for studying at Sapienza University of Rome. The past few years have been a life changing experience for me in Italy. I thank Chiara Cholombaroni, Mattia Giovanni Crespi, Antonio Musso, Guido Gentile, Massimo Guarascio, Gabriele Malavasi, Luca Persia and Liana Ricci for their teachings, support and their cooperartion. I am truly grateful.

I thank my friends Eyad, George, Lorenzo, Nicola, Houshang, Federico, Marj, Amal, Agostino, Basak , Jobin, Jinjin, Alessandra, Matteo and Lory for their friendship, support and their encouragement. I thank Roberta and Natalia for sharing their office space with and guiding me during my research work for the past five months.

I cannot find the right words to describe my sister and my brother-in-law for their commitment and the help that I have received from her during my studies. I am indebted to my parents for having supported me in every way they could to help me finish my master's in Italy.

Table of Contents

Introduction

This project presents various simulations in SUMO and gives an insight on how SUMO works according to the instructions given by the user. There is a strong demand for transportation in urban areas. The major part of the demand in the transportation is covered by the public sector. However, public transport sector cannot cover all the demand in the urban areas. Individual driver's makeup the major part of the transportation. The traffic conditions in the city can be improved by various methods. Infrastructural modifications are not possible because of their cost effectiveness. The other way to control the the traffic is, by the use of traffic signal which are adaptive that recognize the vehicles on the network in real time and make adjustments to tackle certain traffic problems such as traffic jams and high delays.

There are two types of traffic signal control they are fixed time cycle control system and dynamic cycle control system. The phase timing plane of fixed time cycle controlling utilizes a predetermined cycle time. The dynamic cycle control system decides and modifies phase based on the traffic flow or density near the intersection. The detection of the vehicles is carried out by the V2I communication. The induction loop detector or lane area detector on the network are responsible for detecting the traffic flow density and means speed of the vehicle on the network. The traffic model with dynamic control system is simulated in SUMO- Simulation of urban mobility software. The networks were created on NETEDIT software which is available with SUMO. These network models are calibrated with the field data which is taken from the streets of Rome. Then this model is simulated, and same parameters are applied to another network to check the consistency of the model. The performance of traffic flow through consecutive intersections depends upon the coordination between them. The main goal of this system is to minimize the time lost by the vehicles running along the streets by real time traffic signal synchronization through VANET communication. The methods used in this report can be used as a starting point to start simulation in the SUMO. Additional research may be required to implement the methods provided in this report. The performance is analysed in terms of mean timeloss , travel time, mean CO_2emissions and mean speed. The investigation and proposals in this report are a result of experience from traffic engineering and microsimulation models output. The chapters in this report are arranged in a chronological order. The last section appendix gives the insight of the traffic simulations in SUMO.

1.State of the art

1.1 Traffic characteristics

In the event that we need to think about the Traffic attributes and their expectations, it is appropriate to furnish at any rate their short posting with definitions and shared relations. The three fundamental characteristics in traffic flow theory are flow, speed, and density.

1.2 Traffic flow

Traffic flow can be additionally found in some literature named as flow, flow rate or volume. Nevertheless, all these terms can be used interchangeably. It is characterized as various vehicles passing a point in a given timeframe. It is typically communicated in units of vehicles per hour (vph). Other conceivable units are for example vehicles per hour per lane (vphpl), passenger car units per hour (pcu/hr), or passenger car units per hour per lane (pcphpl or in easier readable form pc/h/ln). The traffic flow counting equation is simple: where q is traffic flow (vph), n is number of vehicles passing a spot on the road in a given interval t. The special value of traffic flow is capacity c which defines the maximum hourly rate under prevailing roadway conditions. Since it is convenient to measure traffic flow in 15-minute intervals, a quantity called peak hour factor (PHF) is presented: Hence is obvious that PHF can theoretically be in the interval $0.25 \leq PHF \leq 1.0$. Peak hour factor can be understood as an indicator of flow fluctuations within the hour. Introduction | 10

1.3 Speed

In light of various methodologies how to figure speed, there are two primary understandings they are time-mean speed and space-mean speed. Time-mean speed is defined as the average speed of vehicles passing a spot on a road. The standard notation is ut.

$$U_s = \frac{\sum_{i=1}^{n} u_t}{n}$$

where:

u_s is time-mean speed in km/h,

u_t is speed of vehicle i measured at a point along a road, usually by a radar or laser gun (km/h),

n is number of vehicles.

Space-mean speed is defined as the average travel speed of vehicles between two points at the distance D apart. It is computed as:

where:

u_s is space-mean speed,

D is the distance of two points on the road (km) is the average travel time (h).

Space-mean speed is more valuable with regards to traffic analysis and is resolved based on the time vital for a vehicle to move along some known length of a roadway. Consequently it is likewise meant basically as u.

Space-mean speed gives more accentuation to high ui, hence us ≤ ut. The uncommon estimation of speed is free stream speed (FFS or uf) which characterizes the spacemean speed

on the specific piece of the street which is come to by an unhindered movement stream under winning roadway conditions. HCM2010 characterizes FFS as the mean speed of traveler autos working in stream under 1 000 pc/hr/ln. FFS is controlled by street geometry, cross area, nature of street surface, and every single mental factor having effect on drivers.

1.4 Density

Density is characterized as the quantity of vehicles per unit length of roadway at any instant. Density is usually denoted by k and its unit is vehicles per kilometre (veh/km) or vehicles per kilometre per lane (veh/km/ln). It is expressed as

$$k = \frac{n}{D}$$

where n is the quantity of vehicles involving length D of roadway at some predetermined time. It is difficult to measure density directly unless there is an option of aerial photography or satellite imaging. Along these lines it is all the more regularly evaluated in a roundabout way by estimating the inflow and surge of vehicles at a street segment after some time, however the underlying state must be known all things considered. The special case of density is so called jam density (kj), which is the maximum possible density on the roadway when the speed of the flow is nearly zero.

1.5 Relationships among traffic characteristics

If there is a requirement of analysis of traffic conditions, the macroscopic approach is used. The fundamental equation describing average conditions on a given link for a specific time period is:

$$q = u * k$$

The equation assumes that the flow is uninterrupted and stable, i.e., all travelling at about the same speed. The fundamental diagram in the presented form assumes a linear speed-density model. That assumption is represented by Greenshields's traffic stream model. However, it is not the only traffic stream model which can be used. Even their combinations creating multi-regime models might be applied. The benefit of using a linear representation of the speed-density relationship is that it provides a basic insight into the relationships among traffic flow, speed, and density interactions without clouding these insights by the additional complexity that a nonlinear speed-density relationship introduces. However, it is important to note that field studies have shown that the speed-density relationship u = f(k) tends to be nonlinear at low densities and high densities. In fact, the overall speed-density relationship is better represented by three relationships: (1) a nonlinear relationship at low densities that has speed slowly declining from free flow value u_f, (2) a linear relationship over the large medium density region, and (3) a nonlinear relationship near jam density kj as the speed asymptotically approaches zero with increasing density. (Mannering, & Washburn).

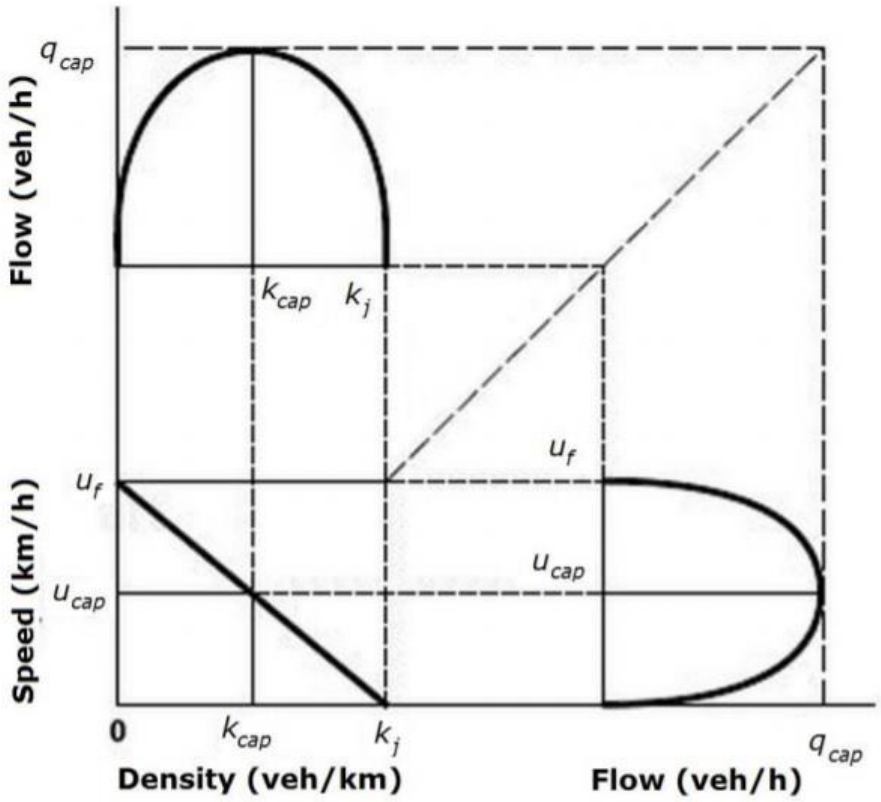


Figure 1Flow-density, speed-density and speed-flow relationships(Mannering and Washburn)

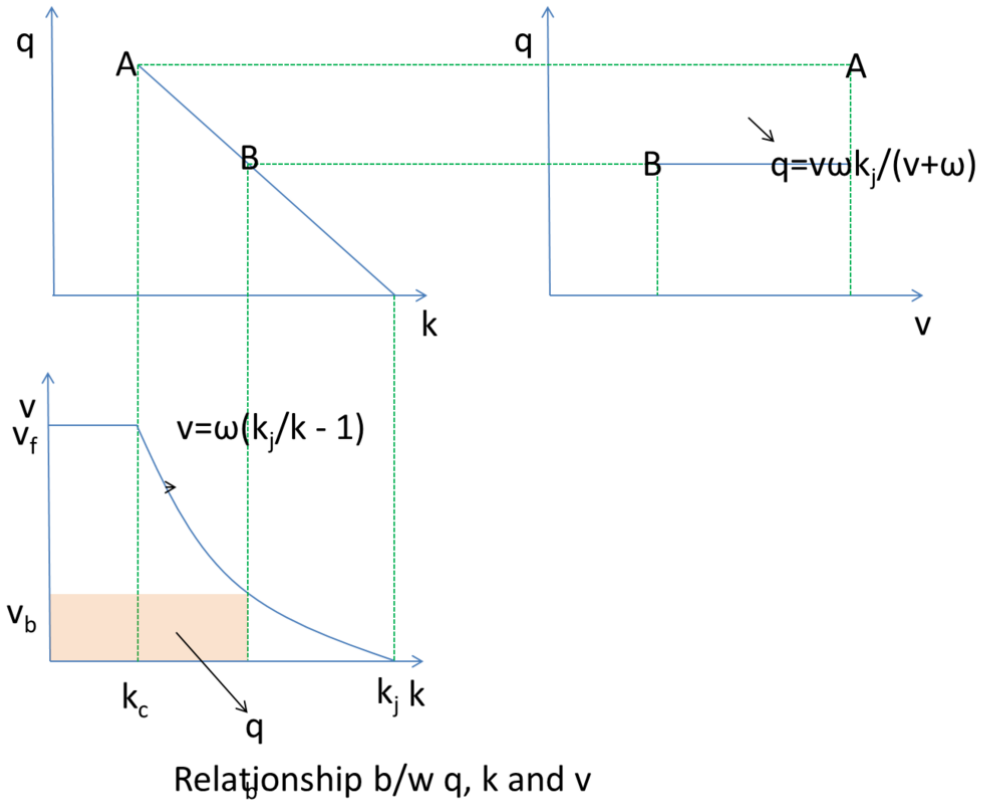


Figure 2 Flow-density, speed-density and speed-flow relationships(Wikipedia)

Abbreviations

K density

k_j jam density

q traffic flow

PHF peak hour factor

FFS free flow speed

1.6 Travel time

It is important to know that speed flow and density are not the only ones what can be measured or computed in the traffic flow assignments. One of the characteristic which is perceived by the end users (i.e. drivers) and affects their decisions about their trips is travel time (TT). TT is defined as a time taken by a vehicle to traverse a given section of a highway with length D. A valuable estimation of movement time may diminish cost impacts by a plausibility to keep away from joins with blockage and enhance a nature of carriage organizations by conveying their shipments in the required period of time. The estimation of TT at a specific direction isn't influenced just by a street geometry, yet in addition by factors particular for the specific moment of the trek. These components are for instance real atmosphere and state of the carriageway, and doubtlessly additionally genuine level of administration in view of movement conditions. It is said as a rule that movement time is influenced by individuals, vehicles, and offices. There is a few alternatives how to quantify TT. The main gathering incorporates ways which record going of an auto through estimating focuses. While strategies from the second gathering use estimating tests moving in rush hour gridlock stream which record their advance. Cases of the main gathering may be tag strategy in any conceivable variation. It discovers distinction in entry time for vehicles touching base in focuses An and B. The suspicion for getting a pertinent outcome is that the timekeepers at the two focuses are synchronized. It should be possible physically by eyewitnesses in its fundamental shape. The more typical way is programmed coordinating of some ID of autos. As an ID can be considered plates read by sensors of programmed vehicle recognizable proof frameworks observing the stream or labels utilized as a part of electronic toll gathering frameworks (locally available units). A calculation of TT in light of information from implanted locators is another illustration having a place among these techniques. Presentation | 14 The second gathering utilizing a moving eyewitness is all the more intriguing in the terms of this theory. These strategies are for the most part in light of utilizing gliding autos as indicators.. In view of activity stream hypothesis and level-of-benefit idea, as movement stream builds, speed diminishes, and in this way travel time increments. There have been produced a few connection execution capacities which exhibit a numerical connection between course travel time and course of traffic stream.

HCM2000 provides Akcelik Delay Function:

$$t = t_0 + D_0 + 0.25T\left[(x-1) + \sqrt{(x-1)^2 + \frac{16JXL^2}{T^2}}\right]$$

Where:

T is expected duration of demand

L is link length

D_0 is the control delay

X is link demand to capacity ratio (v/c)

J is calibration parameter (-)

Federal highway authority uses its BPR function:

$$t = \frac{L}{S}\left[1+\propto \left(\frac{v}{c}\right)^{\beta}\right]$$

Where:

T is time of a link,

L is the length of the link,

S is the free flow speed

V is link traffic flow (volume for which costs are computed) of a link,

C is capacity of the link,

α, β are calibration parameters (typically α=1,β=4)

1.7 Car following Model

Car following model describes the behaviour of the vehicle in a traffic flow on a lane. The following car maintains a safe distance from the lead car to avoid collisions (Gipps, 1981; Krauss, 1997; Treiber et al., 2000). The acceleration of a car depends upon the speed of the car before it(Gazis et al. 1961).There are different parameters that are used to describe various car following models. Acceleration is one of the parameters that is used to specify the acceleration of the vehicles. There is another parameter named desired speed this is the parameter described the drivers desired velocity. The driver accelerated until he reached the desired speed. The reaction time is the time taken by the driver to react to the changes in his environment, this parameter is usually taken as 1 sec as a standard value. Reaction times are also used in lane changing models. Desired headway is the space maintained by the follower from the leader. This is the taken by the follower to reach the leader's same level of deceleration at same speeds in braking phase. The headway will be short if the driver is more aggressive. The minimum headway is independent of speed.

1.8 VANET

VANET stands for vehicular Ad-Hoc network. VANET is a bit of MANET(Mobile Ad-Hoc Networks). An Ad-Hoc network is a wireless network that is decentralised. Every vehicle acts as a node in Ad-Hoc network. Vehicle movement is limited to the roads and traffic rules on these roads, in order to have a continuous and reliable communication between the vehicle certain fixed infrastructure is deployed they are road side units mesh routers etc. Road side unit connects the vehicles to the infrastructure. There are two types of communications in VANET they are vehicle to vehicle communication and vehicle to infrastructure communication. Vehicles have some sensors inside them which detect the conditions on the road and convey this information to the authorities. So, authorities take necessary action in case of an accident or traffic jam. Vehicles provide various information to other vehicles and infrastructure

such as traffic density, road condition and vehicle speed. In the VANET every node stays connected and can freely move within the network coverage.

The nodes are mobile in VANETS and MANETS. The vehicle movement (node movement) in VANET is limited to the road infrastructure. VANET is an emerging technology that can integrate new generation wireless networks to nodes(vehicles). VANET utilizes Dedicated Short-Range Communication(DSRC), with cellular network, WIFI, satellite and WiMAX (Worldwide Interoperability for Microwave Access). The major application of VANET is ITS.VANET improves traffic efficiency and road safety.

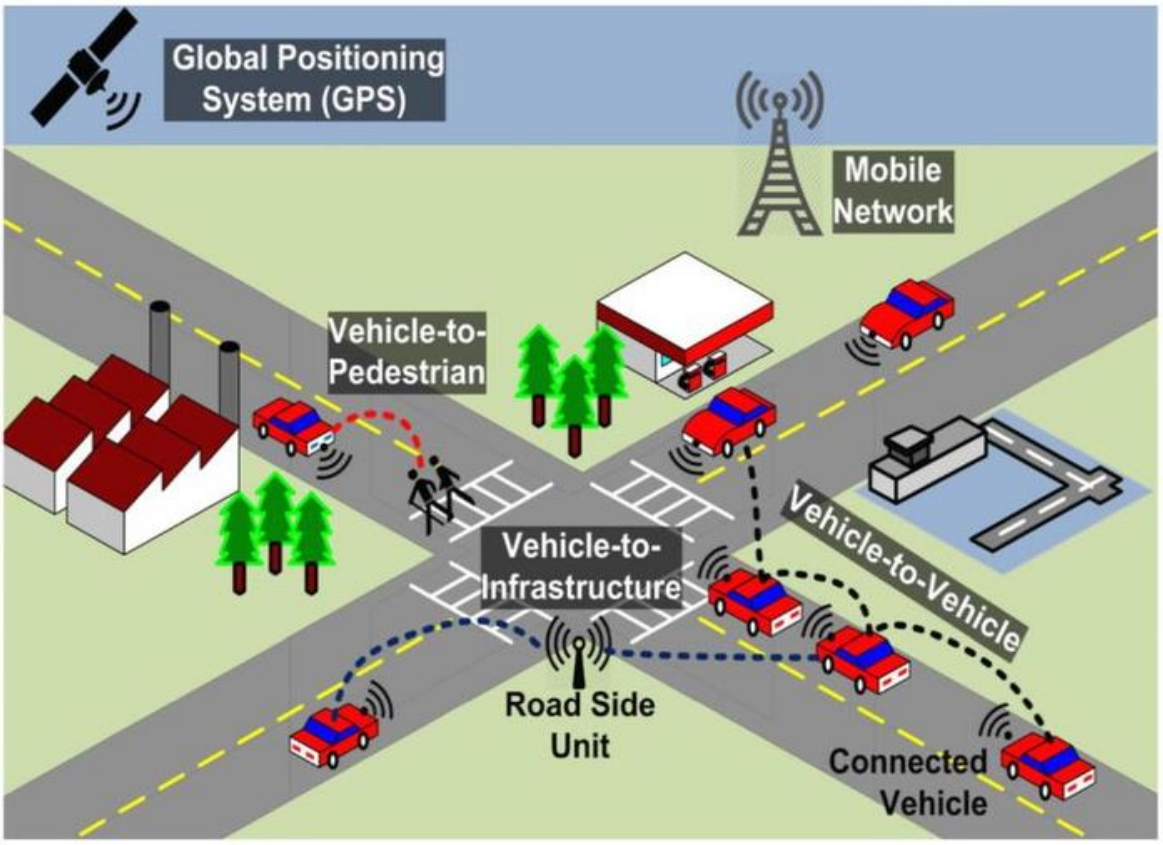


Figure 3:VANET

The vehicles are equipped with onboard communication devices that are responsible for communication with the road side unit. The vehicles receive the information about their position from the GPS signal and convey it to the Road Side Unit. This communication helps in determining the flow, density and speed of the traffic and the authority can take necessary actions to improve these characteristics.

1.9 Signal Priority

Signal priority is an operational strategy which facilitates the movement of the prioritized vehicles through traffic signal-controlled intersections.

Figure 4: Signal Priority at intersections a simplified representation

The general steps involved in signal priority are as follows

The car approaching the intersection is detected at some point P_d upstream of the intersection by various detection methods.

The systems identifies the position of the vehicle with gps and notifies the controller that this vehicle would like to receive priority at the intersection. The request will be processed by the sytem and grants priority according to the conditions defines . The traffic controller C then gives the priority according to our strategy. Typically, if the intersection signals are already displaying a green phase for the approach being used by the priority car, the controller will extend the length of the green phase to enable the bus to pass through the intersection on that phase. If the intersection signals are displaying a red phase on the prioritized car approach, the controller will shorten the green phase on the cross street to provide an earlier green phase for the car approach.

When the car passes through the intersection, clearance is detected by the priority detection system Pc and a communication is sent to the traffic controller that the bus has cleared the intersection.

On being notified that the car with the priority has cleared the intersection, the controller C restores the normal signal timing through a predetermined logic.

1.9.1 Active priority

Active priority detects the vehicle that the priority has to be given and gives the priority to this vehicle by modifying the traffic signals in order to give priority to this system. There are different types of strategies which can be used within the traffic control environment to give priority. They are described below.

When a prioritized vehicle is approaching the intersection when the traffic signal of that artery is about to turn red but the traffic light is still green then a green extension strategy can be used to prolong the green time. This green extension is one of the most effective forms of signal priority since a green extension does not need any additional clearance intervals, yet allows a prioritized vehicle. This strategy reduces the waiting time time of the prioritised vehicle instead of waiting for an early green.

2.Simulation

The process of replicating real world in a computer model is called simulation. The models is used to study the behaviour of the components interacting in it. In the case of traffic simulation Infrastructure and traffic using this infrastructure are the components. Streets, traffic signals, railway lines, induction loop detectors etc components include infrastructure while traffic comprises of buses, trucks, pedestrians, bikes, passenger cars, trains and street cars. Traffic system performance is usually assessed by different measures of effectiveness (MOE) like queue length, system throughput, travel time, delay, etc. There are three types of traffic simulation which are based on the level of detail – macroscopic, mesoscopic and microscopic. A macroscopic model is the one in which traffic is considered as a stream and the level of detail is low. A mesoscopic model has a moderate level of detail with less interaction between the individual vehicles. The highest level of detail is given by the microscopic models and the vehicles are simulated on the basis of second by second. There are various parameters used in the microscopic models such as acceleration, deceleration, gap between the vehicles, reaction time and reaction time to make the interaction of vehicles in models as realistic as possible. Due to their ability to simulate individual vehicles in the network, only microscopic models are able to simulate transit signal priority applications at the individual intersection level. So, in this study we use the microscopic modelling. Simulation helps to develop a new system or redesign an old system by testing it on the computer before applying it on the field. It also can be used to test different scenarios which cannot be tested in real life such as transit signal priority in peak periods on urban arteries. Gives a practical method for testing and assessing distinctive situations Simulation is a fast way to test traffic scenarios than real life offers a knowledge into the qualities of traffic system operations that are vital, enabling the user to settle on an optimal choice.

2.1 Simulation tools

There are many types of simulation tools available in the market. SUMO was chosen because of its flexibility and availability. SUMO cannot make logical decisions during real time simulation. So, python has to be used along with SUMO. Python interacts with SUMO via TCP/IP as it get the data during real time traffic simulations and gives necessary commands to SUMO. SUMO acts as a server during the simulation.

2.1.1 SUMO

SUMO was developed at the German Aerospace Centre. SUMO is microscopic simulator, which means every vehicle is simulated. The vehicles stop for traffic lights, traffic stops, give way to high priority vehicles at an intersection while following their route. In this simulator vehicles can move freely, collision between vehicles and accidents are also simulated. Each vehicle has its own road and vehicle routing is dynamic. The vehicle behaviour is taken into consideration such as changing lanes during emergency deceleration etc. Roads in SUMO are shown as a plurality of lanes. Also, the vehicle width is fixed. And it does not take into account the different types of vehicles. SUMO allows modelling of intermodal traffic systems including road vehicles, public transport and pedestrians. SUMO software is famous for its ability to be manipulated at source code level. SUMO can be enhanced with custom models and provides various APIs to remotely control the simulation. Detectors can be used in the simulation of

SUMO to count the number of vehicles passed through a particular section. SUMO writes the output of a simulation into an xml format file. There are different types of outputs given by SUMO. The output syntax has to be written in the configuration file along with the input files. This output file can be opened in excel and the results can be seen over there. There are three files that are essential for a simulation run in SUMO. They are net file where the information about the network is stored, a route file where the information regarding the attributes of the car and the types of flows used in the simulation and the configuration file which creates the configuration for the simulation. This simulation can be run in SUMO-GUI for the graphical representation. The data flow in the simulation is shown in the figure below. SUMO can be used to simulate railway lines too. There are various types of simulation that can be carried out on SUMO such as bicycle, pedestrian, public transport, railway and waterway. Basic computer skills are required to operate the SUMO simulation software. Netgenerate is a c++ program used in SUMO to generate abstract road networks that can be used in the SUMO simulation software. In SUMO the networks can be imported from various programs such as OpenStreetMap, VISUM, Vissim, OpenDrive, MATsim and DlrNavt eq. SUMO has some additional features such as Emissions, Electric Vehicles, Logistics, wireless device detection and emergency vehicles. Emergency vehicle over take the other vehicles and exceed the speed limit by 150%. The latest version of sumo is 0.30.0 which was released on 17/12/2017.

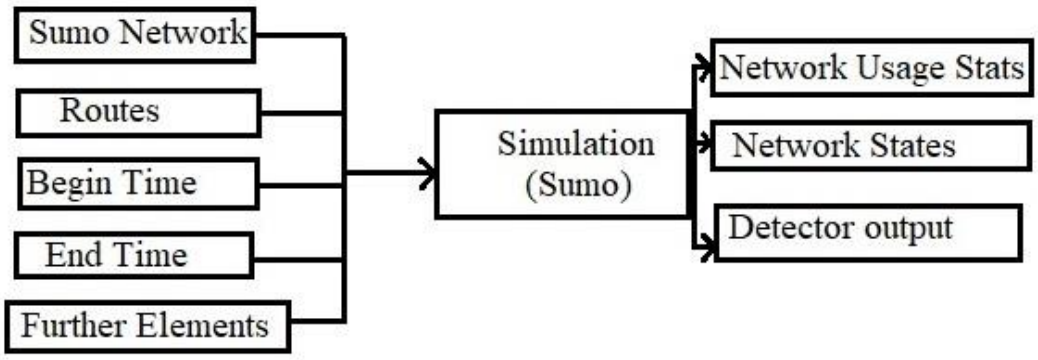


Figure 5: SUMO data flow

Sumo generates output files in XML-format by default. There are various types of output in SUMO such as vehicle-based, output from simulated detectors, network-based information and traffic lights-based information. There are five types of simulated detectors in SUMO. These simulated detectors include inductive loop detectors. SUMO can simulate various modes of transport and various types of vehicle, but a simple passenger car was chosen in this thesis for the simulation. In SUMO network can be built by NETEDIT tool from the scratch or network can be imported from the OSM website. This can be converted to .net.xml format by NETCONVERT tool in SUMO. The network file can be combined with polygon file and edge files to better visualize the network.

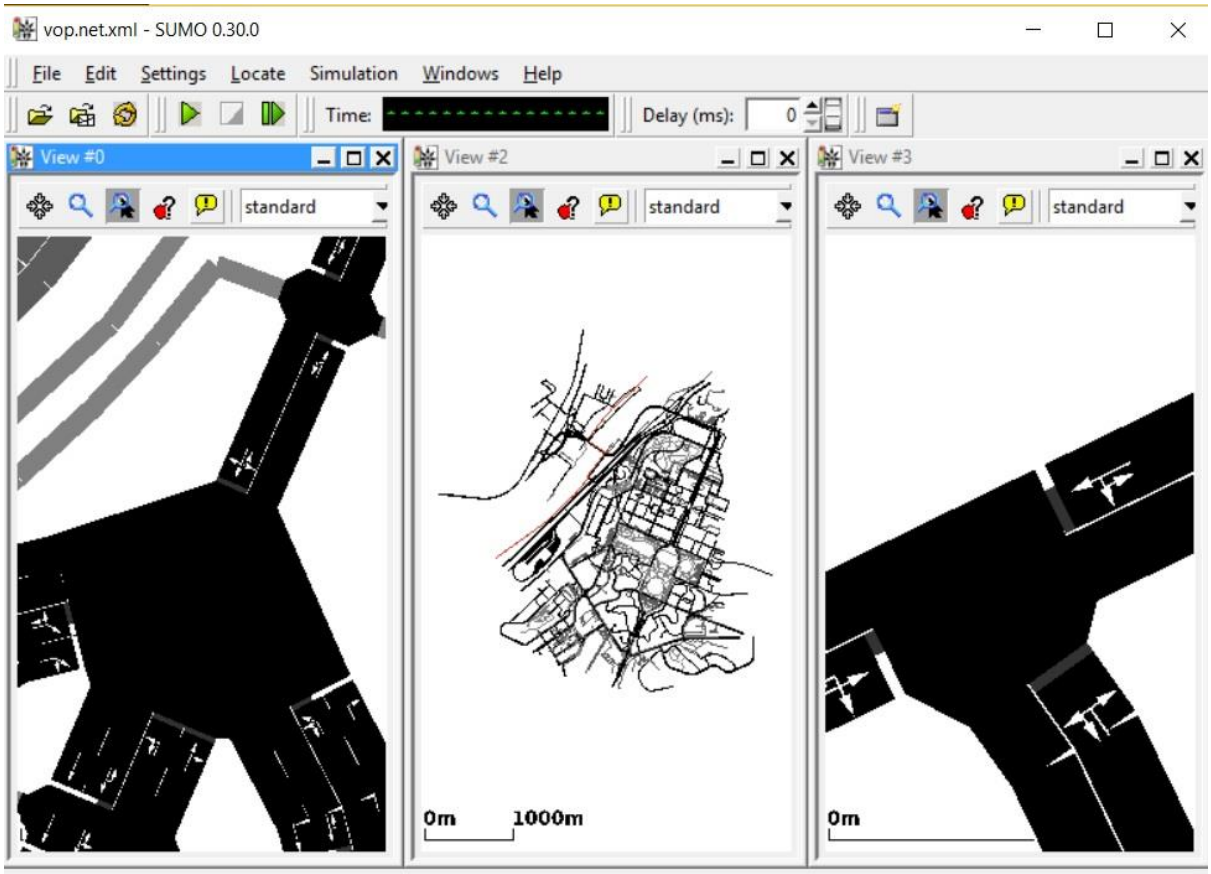


Figure 6:SUMO-GUI

2.2 Detectors in SUMO

There are three types of detectors in SUMO. They are induction loop detectors, lane area detectors and multi-entry-exit detectors. Detectors are responsible for the detecting the vehicles on the network.

2.2.1 Induction loop detectors

An additional file has to be created along with the network file and this file has to be specified in the input as an additional file. There are certain attributed that have to be described in the additional file such as id-name(id) of the detector, lane id is the name of the lane on which the induction loop detector has to be placed upon, position-this is the position of the induction loop detector on the specified lane at a distance from the beginning of the lane and can be described in metres, frequency is the time period in which the induction loop detector aggregates the values and file-the output file will be written in the format along with the name specified in the attribute “file”.

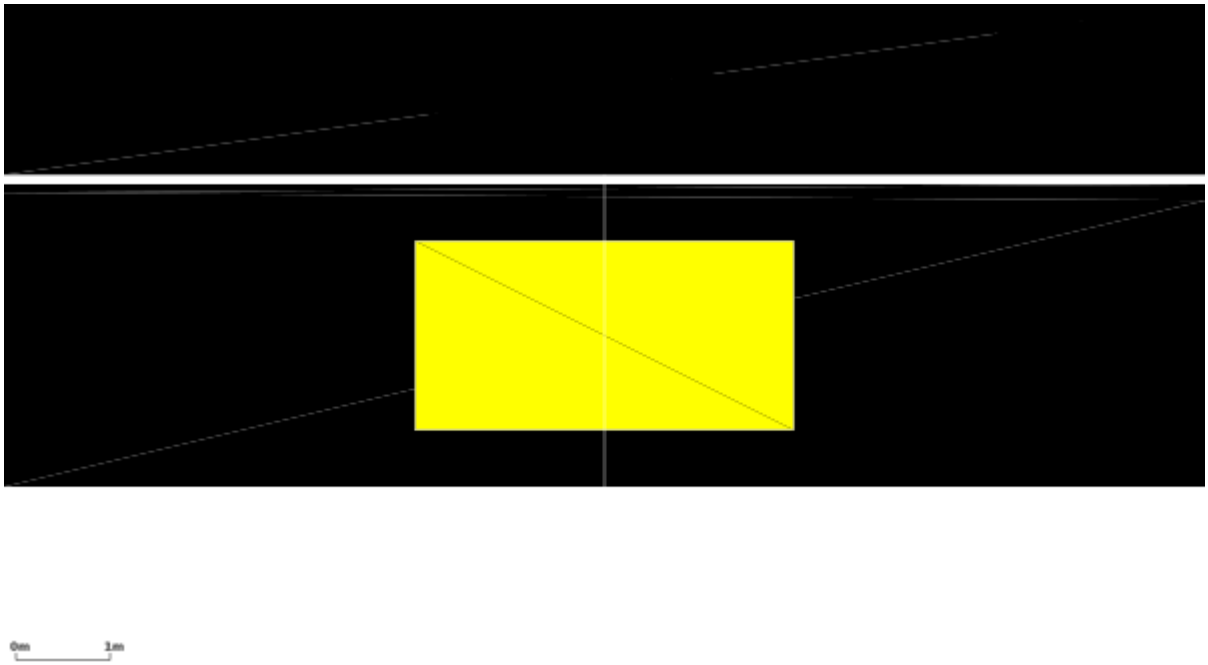


Figure 7: Visualization of induction loop detector

2.2.2 Lane area detector

The lane area detector can be used to capture the traffic on a lane or lanes. lane area detectors are similar to vehicle tracking cameras. Lane area detector has all the attributes as the induction loop detector in addition it has a length attribute which can be described by the attributes pos and endPos. Lane area detector can be coupled with a traffic signal to generate actuated traffic signal. Lane area detectors have to be specified in an addition file which can be loaded in the simulation.

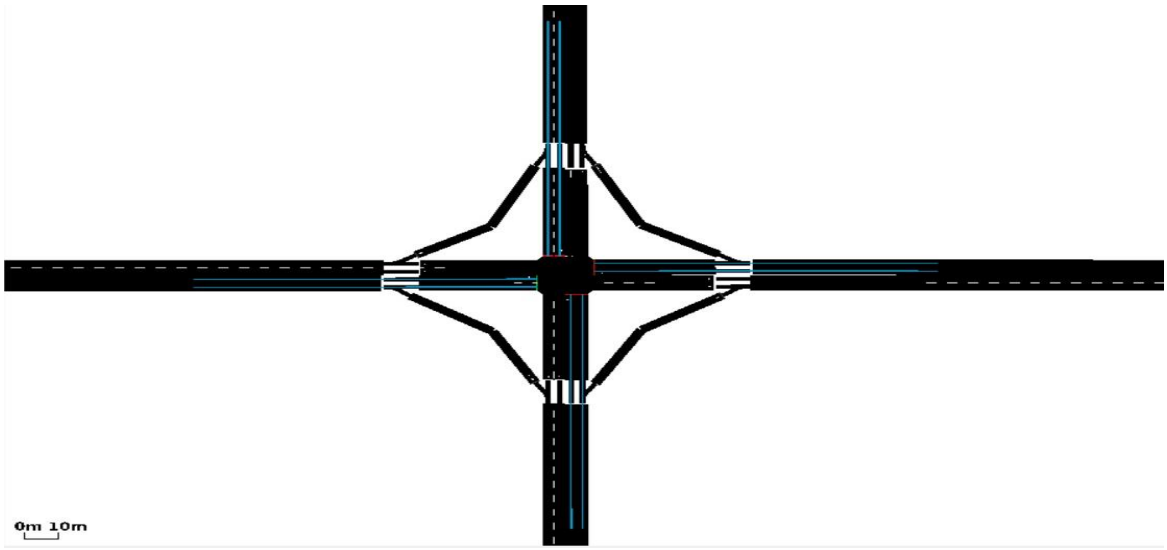


Figure 8: Intersection with Lane area detectors

2.2.3 Multi-Entry-Exit Detectors

These detectors can be described in an additional file and can be loaded on to the simulation.

```
<additionals>
<entryExitDetector id="ell" freq="100" file="out.xml"
   timeThreshold="10" speedThreshold="3">
      <detEntry lane="gneE11_0" pos="27"/>
      <detExit lane="gneE11_0" pos="30"/>
      </entryExitDetector>
</additionals>
```

The about image shows the xml script for the entry exit detectors. This script should be written into an xml file . This xml file has to be loaded in additional files in configuration file. The above additional creates an output as out.xml.

Figure 9: Visualization of multi-entry-exit detectors

2.3 Traffic signal Lights in SUMO

Traffic light is on of the most effective ways of managing the traffic in SUMO. There are three types of traffic lights in SUMO. They are static, actuated and delay based. The static traffic signal has a pre-timed signal stated that it follows. The actuated traffic signal is coupled with lane area detector. In SUMO traffic lights are normally generated by NETCONVERT command during the development of the networks. It is possible to change the duration of the traffic light according to the real traffic light programs by using netedit or by editing the logic of the traffic light in the xml file of the network simulator.

2.3.1 Static traffic signal

Figure 10: Various colour states displayed by the traffic light in SUMO

Character	GUI Color	Description
r		'red light' for a signal - vehicles must stop
y		'amber (yellow) light' for a signal - vehicles will start to decelerate if far away from the junction, otherwise they pass
g		'green light' for a signal, no priority - vehicles may pass the junction if no vehicle uses a higher priorised foe stream, otherwise they decelerate for letting it pass. They always decelerate on approach until they are within the configured visibility distance
G		'green light' for a signal, priority - vehicles may pass the junction
s		'green right-turn arrow' requires stopping - vehicles may pass the junction if no vehicle uses a higher priorised foe stream. They always stop before passing. This is only generated for junction type *traffic_light_right_on_red*.
u		'red+yellow light' for a signal, may be used to indicate upcoming green phase but vehicles may not drive yet (shown as orange in the gui)
o		'off - blinking' signal is switched off, blinking light indicates vehicles have to yield
O		'off - no signal' signal is switched off, vehicles have the right of way

Table 1:Table describing colour states of the traffic signal in SUMO

2.3.2 Based on Time gaps

This control scheme is a gap-based actuated traffic control. This traffic signal prolongs the traffic phase whenever a continuous stream of traffic is detected. The traffic signal changes to next phase after maximum duration of the phase or after detecting a sufficient time gap between successive vehicle. The quantity of time gap can be described in the parameters. This traffic control scheme couples with the lane area detector.

2.3.3 Delay-based traffic signal

This traffic signal is based on time loss by the vehicles waiting at the intersection. There are various parameters that must be specified for this typed of actuated signal. SUMO manual

suggests creating an additional file and specify that additional file containing the traffic light logic but we have found some errors by following this procedure, so it recommended to write the traffic light logic in the xml by finding the logic in the network xml file and replacing it with the delay based traffic signal parameters using the notepad++.

2.4 Open Street Map

OSM stands for open street map, OSM is a free and editable map of the world, released with an open-content license (OpenStreetMap Wiki, 2016a). According to Zilske et al. (2011), OSM data has proven to be a valuable data source for network creation. OSM offers in one source. Very high point coordinate accuracy, a strength of commercial data sources. OSM provides both geographic data and traffic signal information. OSM uses the elements node, way and relation, structured in an xml-format. A node is a single point in space defined by its latitude, longitude and id. A way connects two nodes, but ways are not only used for roads but all line elements present on a map. Relations group ways and nodes to logical entities such as buildings, rivers, forests, borders or public transit routes. OSM offers tags to define spatial public transit data such as routes and stops. OSM provides the means to define the actual path a vehicle takes. Instead of designing the network on the NETEDIT software. OSM data can be used to import real network on to SUMO. But, the OSM data has to be converted to xml format to be able to import into SUMO. Then Python can be used to assign flows on to this network.

2.5 Procedure to import OSM into SUMO

The working PC has to be installed with the following software such as SUMO Python Notepad++.In SUMO networks can be built using netedit software or they can be downloaded from the openstreetmap.org. we can manually select the area that we want to export from the OpenStreetMap .

The file that is exported from the OpenStreetMap is in the format of .osm. Microsoft Dos can be used to convert this file from .OSM format to .XML. In order to do that first the user has to run the DOS from there he has to navigate to the directory to where the SUMO software is installed. In this SUMO folder there is a folder named “bin”. In this bin folder you can find the start-command-line.bat.

The above command generates a file.rou.xml file. The next step is make a configuration file. In this file input files and time has to be specified. In order to create a configuration file. One can find a file named “test.sumo.cfg”. This file has to be copied into the map folder and then it has to edited using notepad++ with the following lines.

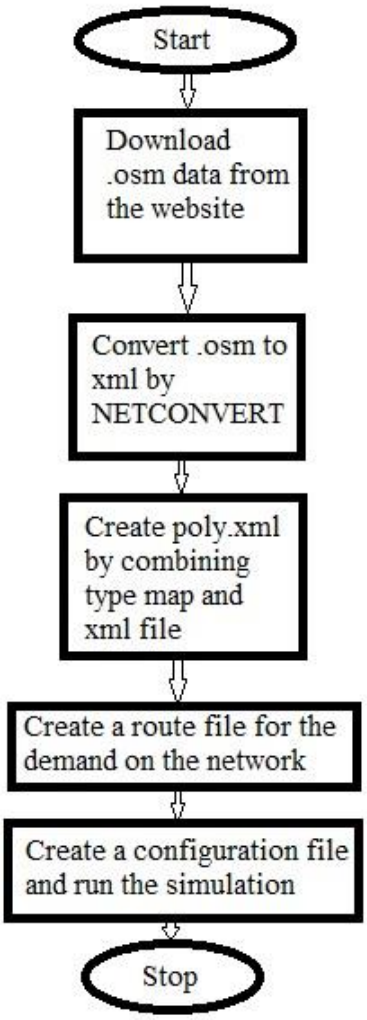


Figure 11: Using OSM file for SUMO simulation

All files required for the simulation have been prepared now. The simulation can be run by the following command SUMO-gui test.SUMO.cfg.

2.6 NETEDIT

NETEDIT is visual GUI software provided with SUMO. This software can be used to design networks from the scratch. This software can also be used to modify existing networks. Edges can be created or deleted. Two junctions which are nearby each other can be joined together. Restricted lanes can be created for a particular vclass. Many additional elements can be created in this software they are induction loop detectors, lane area detectors, entry exit detectors, bus stop, charging station, vaporizer, container stop, rerouter, calibrator and variable speed sign. The shape of the network can be changed as per the preferences. This tool saves the network file in .xml format. There are nine modes by which the NETEDIT tool can be used.

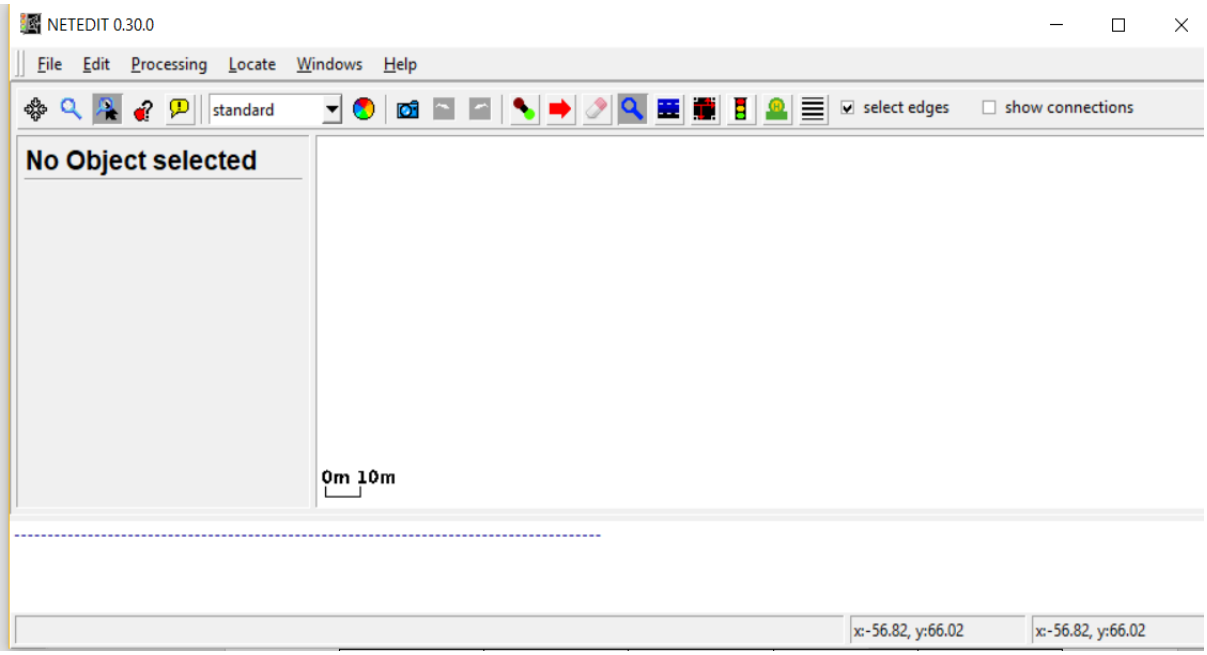


Figure 12:NETEDIT graphical user interface

2.7 Simulation Tests

In this case study the delay results on a single lane approach at a signalized intersection is determined. The length of the approach is 500m. The speed limit is 70 km/h.The traffic consists of passenger cars with a length of 4m. The signal has a cycle time of 90 seconds. The effective green is 50 seconds. SUMO implements the Messnne Teister algorithm for generating random numbers The length of each simulation run was 900 seconds which corresponds to the HCM study length. The purpose of this simulation is to study the variation of delay according to car following model characteristics. The SUMO generates a huge amount of output data including mean speed travel time duration waiting time max speed on a lane. The default lane capacity in SUMO is 2400veh/h/lane.

Traffic flow conditions

Traffic demand (veh/hr)	HCM v/c ratio	Uniform delay	Incremental delay	Average control delay
240	0.1	9	0.3	9.2
480	0.2	9.4	0.4	10.2
960	0.4	9.88	1.1	12..5
1440	0.6	10.58	2.3	15.7
1920	0.8	11.58	5.8	21.8
2400	1	13.67	23.1	45.5
2880	1.2	20	42.3	78.3

Table 2:Traffic flow conditions obtained by the HCM methodology.

Now the length of the artery approaching the intersection has been varied by 300m,500m 750m and 1000m and the results are shown in the table below

Average delay SUMO(500m)	Average delay SUMO(300m)	Average delay SUMO(1000m)	Average delay SUMO(750m)
9.2	9.1	9.2	9.2
10.3	9.9	10.9	10.5
11.3	10.4	12.1	11.6
13.5	12.4	15.5	14.5
18.7	16.8	19.4	18.8
38.2	34.2	39.5	38.9
75.4	73.1	78.1	76.8

Table 3: SImulation results

There are eleven types of car following models that are currently implemented in SUMO. Some car following models describe the behaviour of the driver in all situations whereas some car following models describe drivers behaviour when following other vehicle. Most car following models use several schemes to describe the followers behaviour.

Element Name	Description
carFollowing-Krauss	The Krauß-model with some modifications which is the default model used in SUMO
carFollowing-Kraussorig1	The original Krauß-model
carFollowing-PWagner2009	This model uses Todoslev's action points by Peter Wagner
carFollowing-BKerner	A Boris Kerner's model
carFollowing-IDM	Martin Treiber's intelligent driver model
carFollowing-IDMM	A variant of IDM
carFollowing-KraussPS	The default Krauß-model with road slope in consideration
carFollowing-KraussAB	Krauß-model with bounded acceleration
carFollowing-SmartSK	Variant of the Krauß-model
carFollowing-Wiedemann	A model by Wiedemann
carFollowing-Daniel	A model by Daniel Krajzewicz

Table 4: Car following models available in SUMO

First the Krauss car following model had been tested in SUMO.

The following car following models were studied in this study. They are described in the list below.

1.Krauss

2.IDM

4.Daniel

5.Kerner's

6.Krauss

7.Pwagner

8.Wiedemann

2.7.1 Krauss

Figure 13:Car following model test network

The above picture shows a one-ways direction road section with two lanes. A circular network has been chosen to regulate the density. The length of this circuit is 1000 metres. All the car following models were used to study the behaviour of the vehicles on this network. The first model is the Krauss car following model. The parameters used in this model are taken as suggested by the SUMO website.

Acceleration(m/s^2)	Deceleration(m/s^2)	Length(m)	Max speed(m/s)	Sigma
2.5	4.5	4	13	0.5

Table 5:Krauss car following model parameters

The simulation was run for a duration of 60 minutes and the results are show in the fundamental diagram of Flow-Density, Speed-Flow and Speed-Density. All of the graphs displayed below show the relationship between the basic traffic parameters as expected.

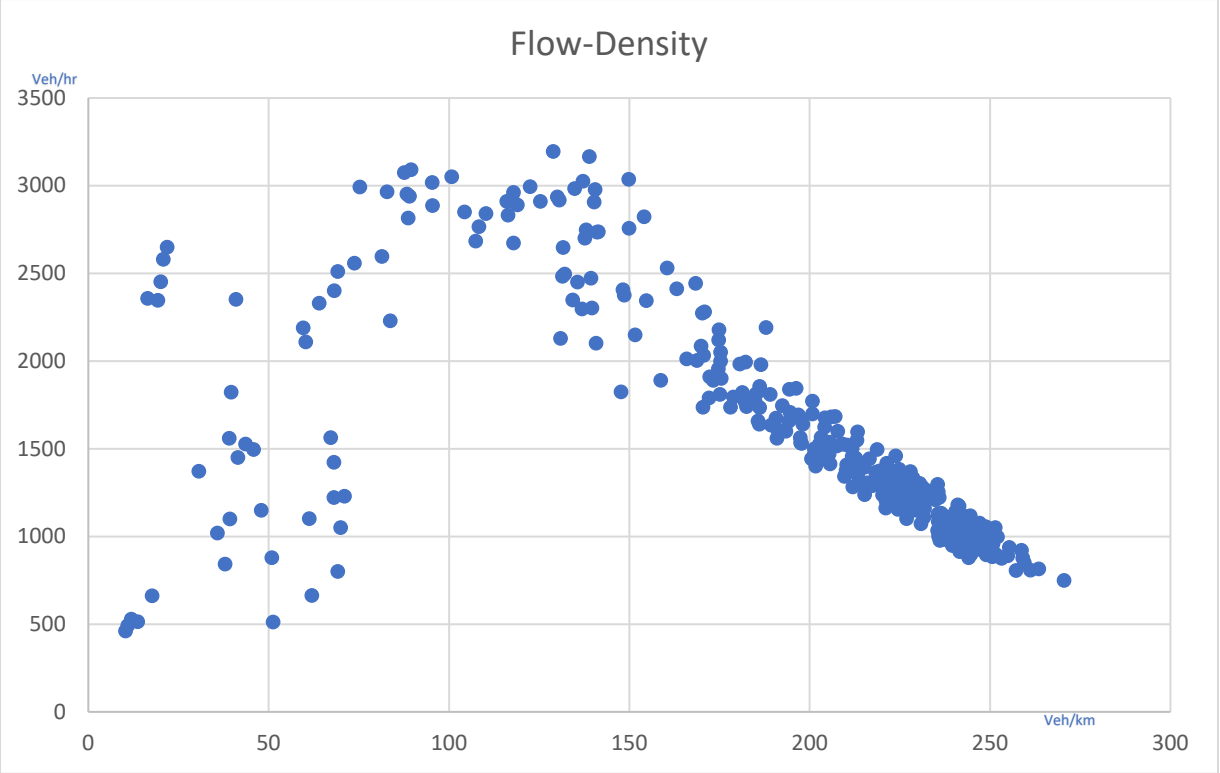


Figure 14Flow density fundamental diagram of Krauss car following model.

The highest flow observed from the graph is 3195 Veh/hr where the density is 128 Veh/km. Hence the optimal density by Krauss following model is 128 Veh/km. The jam density is 260 Veh/km

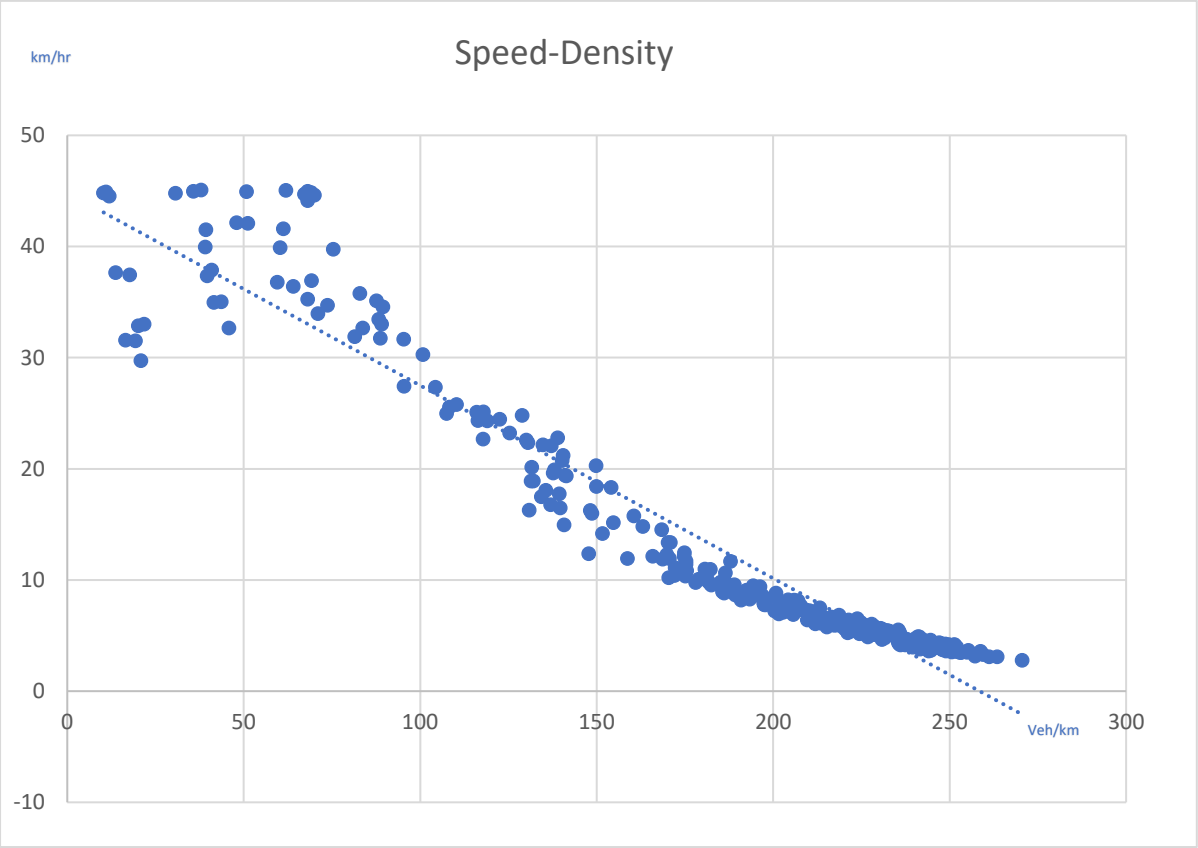


Figure 15:Speed density fundamental diagram of Krauss

The traffic flows freely from 30 to 50 km/hr after which the bound flow ranges from 10 to 30 km/hr. The congestion starts when the traffic density exceed 180 Veh/km which leads to a traffic speed of less than 10Km/hr.

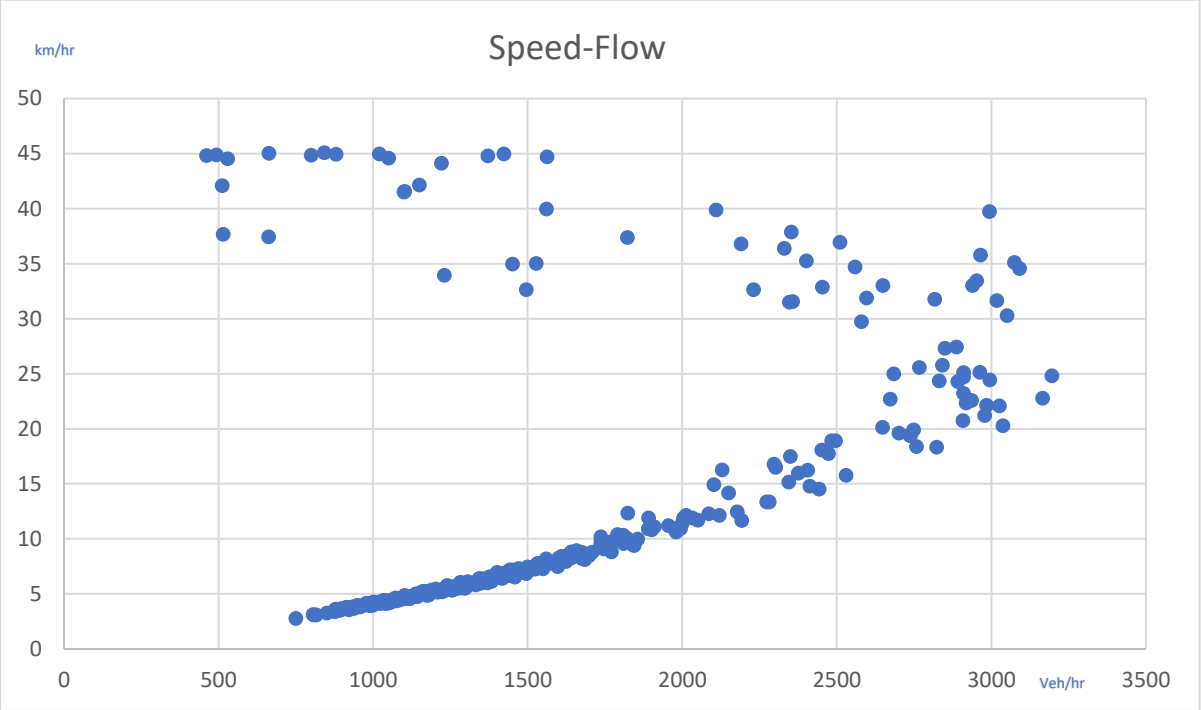


Figure 16: Speed density fundamental diagram of Krauss

One can observe from the fundamental diagram of Krauss car following model the maximum flow which is around 3195 Veh/km and the critical velocity is 24 Km/hr.

Then the Krauss model has been computed outside SUMO in a mathematical software. In this scenario a vehicle on a single lane which with an initial speed of 20m/s comes to rest and then accelerates to 20m/s. The behavior of the following vehicles is calculated using the Krauss car following model mathematical formula. Their results are shown in the Space-Time, Speed

Time and Acceleration time graph below.

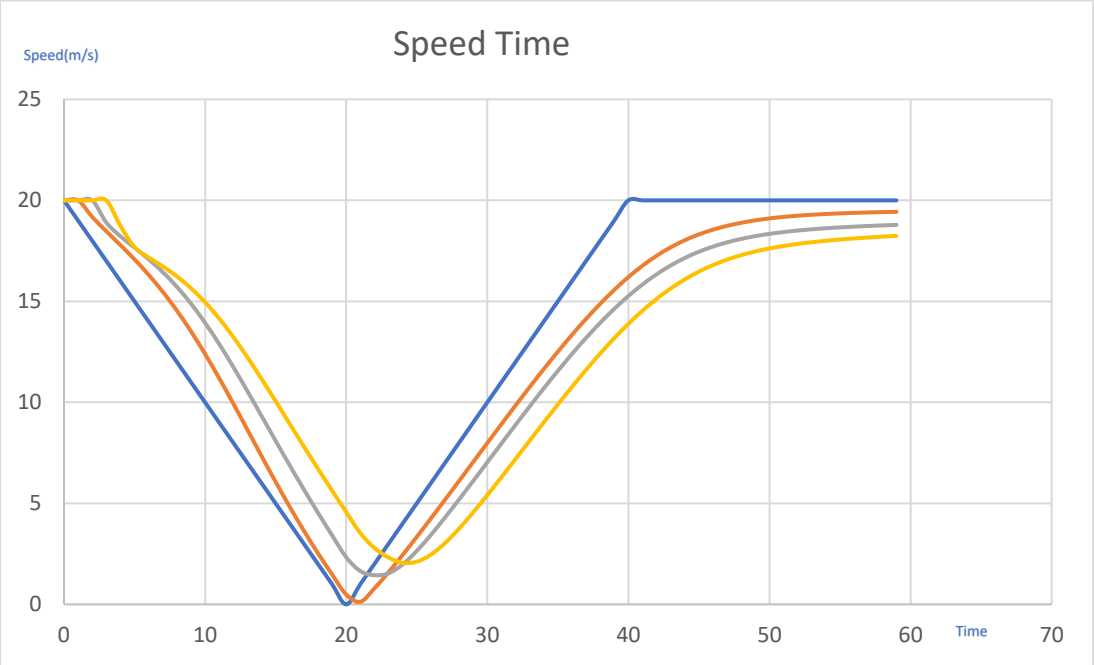


Table 6:Speed time graph of Krauss car following model

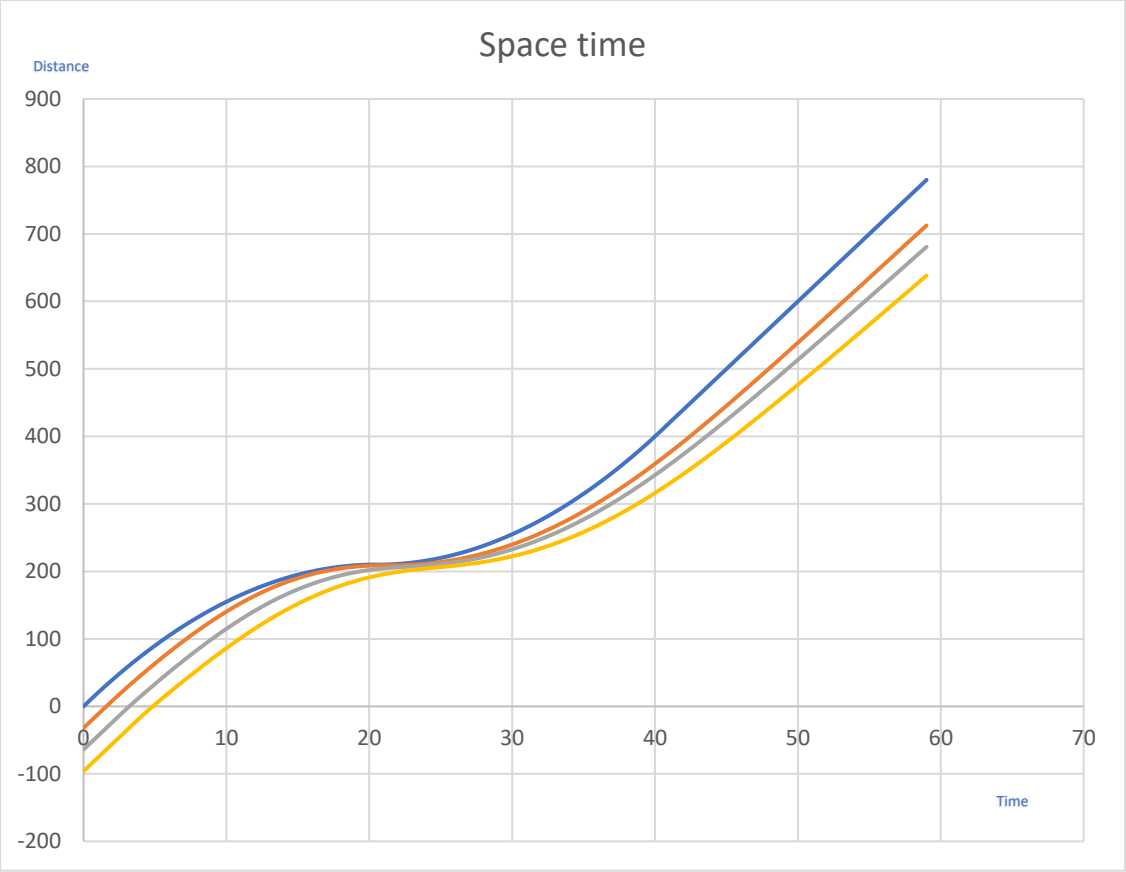


Table 7:Space time graph of krauss car following model

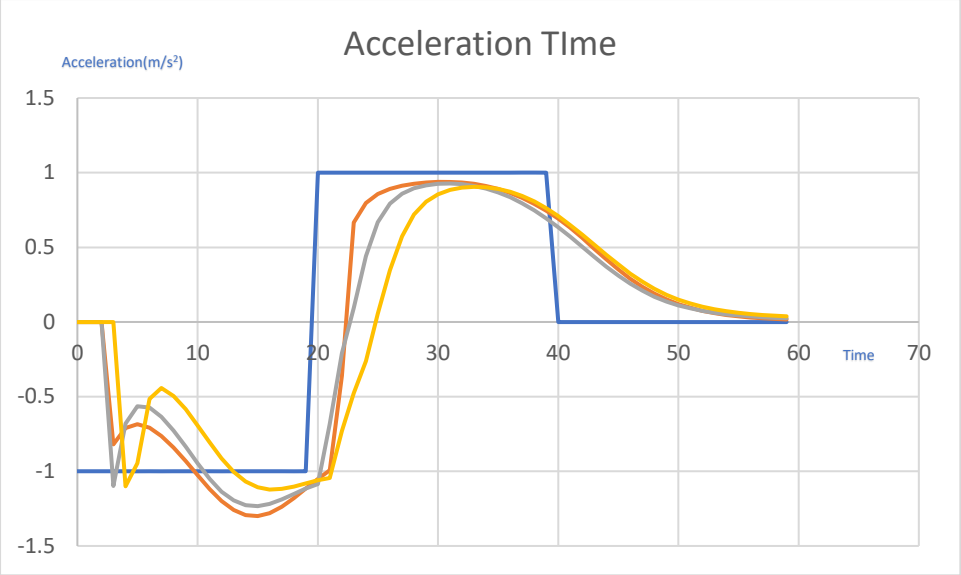


Figure 17;Krauss car following model results from

During the simulation in SUMO one can visualize the network parameters. These network parameters are shown in the image displayed below.

network: Parameter

Name	Value	Dynamic
loaded vehicles [#]	264	✓
insertion-backlogged vehicles [#]	108	✓
departed vehicles [#]	156	✓
running vehicles [#]	98	✓
arrived vehicles [#]	58	✓
collisions [#]	0	✓
teleports [#]	0	✓
end time [s]	3600	✗
begin time [s]	0	✗
step duration [ms]	55	✓
simulation duration [ms]	23	✓
idle duration [ms]	32	✓
duration factor []	43.48	✓
ups [#]	4260.87	✓
mean ups [#]	6299.38	✓
nodes [#]	10	✗
edges [#]	10	✗
total edge length [km]	0.96	✗
total lane length [km]	1.92	✗
network version	0.27	✗

Figure 18:Network parameters

2.7.2 IDM

IDM stands for intelligent driving model. This is a type of car following model which is time continuous used to simulate the traffic on freeway and urban traffic. The parameter specific to this model is Delta which is the acceleration exponent. The simulation was run for a duration of 60 minutes. The following parameters are used to simulate the traffic on the network shown in thee figure above.

IDM	
Acceleration(m/s^2)	0.73
Deceleration(m/s^2)	1.67
Length(m)	4
Max Speed(m/s)	13
Tau(Seconds)	1.5
Delta	4

Table 8:IDM car following model parameters

The results are show in the fundamental diagram below

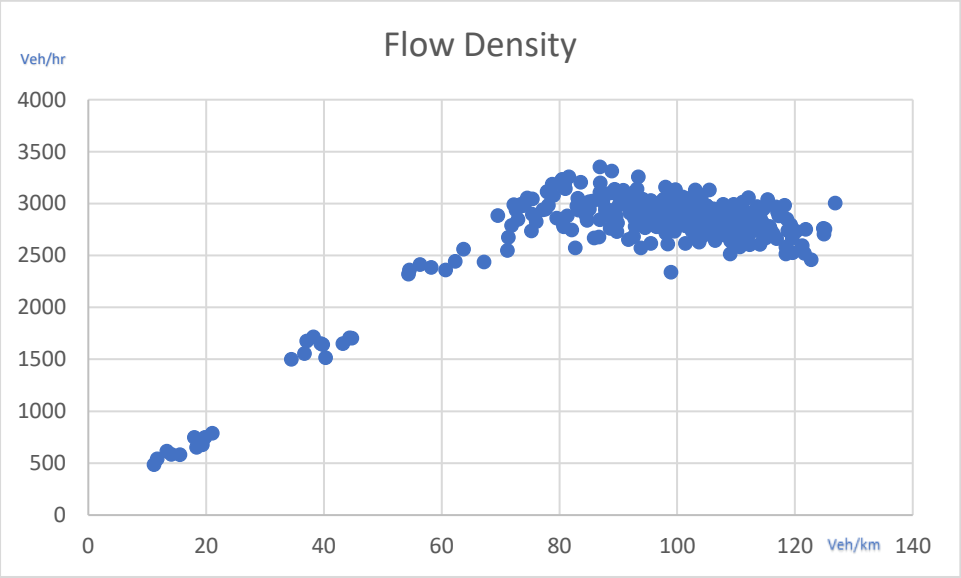


Figure 19 Flow Density fundemental diagram of IDM

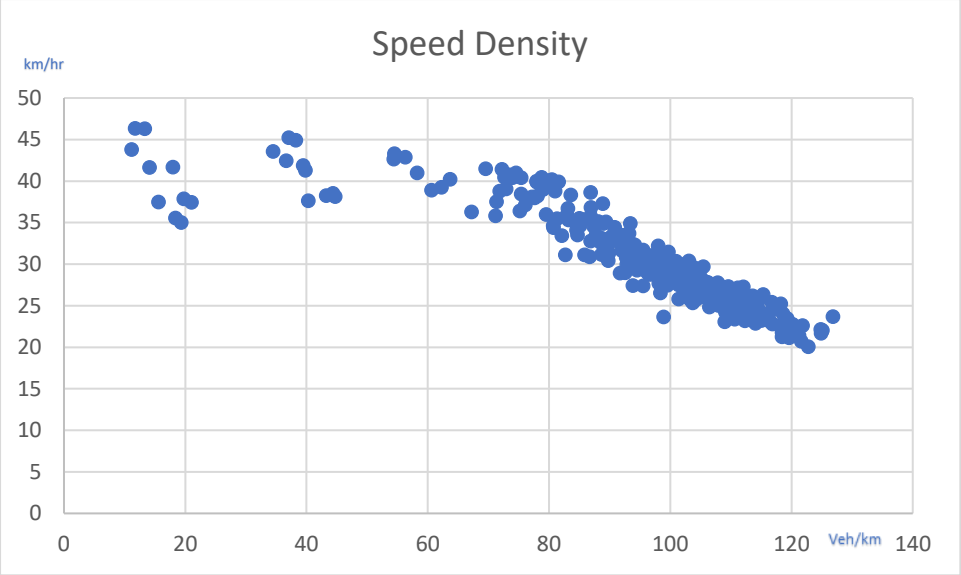


Figure 20Speed Density fundamental diagram of IDM

The congestion for IDM car following model starts at 100veh/km.

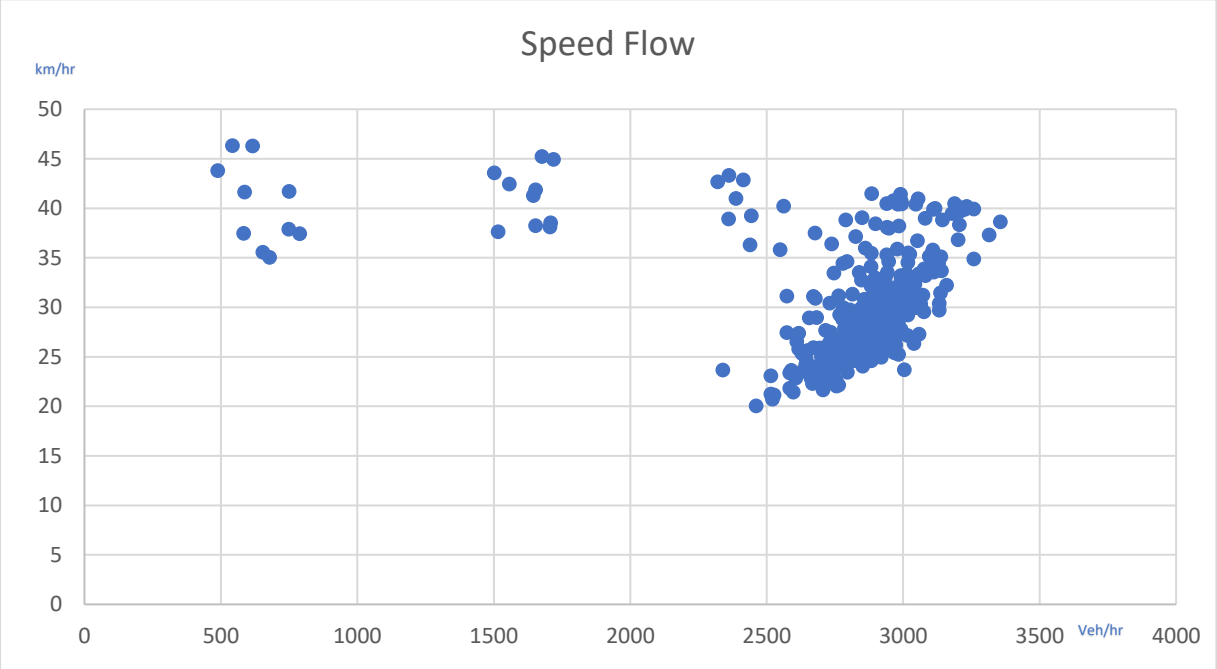


Figure 6:IDM car following model simulation results

It can be observed from the above speed flow fundamental diagram of IDM car following model that the critical velocity is around 38Km/hr.

2.7.3 Krauss Vs IDM

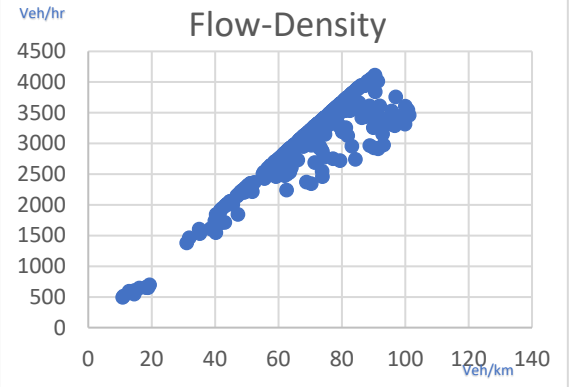
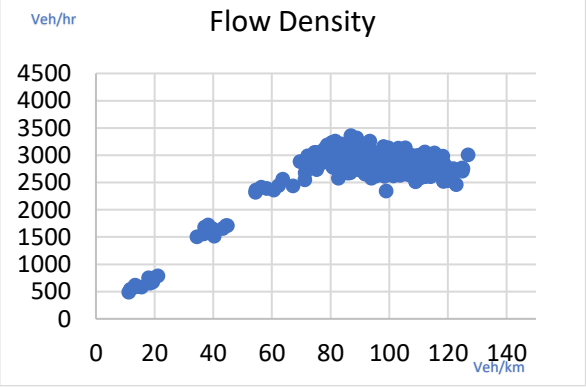
Next the simulation tests were run considering the same parameters for both the Krauss and IDM. The parameters are described in the table below

Krauss	IDM

Acceleration =1m/s^2	Acceleration=1m/s^2
Deceleration=2m/s^2	Deceleration=2m/s^2
Max Speed=13m/s	Max Speed=13m/s
Sigma=0.5	Length=4m
Length=4m	Tau=1.5s
	Delta=4

Table 9:IDM and Krauss car following model with same parameters

The results of this test are show in the fundamental diagrams below

KRAUSS	IDM
	

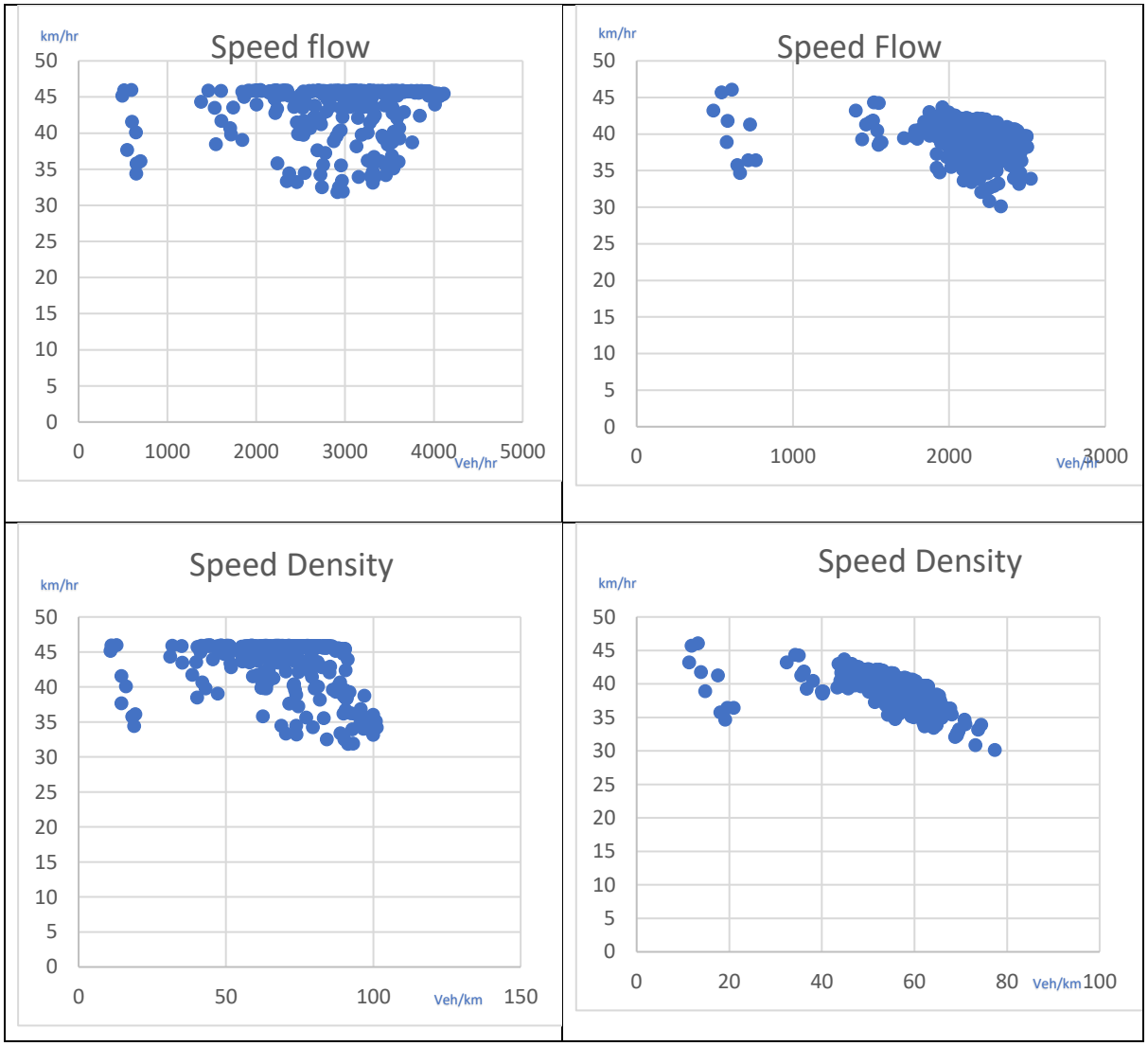


Figure 21 Comparison between Krauss and IDM

Both the Krauss and IDM car following models performed poorly when their parameters were changed. Krauss car following model has a maximum traffic flow of 4109 Veh/hr where as IDM car following model has a maximum traffic flow of 3315 Veh/hr. When the parameters of both these car following models altered they di not show the characteristics of a typical car following model. One can observe this issue from the velocity flow fundamental diagrams.

2.7.4 Daniel

The following parameters were used in testing the Daniel car following model.

Name	Value
Acceleration	1.5 m/s^2
Deccelaration	3.5 m/s^2
Mingap	3 m/s^2
Emergency deceleration	9.5 m/s^2
Max speed	13 m/s
Tau	1 sec

Table 10:Daniel car following model parameters

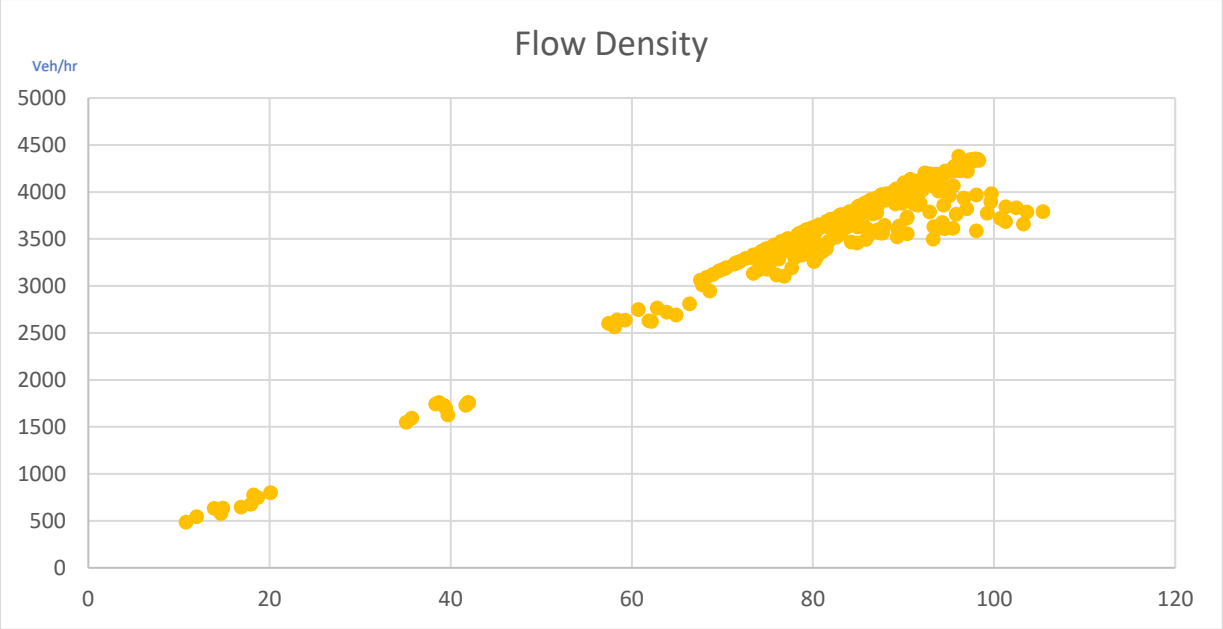


Figure 22: flow density fundamental diagram of Daniel car following model

The graph provides the relationship between the flow and density of a road section using the Daniel's car following model. As is observed from the graph

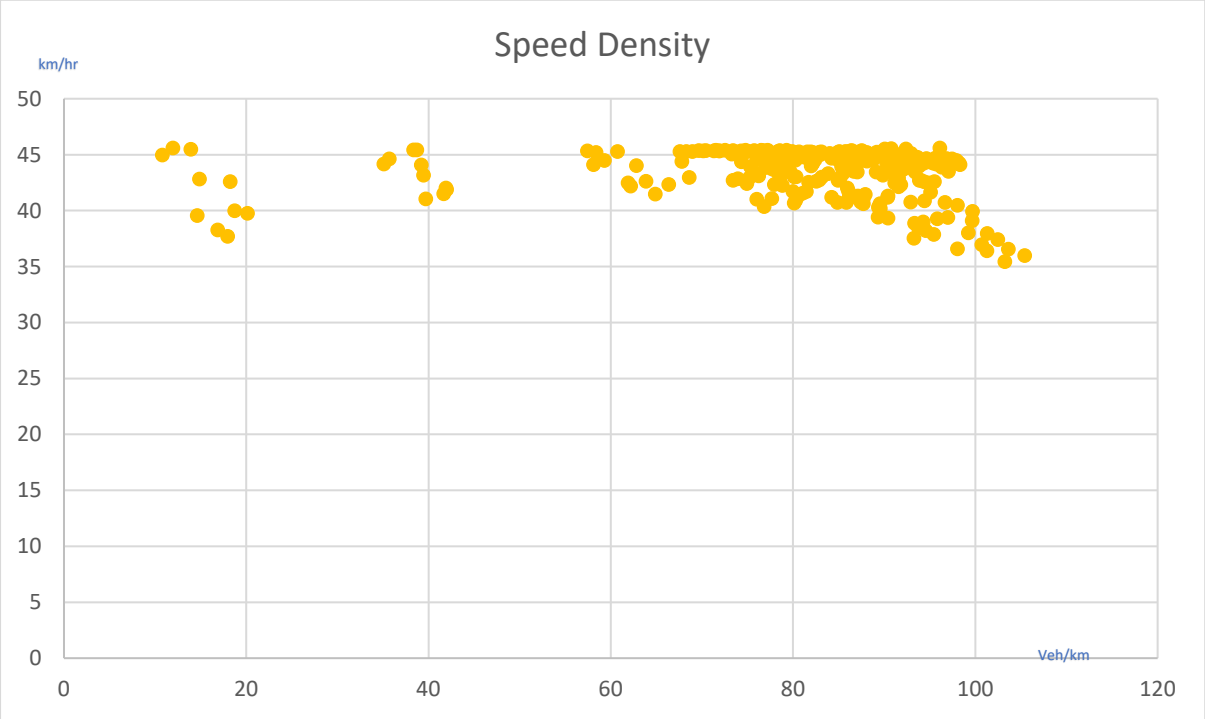


Figure 23:Speed density fundamental

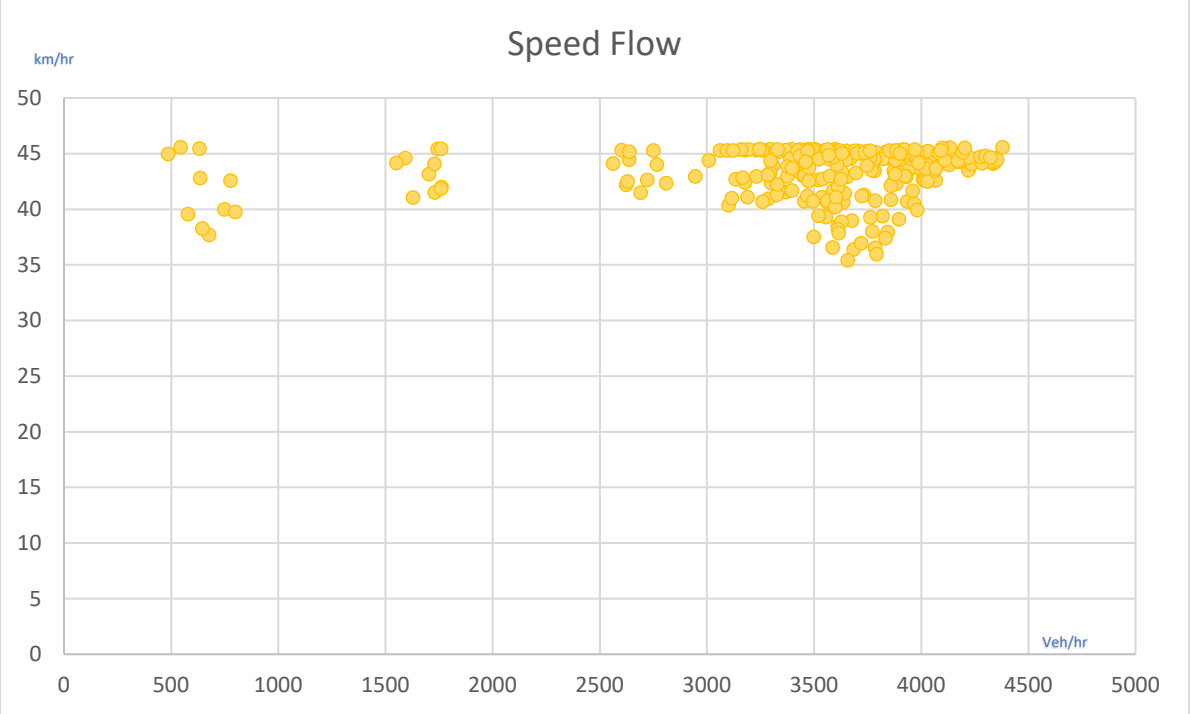


Figure 24: Speed flow fundamental diagram of Daniel Car following model

2.7.5 Kerner's

The following parameters were used in testing the Kerner's car following model.

Name	Value
Acceleration	1.5 m/s^2
Deceleration	3.5 m/s^2
Mingap	3 m/s^2
Emergency deceleration	9.5 m/s^2
Max speed	13 m/s
Tau	1 sec

Table 11: Kerner's car following model parameters

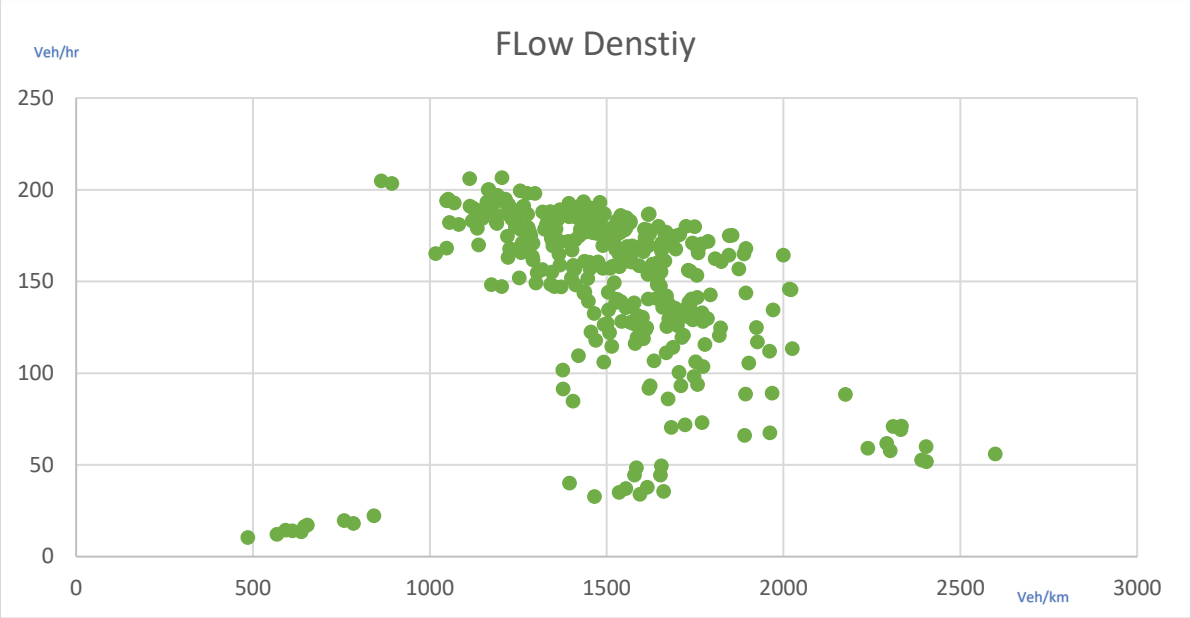


Figure 25:Flow-Density fundamental diagram of Kerner car following model.

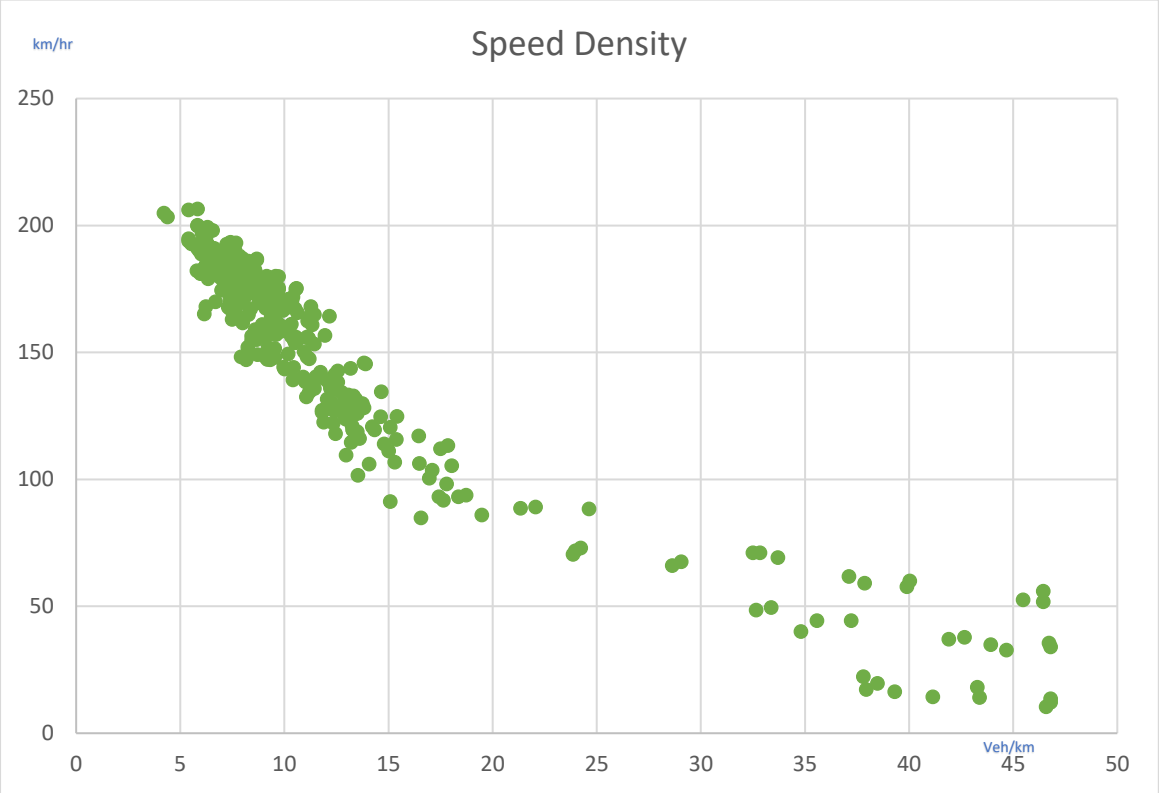


Figure 26:Speed density fundamental diagram of Kerner's car following model

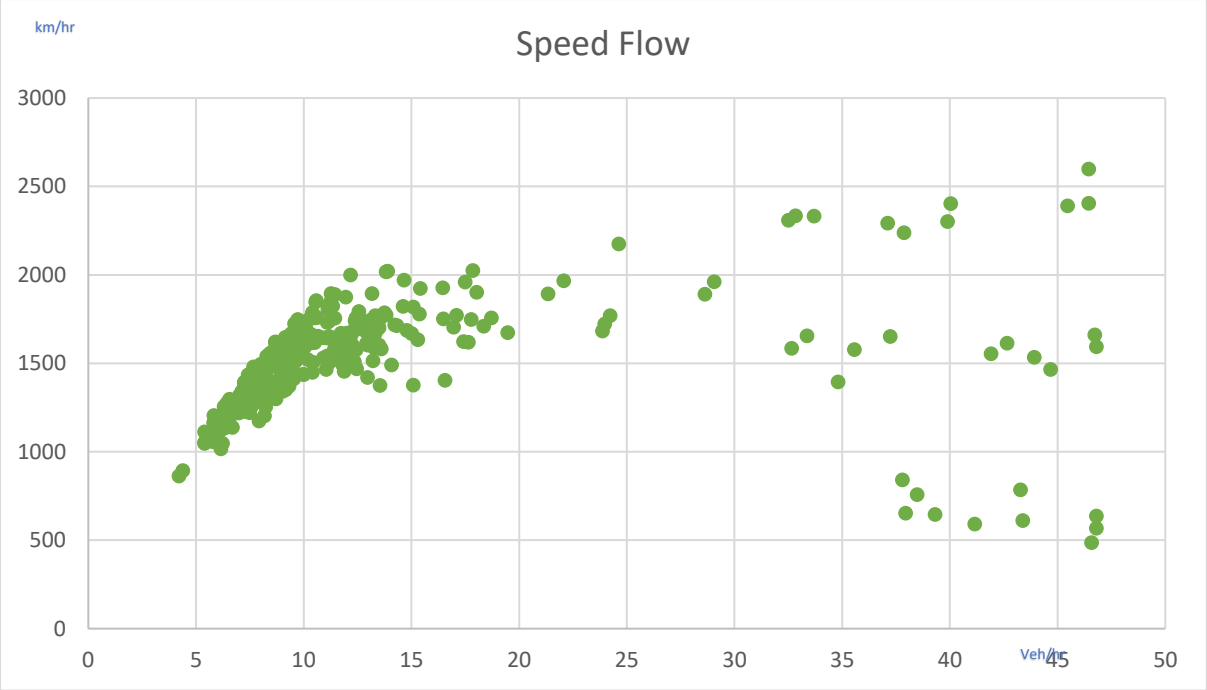


Figure 27:Speed flow fundamental diagram of Kerner's car following model

2.7.6 Pwagner

The following parameters were used in testing the Peter Wagner car following model.

Name	Value
Acceleration	1.5 m/s^2
Deceleration	3.5 m/s^2
Mingap	3 m/s^2
Emergency deceleration	9.5 m/s^2
Max speed	13 m/s
Tau	1 sec

Table 12: Peter Wagner's car following model parameters

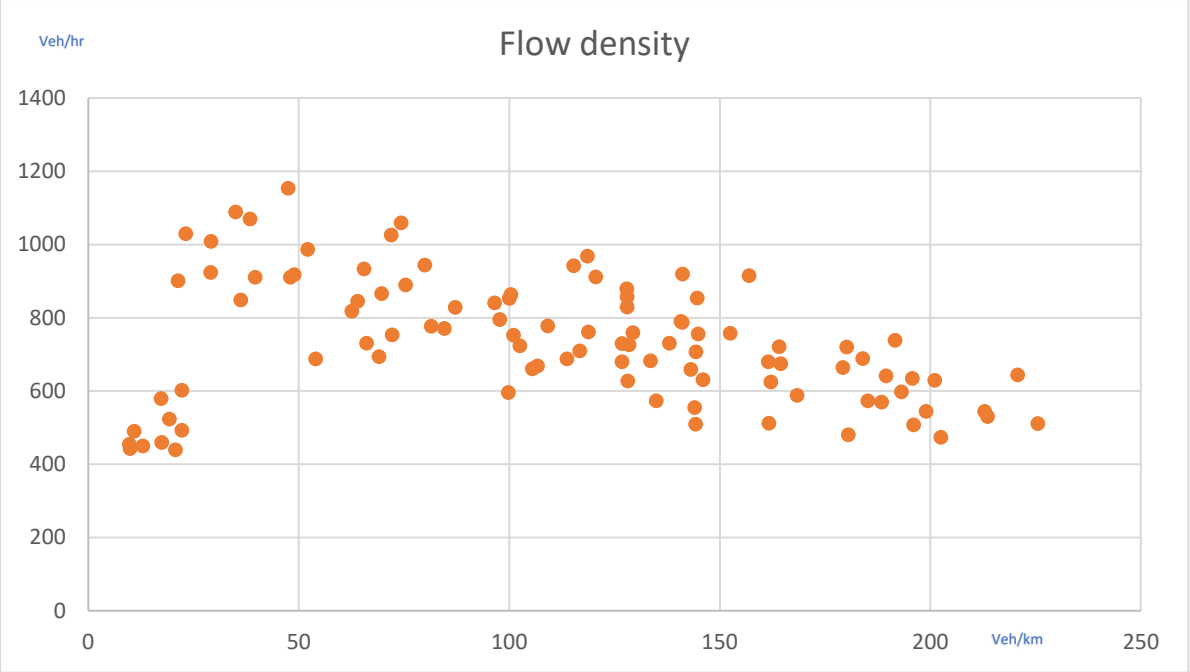


Figure 28:Flow density fundamental diagram of PWagner's Car following model

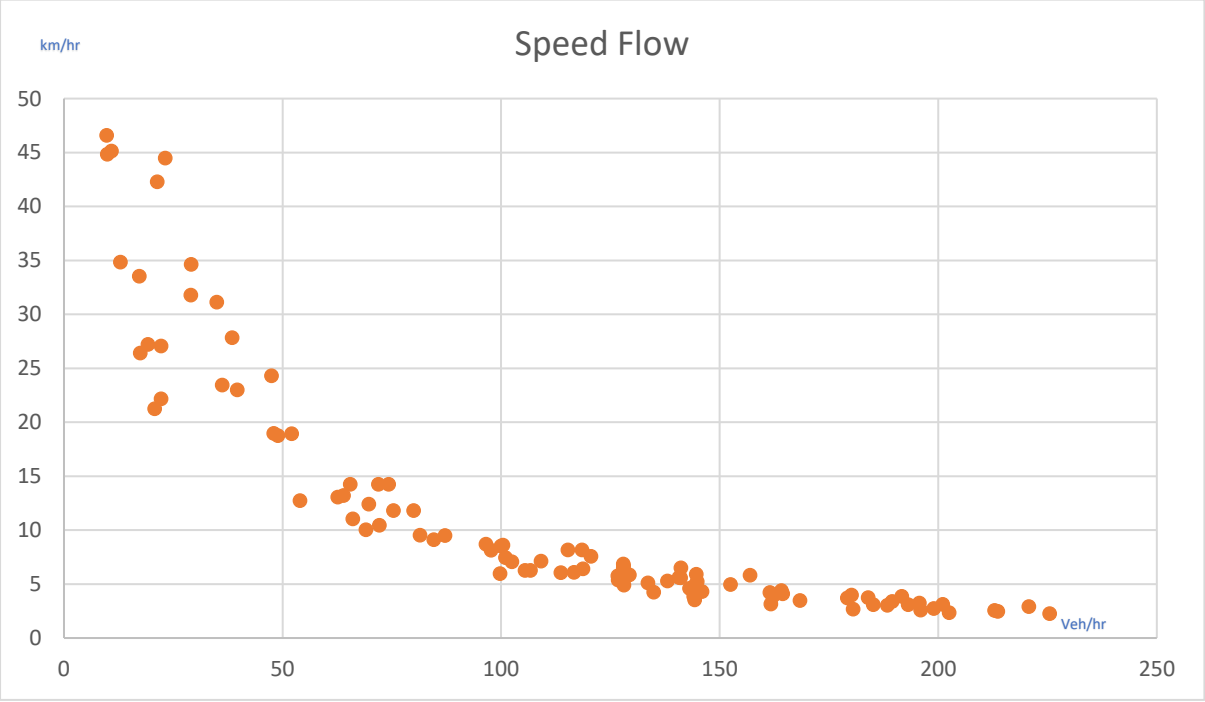


Figure 29 Pwagner car following model

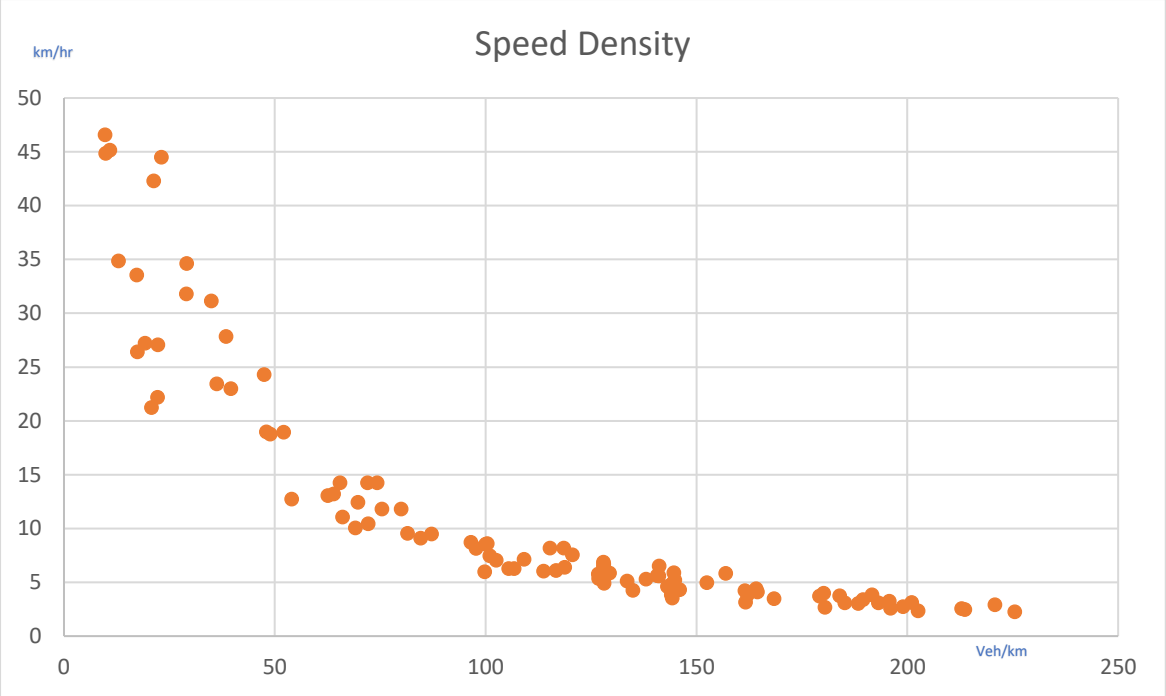


Figure 30 Pwagner car following model

2.7.7 Wiedemann

The following parameters were used in testing Wiedemann car following model.

Name	Value
Acceleration	1.5 m/s^2
Deceleration	3.5 m/s^2
Mingap	3 m/s^2
Emergency deceleration	9.5 m/s^2
Max speed	13 m/s
Tau	1 sec

Table 13:Weidemann car following model parameters

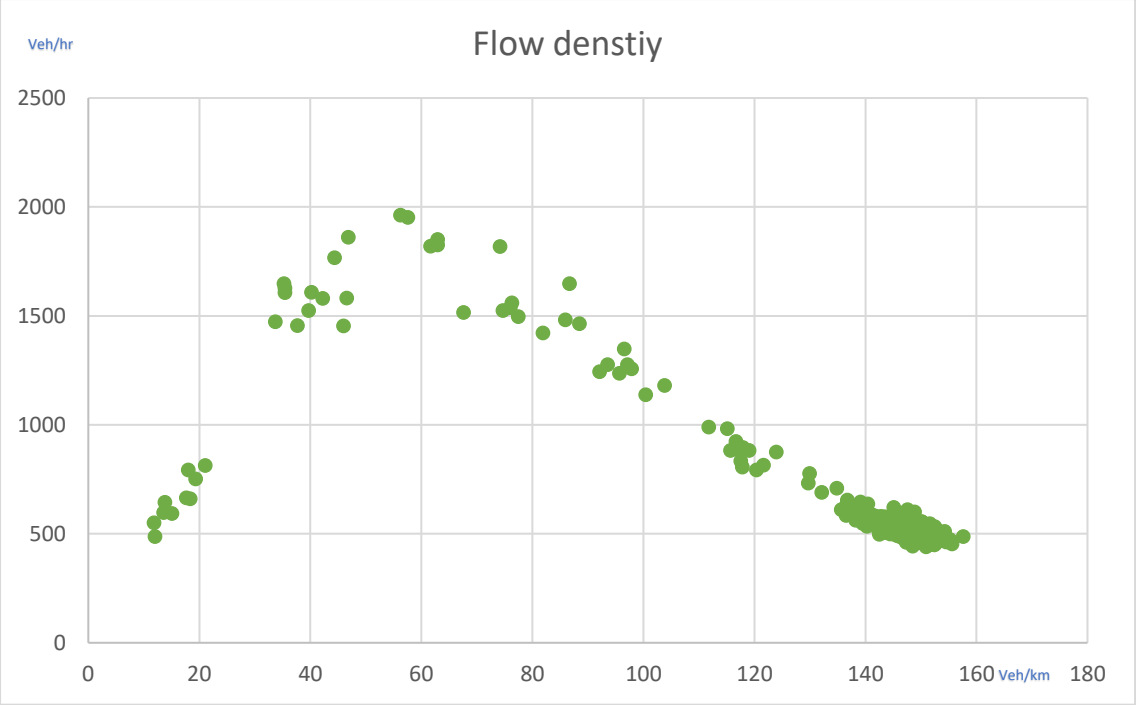


Figure 31: Wiedemann's car following model

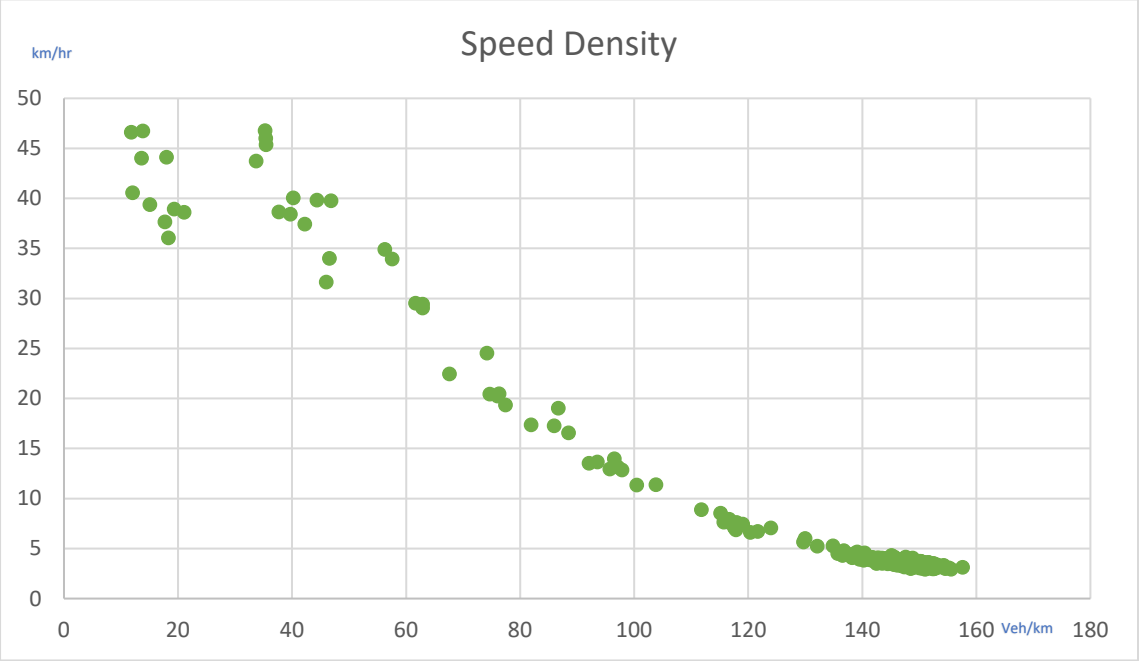


Figure 32:Speed-Density fundamental diagram of Wiedemann's car following model

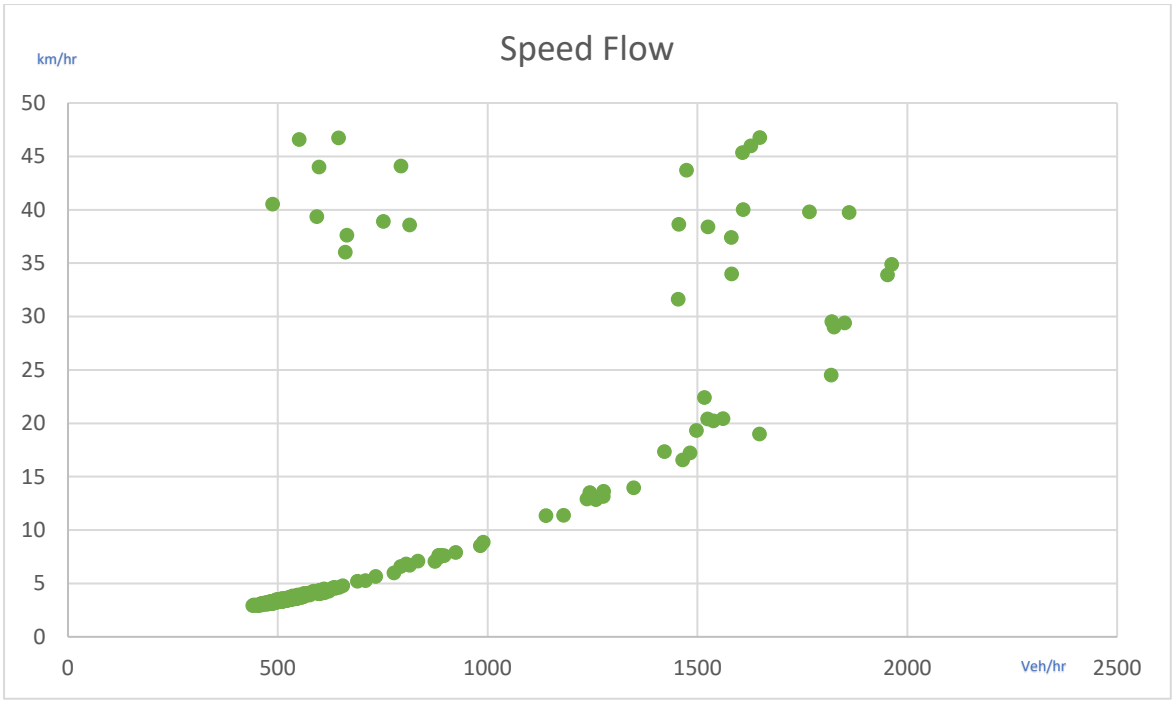
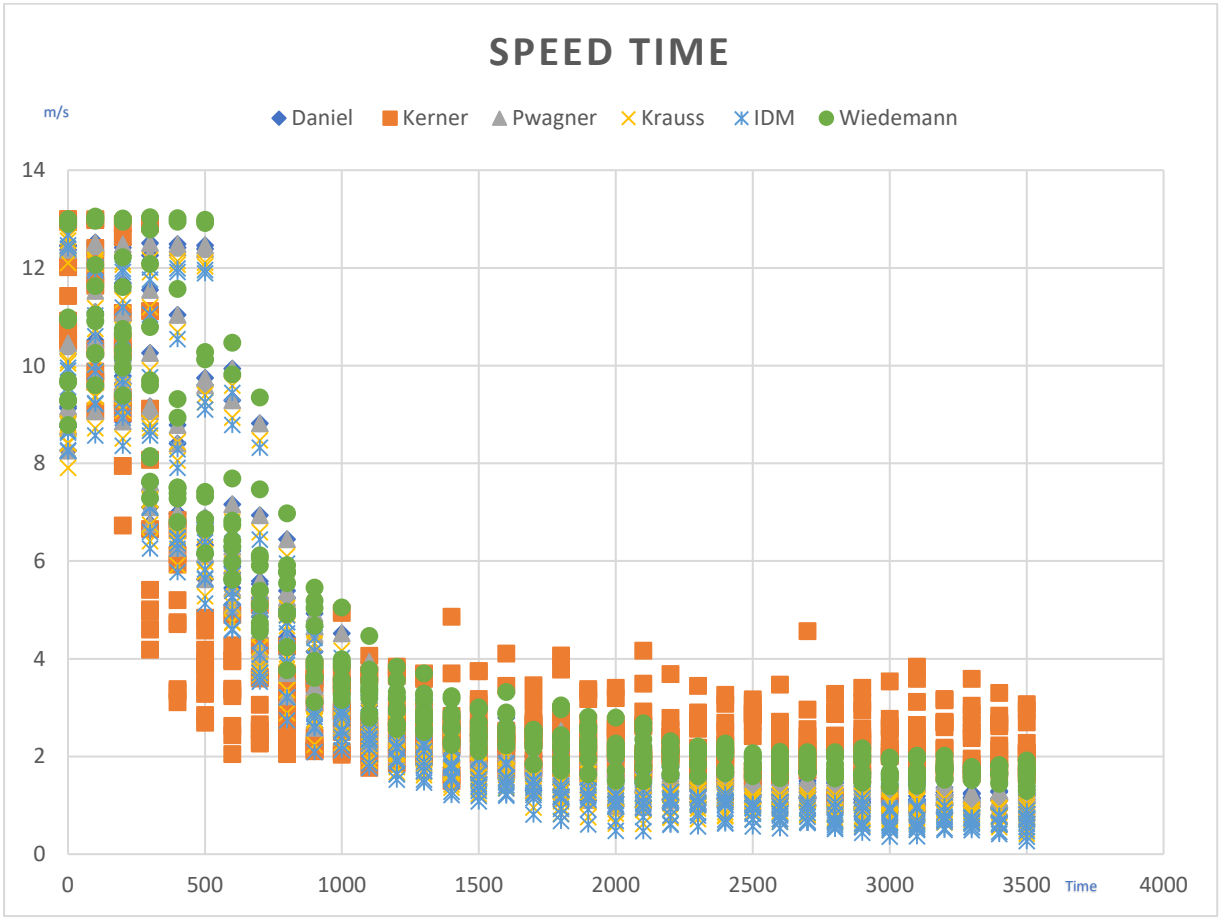


Figure 33:Speed Flow fundamental diagram of Weidemann car following model



2.8 Lane Changing Model

Traffic flows are influenced by lane changing model in addition to the car following model. There are three lane changing models that are available in SUMO. They are tested on a circular circuit of 1km length and consists of 3 lanes. There are three inductions loop detectors on each lane.

1. LC2013
2. SL2015
3. DK2008

Two models have been tested in this study They are LC2013 and SL2015.

2.8.1 LC2013

The flowing parameter were used in testing the lane change model

lcStrategic: The driver enthusiasm to change the lane. This vale ranges from 0 to infinite. Higher values will influence the driver change the lane early.

lcCooperative: The readiness to execute cooperative lane changing. Higher value leads to the driver being more cooperative to other drivers. The value ranges from 0 to 1. Default value is 1.

LcSpeedGain: The driver's eagerness for executing lane change in order to gain speed. Higher value results in more lane changing. The value of this parameter changes from 0 to 1.

lcKeepRight: The willingness for following the rule keep right. The value ranges from 0 to infinite.The higher the values the earlier the lane change.

lcLookaheadLeft: This is driver's ability to recognize the vehicles in a certain distance before changing the lane to the left. The value ranges from 0 to infinite and the default value is 2.

lcSpeedGainRight: Factor for configuring the threshold asymmetry when changing to the left or to the right for speed gain. Naturally the choice for changing to the right takes more consideration. Symmetry is achieved when set to 1.0. *default: 0.1, range]0-inf[*

All the above parameters were used in the model SL2015 too.

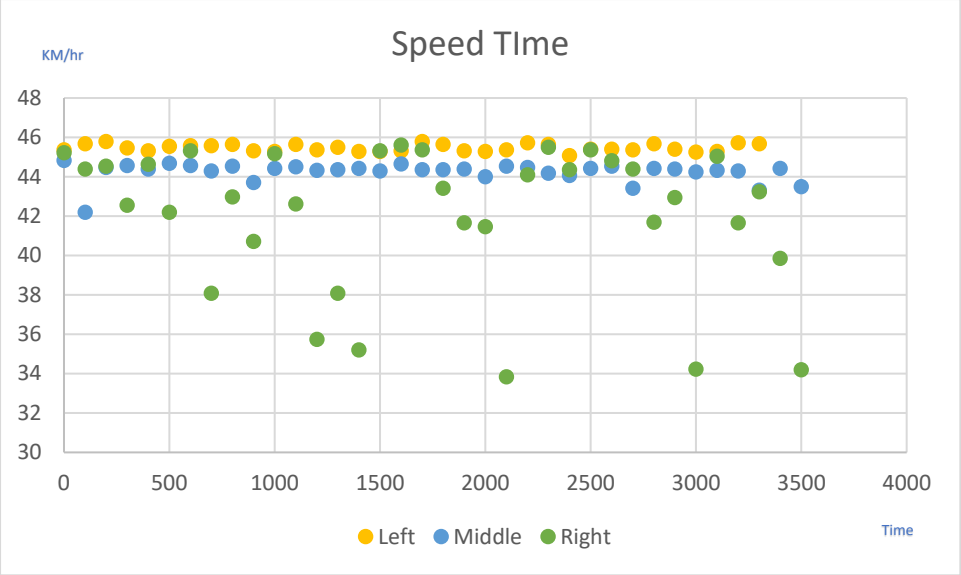


Figure 34: Speed Time diagram of LC2013 model

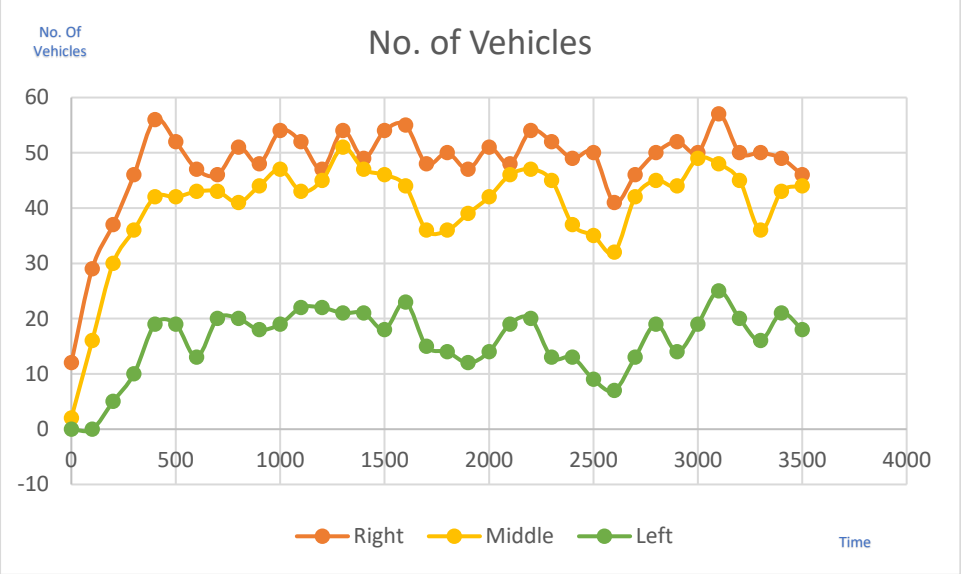


Figure 35: Throughput of LC2013 model

One can observe from the graph that the number of vehicles on the right lane are always high during the simulation. The number vehicles on the middle lane came close to the number of vehicles on the right lane at the 3000 sec instance. The number of vehicles on the left lane is always low as it the lane for the vehicles with high velocity.

2.8.2 SL2015

lcSublane: The enthusiasm for utilizing the Configure lateral alignment inside the path. Higher values will lead in drivers sacrificing speed for lateral alignment.

lcPushy : Ability to cut in laterally on other drivers. The value ranges from to 1.

lcPushyGap: The minimum gap desired from the other drivers while impinging laterally on other drives.It ranges from 0 to minGapLat.

lcAssertive: The driver's willingness to accept front and rear gaps on the target lane. It's default value is 1. All positive real numbers can be used in this parameter.

lcImpatience: Time changing fator for altering IcAssertive and IcPushy. It ranges from to -1 to 1. Default value is zero.

lcTimeToImpatience: Impatience grows whenever a manoeuvre to change lane is blocked. In this parameter the time to reach maximum impatience can be described. Default value is infinite(disables the growth of impatience).

lcAccelLat: Maximum lateral acceleration per second.

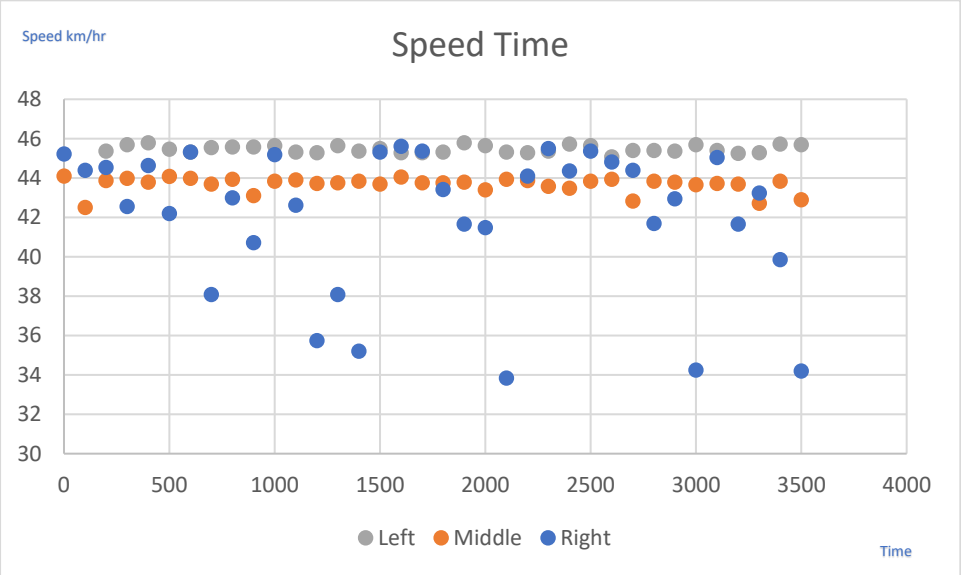


Figure 36:Speed Time diagram of SL2015 model

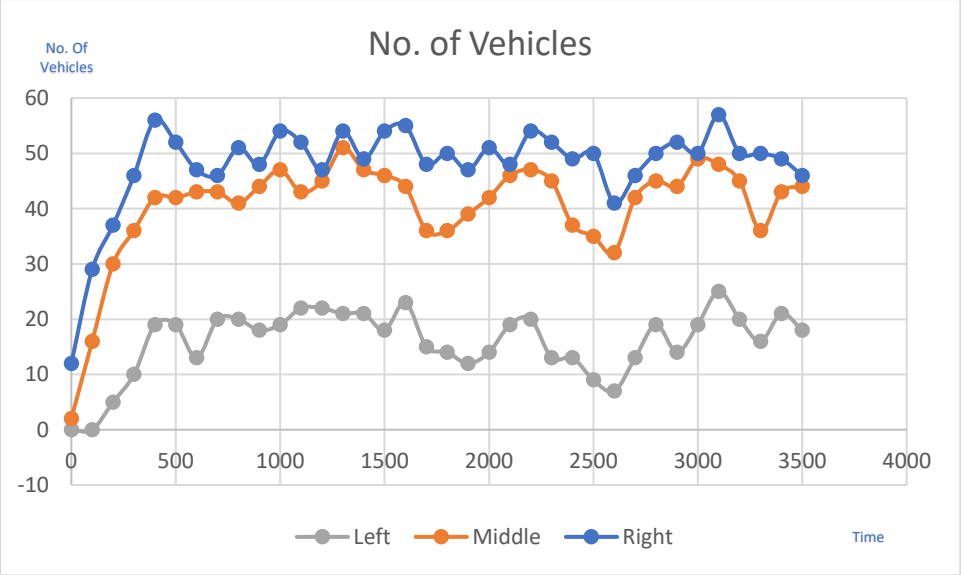


Figure 37: Throughtput of SL2015

3.Case Study

The area selected for our study is that of a major road with a ring function that almost surrounds the west part of the district core. Viale Egeo allows you to travel outside the nucleus of the district without entering the same. This allows to regulate the flow of vehicles in the study area, however the level of congestion is high due to the presence of many offices and presence of underground stations and associated multi-level parking lots.

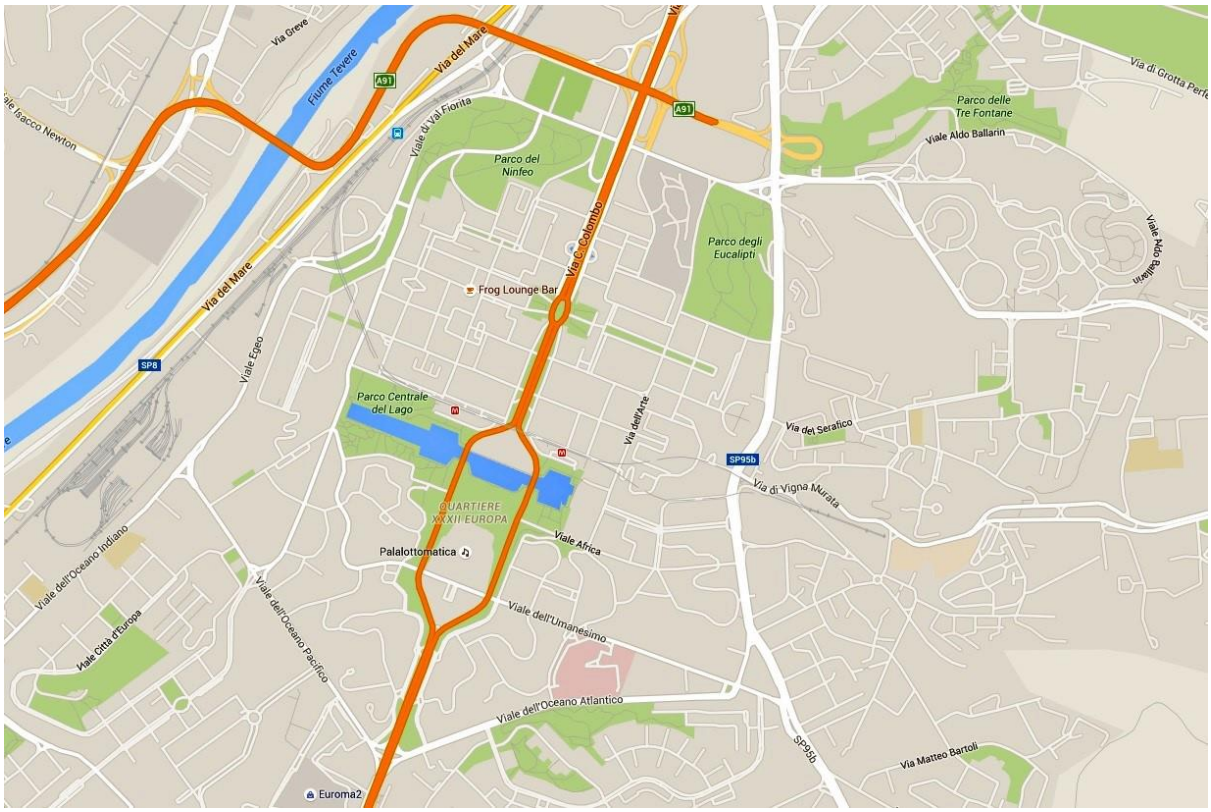
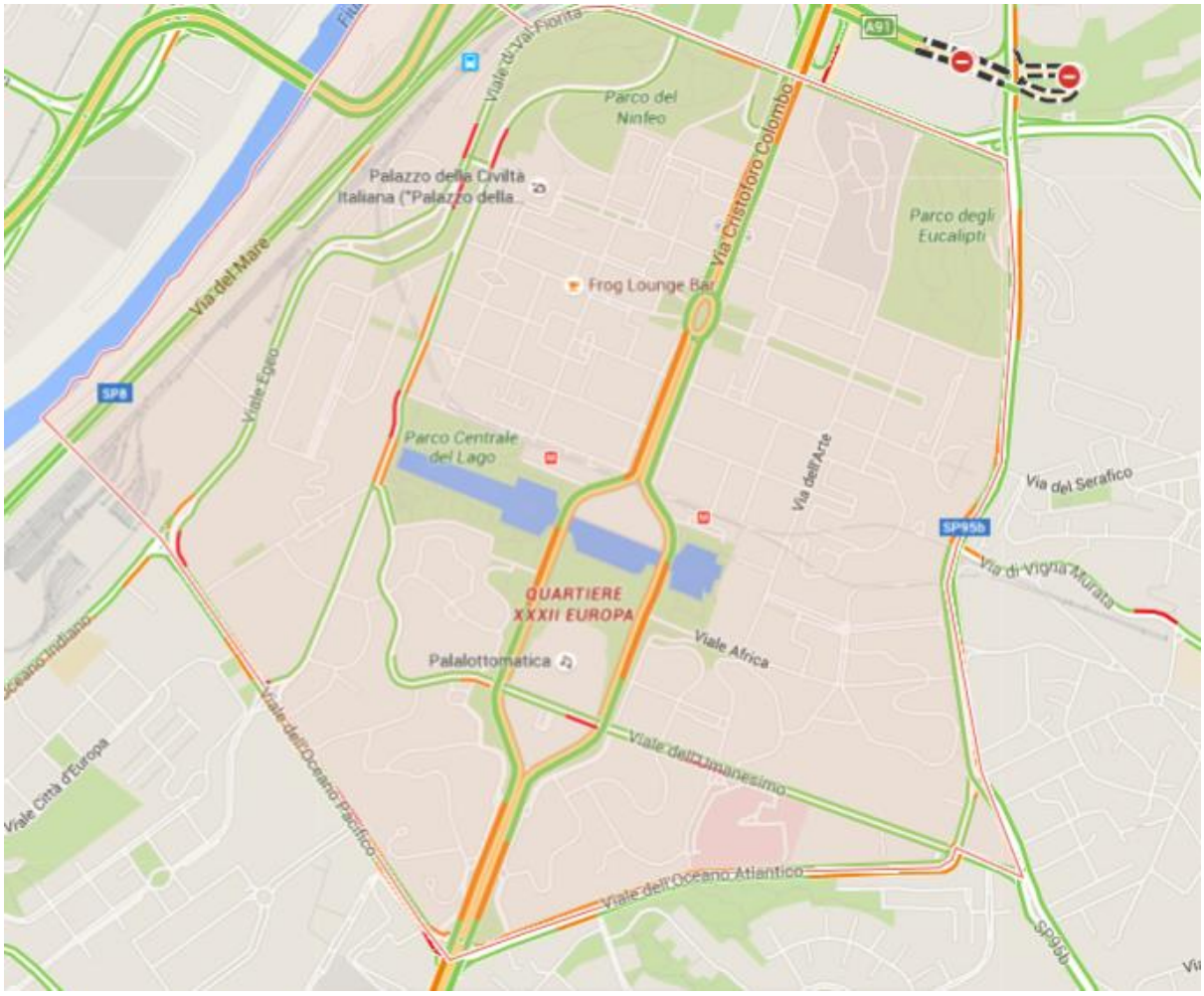


Figure 38: Google map showing the studied intersection

Selected intersections delimit half the pentagon that surrounding the district. We are in the western part of the Pentagon and in this part, the core of the district fastens to neighboring districts as the Torrino and Mostacciano. The increased flow of cars we have in the early hours of the morning and late in the afternoon as in the rest of the city. It is to be highlighted that most of the flow is due to the change of mode of transport to the Eur Magliana parking, and this worsens the traffic situation. The first phase of analysis has been prepared with a graphical table of the location of the Intersections and route.



Int. 1 Viale dell'Oceano Pacifico - Viale Avignone - Via-le Giorgio Ribotta

Int. 2 Viale dell'Oceano Pacifico - Viale della Tecnica - Viale grande Muraglia

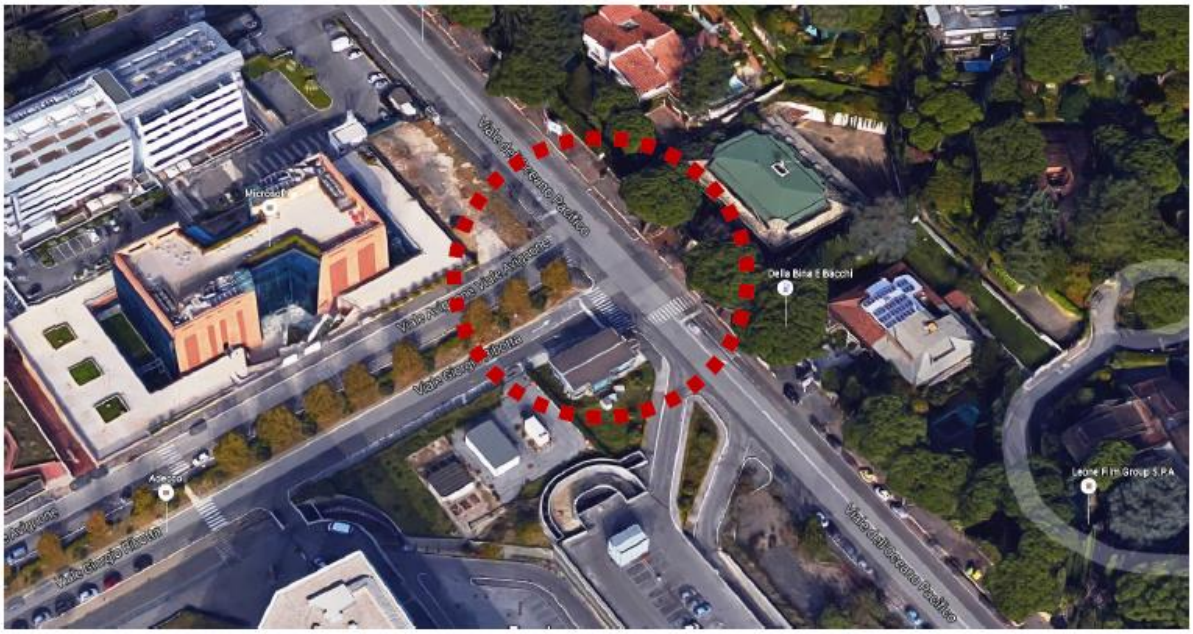


Figure 39: Int. 1 Viale Oceano Pacifico - Viale Avignone - Via-le Giorgio Ribotta

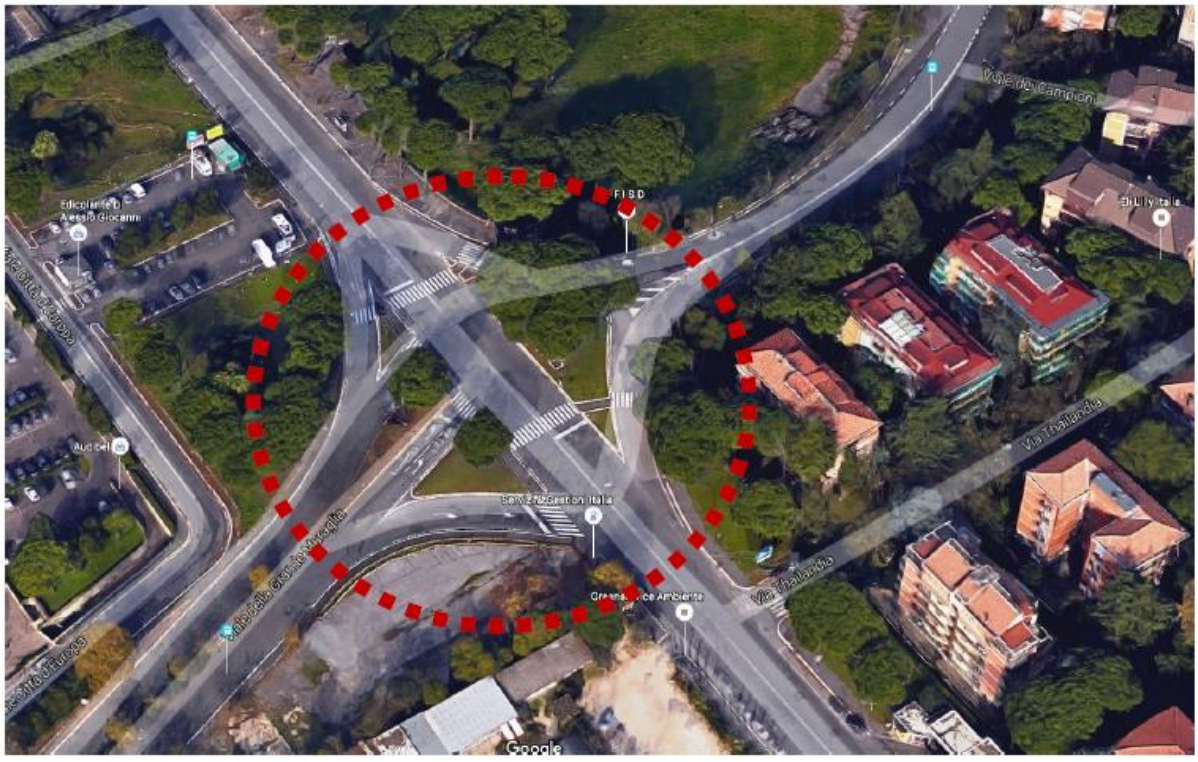


Figure 40: Int. 2 Viale Oceano Pacifico - Viale della Tecnica - Viale grande Muraglia

3.1 Operational Analysis of signalized junctions

HCM stands for highway capacity manual. The National Academy of Science in the united states publishes HCM. HCM contains various procedures for analysing and calculating the capacity, Level of service of the traffic systems such as freeway, bike lane, traffic intersection and pedestrians. There are several editions in the HCM. The fifth edition HCM 2010 has been used to compute the capacity and Level of Service for the intersections that we have selected.

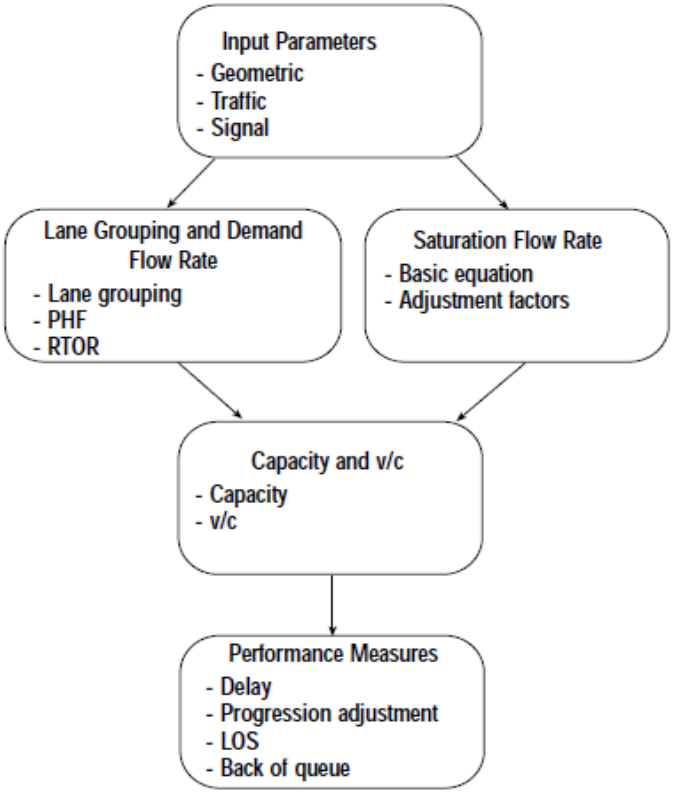


Figure 41:HCM methodology

3.2 Collecting Data

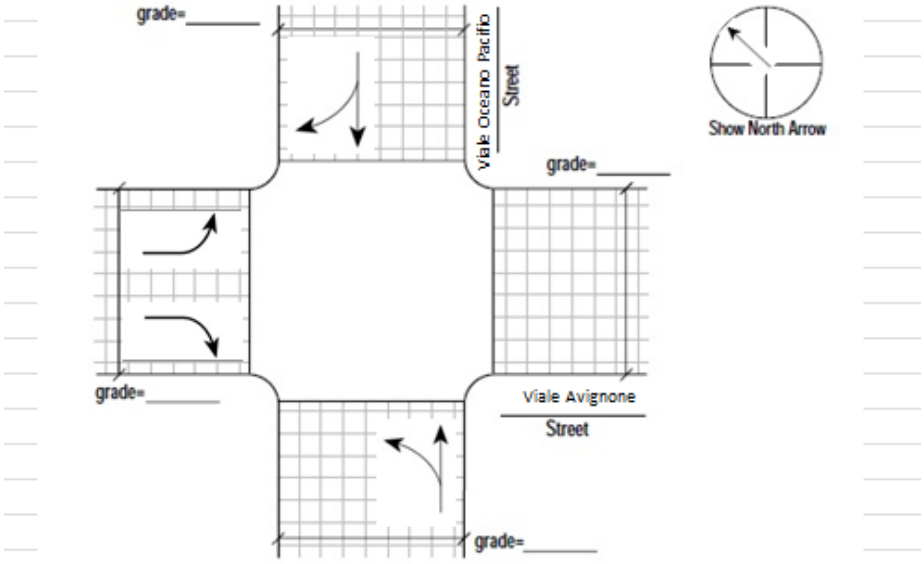
The input data were selected using survey following number of benchmarks, including: Geometric Conditions, Traffic Conditions, and signalization conditions. This is a scheme of the parameters collected during the Inspections.
Complete information regarding signalization is considered to perform the analysis. This information includes a phase diagram illustrating the phase plan, cycle length, green times, and change-and-clearance intervals. Lane groups operating under actuated control must be identified, including the existence of push-button pedestrian-actuated phases. If pedestrian timing requirements exist, the minimum green time for the phase is indicated and provided for in the signal timing. The methodology for signalized intersections is disaggregate; that is, it is designed to consider individual intersection approaches and individual lane groups within approaches. Segmenting the intersection into lane groups is a relatively simple process that considers both the geometry of the intersection and

the distribution of traffic movements. Traffic data can be collected by various techniques. The data can be fed into the system by means of special equipment called sensors. In addition, field visits do help to get the number of vehicles, signal timings, speed at a particular intersection. Highway capacity manual does provide input work sheet for the purpose of getting the data. This collected data helps us calculate the traffic demand, analysis of traffic and traffic control systems.

3.3 Operational analysis of interesection 1

Viale Oceano Pacifico - Viale Avignone - Via-le Giorgio Ribotta

INPUT WORKSHEET			
General Information		Site Information	
Analyst	Vivek Reddy Marredd	Intersection	Viale O. P.
Date Performed	16-mag	Area Type	CBD
Analysis Time Period	18.00 / 18.17	Jurisdiction	
Weather	Partially Cloudy	Analysis Year	



Volume and Timing Input	Eastbound		Northbound			Southbound		
	LT	RT	LT	TH	TH(LT)	RT	TH(RT)	TH
Volume, V (veh/h)	318	152	73	330	170	247	352	601
% Heavy Vehicles, % HV	0%	0%	2%	2%	2%	O%	1%	1%
Peak - Hour Factor, PHF	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Pretimed (P) or Actuated (A)	/	/	/	P	/	P	P	/
Start-up lost time, l1 (s)	1	1	1	1	1	1	1	1
Extension of effective green time, e (s)								
Arrival Type, AT	3	3	3	3	3	5	5	5
Approach pedestrian volume, 2^vped (p/h)	20		20			20		
Approach bicycle volume, 2^vbic (p/h)	0		0					
Parking (Y or N)	N		Y			N		
Parking Maneuvers, Nm (Manouvers/h)	10		/			/		
Bus Stopping, Nb (buses/h)	/		/			/		
Min. timing for pedestrians, 3^Gp (s)	17.35		17.35			17.35		

SIGNAL PHASING PLAN						
D	1		2		3	
I						
A						
G						
R						
A						
M						
Timing	G	Y	G	Y	G	Y
	68	5	24	5	48	5
Total Cycle L.	102					

Lane width	W	N,S
	3.75	3.75

Table 14:Input worksheet of car following model for intersection 1

VOLUME ADJUSTMENT AND SATURATION FLOW RATE WORKSHEET								
Volume Adjustment - Project Description	Eastbound		Northbound			Sorthbound		
	LT	RT	LT	TH	TH(LT)	RT	TH(RT)	TH
Volume, V (veh/h)	318	152	73	330	170	247	352	601
Peak - Hour Factori, PHF	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Adjusted flow rate, Vp = V/PHF (veh/h)	353.333	168.889	81.1111	366.667	188.889	274.444	391.111	667.778
Lane Group		2 Lane	2 Lane					
Adjusted flow rate in lane group, v (veh/h)	522.2222222		636.6666667			1333.333333		
Proportion of LT or RT (Plt or Prt)	0.6766		0.18114			0.20583		
SATURATION FLOW RATE								
Base saturation flow, S0 (pc/h/ln)	2200	2200	2200	2200	2200	2200	2300	2200
Number of lanes, N	2		2	2		2		
Lane width adjustment factor, fw	1.016666667		1.01667	1.01667		1.01667		
Heavy-vehicle adjustment factor, fHV	0.9		0.98039	0.98039		1		
Grade adjustment factor, fG	1		1	1		1		
Parking adjustment factor, fP	0.9		0.9	0.9		0.9		
Bus blockage adjustment factor, fbb	1		1	1		1		
Area type adjustment factor, fa	0.9		0.9	0.9		0.9		
Lane utilization adjustment factor, fLU	1		0.925	0.925		1		
Left turn adjustment factor, fLT	0.95		0.99	0.99		1		
Right turn adjustment factor, fRT	0.85		0.9025	0.9025		0.969		
Left turn ped/bike adjustment factor, fRPB	0.999		0.99	0.99		1		
Right turn ped/bike adjustment factor, fRPB	0.999		1	1		0.99		
Adjusted saturation flow, s (veh/h)								
s = s0 N fW fHV fG fP fBB fA fLU fLT fRT fLPB fRPB	2628.041971		2906.534222			3475.963854		

Capacity Analysis			
Phase number	1	2	3
Phase type	P	P	P
Lane group			
Adjusted flow rate	443.3333	437.7778	1611.111
Saturation flow rate	2906.534	2628.042	3475.964
Lost time	4	4	4
Effective green time	69	25	49
Green ratio	0.666667	0.235294	0.470588
Lane group capacity	1937.689	618.3628	1635.748
v/c ratio	0.228795	0.707963	0.984939
Flow ratio	0.15253	0.166579	0.463501
Critical lane group/phase			
Sum of flow ratios for critical lane groups, Yc	0.936757		
Total lost time per cycle	4	4	4
Critical flow rate to capacity ratio,Xc	0.974992		

Lane goup	NB	EB	SB
Adjusted flow rate,	443.3333	437.7778	1611.111
Lane goup capacity	1937.689	618.3628	1635.748
v/c ratio	0.228795	0.707963	0.984939
Total green ratio	0.666667	0.235294	0.470588
Uniform delay	6.704313	35.78449	21.66103
Incremental delay calibration	0.5	0.5	0.5
Incremental delay d2	0.275372	6.713707	18.95425
Initial queue delay d3	0	0	0
Uniform delay d1	/	/	/
Progression adjustment factor	0.7	0.57	0.5
Delay	4.968391	27.11087	29.78476
LOS by lane group	A	C	D
Delay by approach	4.968391	27.11087	29.78476
LOS by approach	A	C	D
Approach flow rate	443.3333	437.7778	1611.111
Intersection delay	61.86402		
INTERSECTION LOS	E		

Table 15:Saturation flow and level of service worksheets for intersection 1

Now the OSM data of this intersection is downloaded and loaded into SUMO. The resulted network image is shown below.

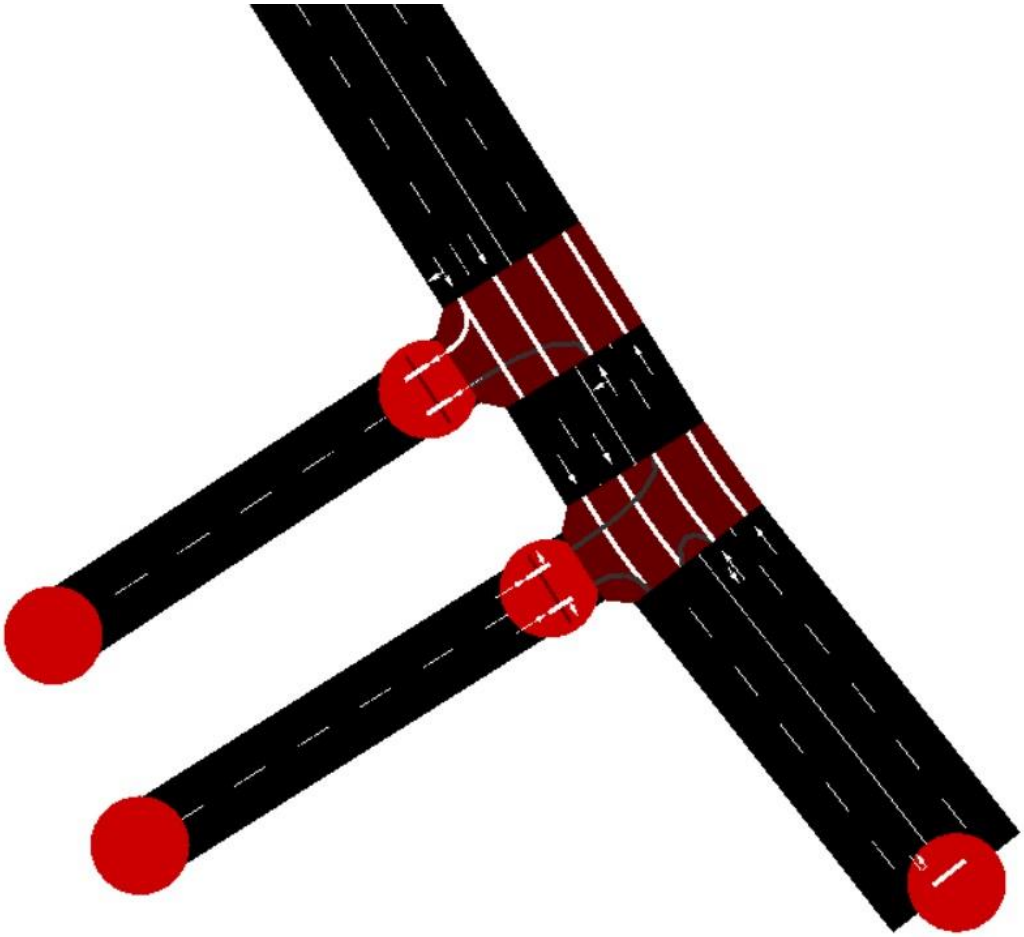


Figure 42: Network obtained by converting the OSM data

The resulted network the has been obtained from the SUMO has two intersections, because SUMO recognised Via dell'oceano pacifico-Viale Avignone as one intersection and Via dell'oceano pacifico-Viale Giorgio Ribotta as another intersection and signs both of them two separate traffic signals. There is a command in SUMO which is "**--junctions.join"** to join two intersections nearby and the resulted network is shown below.

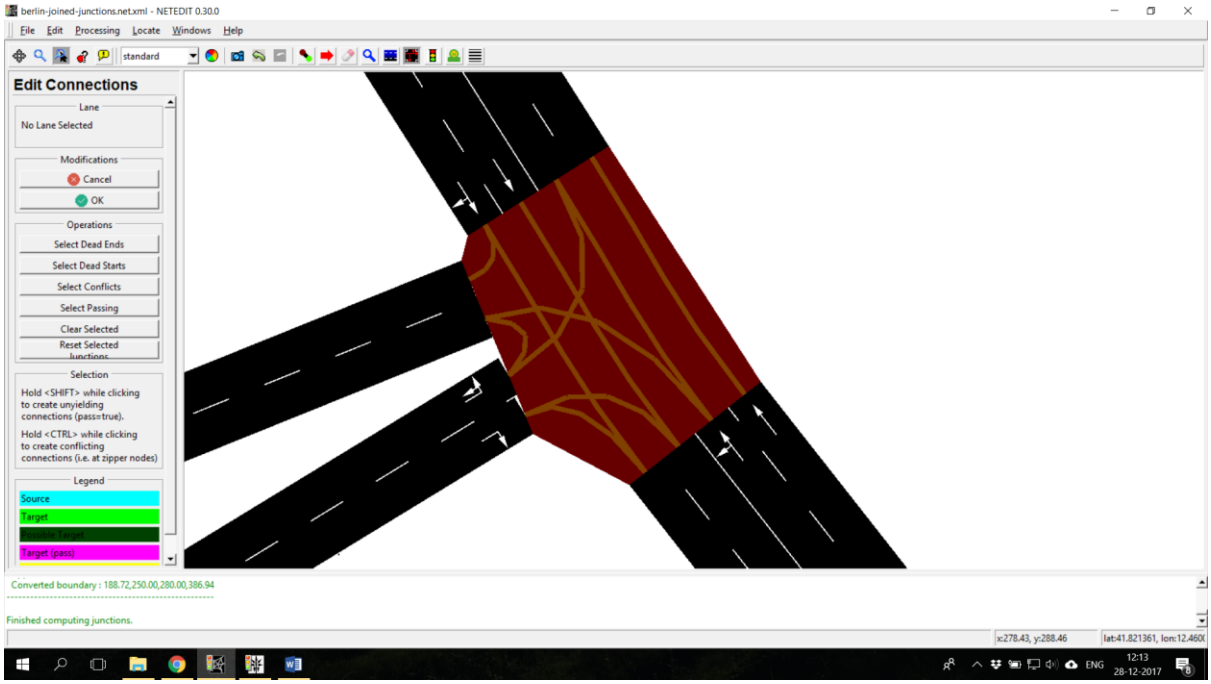


Figure 43: Corrected intersection

In the above image the shows that the traffic light has been incorporated and shown with the signal phasing according to the field visit.

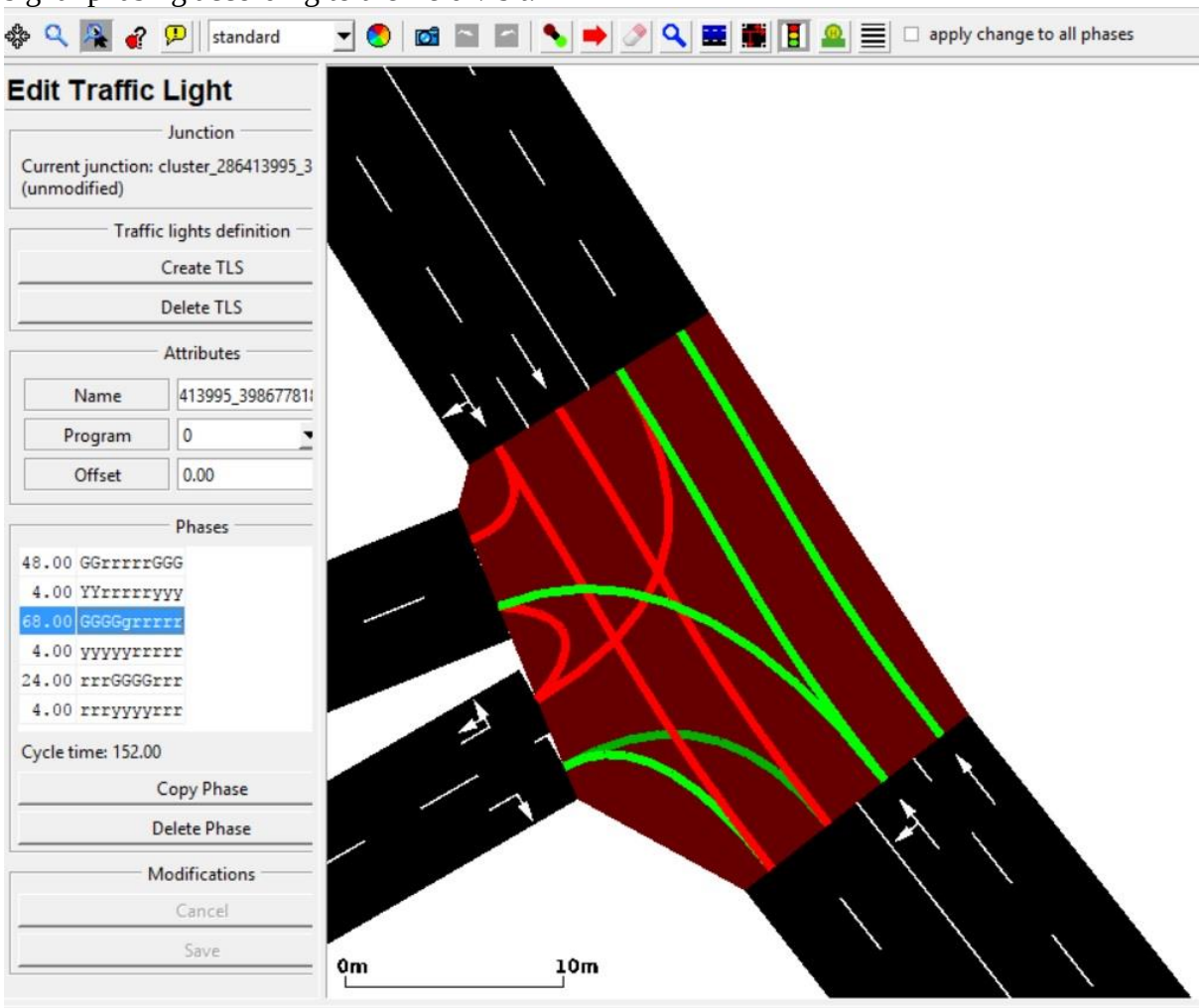


Figure 44: Traffic flow through the intersection

The phases column in the above image show the characters in which each character describes different states of traffic signals in the phase. There are three phases in this traffic signal setting. The following signal colours are used
r – red signal
y- amber (yellow) signal
G- Green signal
There is a need for eight different types of flows in this intersection. Next the vehicle flows through the intersection has been generated by the commands below.

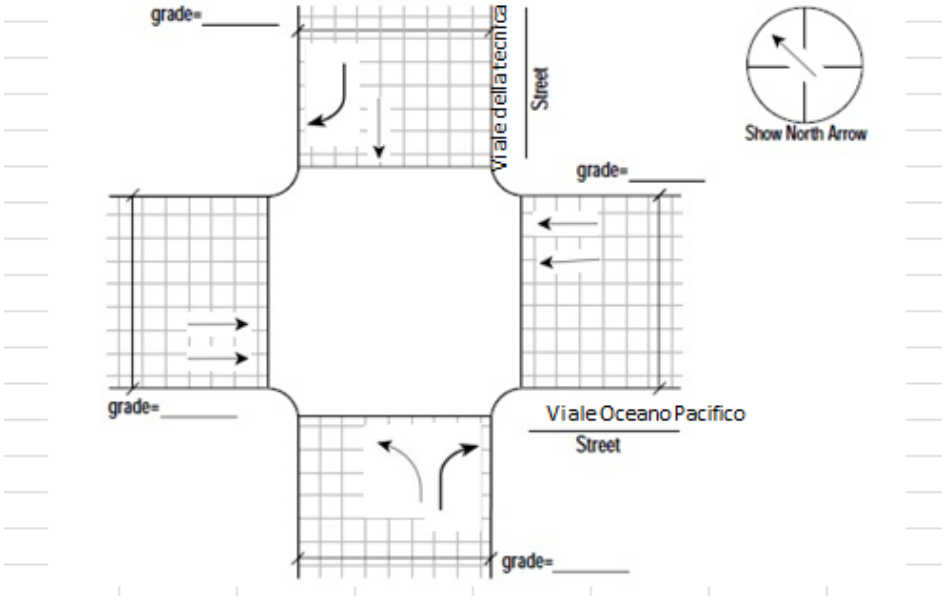
```
<routes>
   <vType accel="1.0" decel="5.0" id="Car" length="2.0" maxSpeed="13.0" sigma="0.5s" />
   <flow id="type1" color="1,1,0"  begin="0" end= "7200" vehsPerHour="318" type="Car">
   <route edges="gneE14 -119306556#2"/>
         </flow>
         <flow id="type2" color="1,1,0" begin="0" end= "7200" vehsPerHour="152"
type="Car">
   <route edges="gneE14 335286725#0"/>
         </flow>
         <flow id="type3" color="1,1,0" begin="0" end= "7200" vehsPerHour="73"
type="Car">
   <route edges="-335286725#0 gneE8"/>
         </flow>
         <flow id="type4" color="1,1,0" begin="0" end= "7200" vehsPerHour="330"
type="Car">
   <route edges="-335286725#0 -119306556#2"/>
         </flow>
         <flow id="type5" color="1,1,0" begin="0" end= "7200" vehsPerHour="170"
type="Car">
   <route edges="-335286725#0 -119306556#2"/>
         </flow>
         <flow id="type6" color="1,1,0" begin="0" end= "7200" vehsPerHour="247"
type="Car">
   <route edges="119306556#2 gneE8"/>
         </flow>
         <flow id="type7" color="1,1,0" begin="0" end= "7200" vehsPerHour="352"
type="Car">
   <route edges="119306556#2 335286725#0"/>
         </flow>
         <flow id="type8" color="1,1,0" begin="0" end= "7200" vehsPerHour="601"
type="Car">
   <route edges="119306556#2 335286725#0"/>
   </flow>
</routes>
```

The simulation was run using the above procedure. Sigma in the car following model describes the driver imperfection it ranges from 0 to 1. When the sigma is 0 the drivers vary their speed randomly. The average waiting time obtained from the simulation run using the above parameters if 49.6 seconds. The intersection delay obtained by the methodology of HCM is 61.8 seconds.

3.4 Operational analysis of interesection 2

Viale dell'Oceano Pacifico - Viale della Tecnica - Viale grande Muraglia

IMPUT WORKSHEET		
General Information		
Analyst	Vivek Reddy Marredd	Int. 2 Viale O.P./ Viale G.M.
Date Performed	16-mag	CBD Central Buisness District
Analysis Time Period	18.00 / 18.17	Municipio IX, Roma Capitale
Weather	Partially Cloudy	2016



Volume and Timing Input	Eastbound	Westbour	Northbound		Southbound	
	TH	TH	LT	RT	TH	RT
Volume, V (veh/h)	288	745	163	305	837	117
% Heavy Vehicles, % HV	2%	2%	2%	2%	1%	1%
Peak - Hour Factor, PHF	0.9	0.9	0.9		0.9	
Pretimed (P) or Actuated (A)	P	P	P		P	
Start-up lost time, l1 (s)	2	2	2		2	
Extension of effective green time, e (s)						
Arrival Type, AT	2	2	3	2	3	3
Approach pedestrian volume, 2^vped (p/h)				30	/	/
Approach bicycle volume, 2^vbic (p/h)	0	0	0	0	0	0
Parking (Y or N)	N	N		N	N	N
Parking Maneuvers, Nm (Manouvers/h)	/	/	/		/	/
Bus Stopping, Nb (buses/h)	/	/	/		/	/
Min. timing for pedestrians, 3^Gp (s) t	17.35	17.35	17.35		17.35	

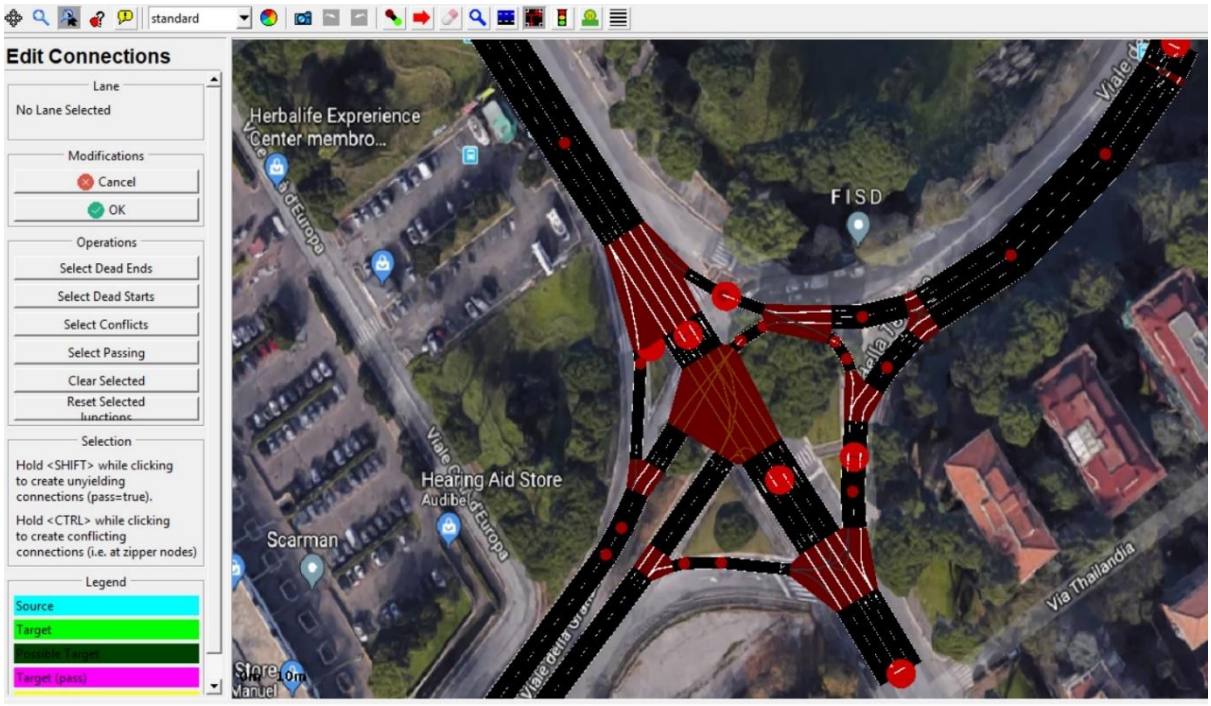
SIGNAL PHASING PLAN						
D	1		2		3	
I						
A						
G						
R						
A						
M						
Timing	G	Y	G	Y	G	Y
	25	5	39	5	31	5
Total Cycle L.	117					

VOLUME ADJUSTMENT AND SATURATION FLOW RATE WORKSHEET						
Volume Adjustment - Project Description	East	West	Northbound		Southbound	
	TH	TH	LT	RT	TH	RT
Volume, V (veh/h)	745	288	404	156	837	155
Peak - Hour Factori, PHF	0.9	0.9	0.9		0.9	
Adjusted flow rate, Vp = V/PHF (veh/h)	827.778	320	448.889	173.333	930	172.222
Lane Group			2	2	2	2
Adjusted flow rate in lane group, v (veh/h)			622.222		1102.22	
Proportion of LT or RT (Plt or Prt)	/	/	0.72143	0.27857	0.84375	0.15625
SATURATION FLOW RATE						
Base saturation flow, S0 (pc/h/ln)	1900	1900	1900	1900	1900	1900
Number of lanes, N	2	2	2	2	2	2
Lane width adjustment factor, fw	1.01667	1.01667	1.01667	0.6	1.01667	0.6
Heavy-vehicle adjustment factor, fHV	0.98039	0.98	0.9901	0.9901	0.9901	0.9901
Grade adjustment factor, fG	1	1	1	1	1	1
Parking adjustment factor, fP	1	1	1	1	1	1
Bus blockage adjustment factor, fbb	1	1	1	1	1	1
Area type adjustment factor, fa	0.9	`0.9	0.9	0.9	0.9	0.9
Lane utilization adjustment factor, fLU	0.925	0.925	1	1	1	1
Left turn adjustment factor, fLT	1	1	1	1	1	1
Right turn adjustment factor, fRT	1	1	1	0.85	1	0.85
Left turn ped/bike adjustment factor, fRPB						
Right turn ped/bike adjustment factor, fRPB						
Adjusted saturation flow, s (veh/h)					3442.57	
s = s0 N fW fHV fG fP fBB fA fLU fLT fRT fLPB fRPB	3153.16	3502.11	3442.57	1726.93		1726.93

General information				
Project Description				
Capacity Analysis				
Phase number	1	2	2	3
Phase type	p	p	p	p
Lane group				
Adjusted flow rate(v)	622.2222222	320	827.7778	1102.222222
Saturation flow rate(s)	3442.574257	3502.112	3153.162	3442.574257
Lost time	4	4	4	4
Effective green time	25	39	39	31
Green ratio	0.213675214	0.333333	0.333333	0.264957265
Lane group capacity(c)	735.59279	1167.371	1051.054	912.1350597
v/c ratio(X)	0.845878631	0.27412	0.787569	1.208398044
Flow ratio v/s	0.180743297	0.091373	0.262523	0.320173841
Critical lane group/phase				
Sum of flow ratios for critical lane groups, Yc	0.27211674			
Total lost time per cycle	4			
Critical flow rate to capacity ratio,Xc	0.281749191			

Lane goup				
Adjusted flow rate,	622.2222222	320	827.7778	1102.222222
Lane goup capacity	735.59279	1167.371	1051.054	912.1350597
v/c ratio	0.845878631	0.27412	0.787569	1.208398044
Total green ratio	0.213675214	0.333333	0.333333	0.264957265
Uniform delay	43.85697629	28.39466	28.39466	46.28306819
Incremental delay calibration	0.5	0.5	0.5	0.5
Incremental delay d2	11.51746706	0.581257	5.975656	104.0885126
Initial queue delay d3	0	0	0	0
Uniform delay d1				
Progression adjustment factor	1	1.286	1.286	1
Delay	11.51746706	0.581257	5.975656	104.0885126
LOS by lane group	B	C	C	F
Delay by approach	55.37444335	28.97592	34.37032	150.3715808
LOS by approach	B	C	C	F
Approach flow rate	622.2222222	320	827.7778	1102.222222
Intersction delay	82.8353			
LOS of intersection	F			

For the second intersection a background satellite image of the intersection has been inserted into the network model for enhancing the visualization of the intersection.



Once the network and route files are ready the configuration file has to be written which is described below.

```
<configuration>

   <input>
      <net-file value="2.net.xml"/>
      <route-files value="4.rou.xml"/>
   </input>
         <output>
   <queue-output value="results.xml"/>
   </output>
         <processing>
<ignore-route-errors value="true"/>

   <time>
      <begin value="0"/>
      <end value="1200"/>
   </time>
          <time-to-teleport value="-1"/>
</configuration>
```

SUMO software does not show the output results after the simulation has been run. The output results can be found in the output file results.xml.The average waiting time of a vehicle is 74.96seconds.The intersection delay computed by HCM methodology is 82 seconds.

3.5 Integrated Network

In this scenario a network with both the intersection have been develop and simulated. The value of demand is the numbers used from the field visits. Multi entry exit detectors were used in on the viale Oceano Pacifico lane to detect the average waiting time of the vehicle passing through these intersections. The image shown below displays the entire network that was developed by converting the openstreet map.

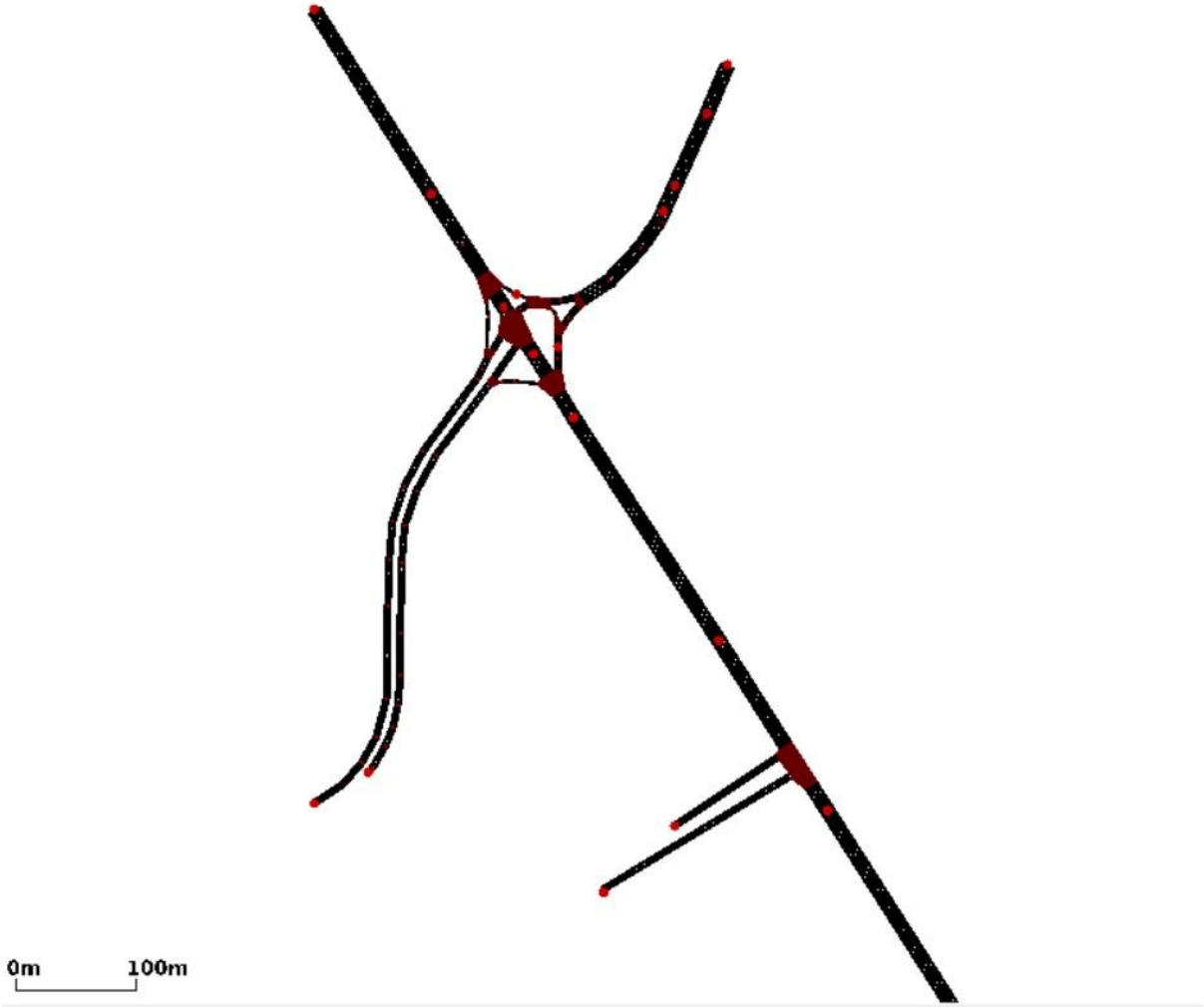


Figure 45: Converted network from OSM

Now the synchronization between the two signals has to be developed to have minimum waiting time on the viale dell'Oceano Pacifico. The Offset between the cycles can be selected by using the formula described in the highway capacity manual and it is compared with results from the simulation runs on SUMO. On SUMO network exit entry detectors have been used to calculate measure the mean speed, Average waiting time along the viale dell'Oceano Pacifico. The program for the exitenry detectors has to be written into an additional file in the format of xml as always. This program forces the SUMO software to write the output into an xml file. The program for the exit entry detectors is shown below.

```
<additionals>
<entryExitDetector id="Full" freq="100" file="fulle2e.xml"
   timeThreshold="10" speedThreshold="15">
      <detEntry lane="-335286725#2_1" pos="23"/>
      <detEntry lane="-335286725#2_1" pos="23"/>
      <detExit lane="119306574_1" pos="55"/>
      <detExit lane="119306574_0" pos="55"/>
   </entryExitDetector>
</additionals>
```

The travel time to pass these two intersections as determined by the simulation is 50.9 seconds. The average waiting time to pass the two interesctions on the Viale dell'Oceano Pacifico is 60.2 seconds. When both the intersections were synchronized the waiting time is 47 seconds which is a 21% reduction in travel time.

3.6 TraCI

Traffic control Interface, shortly known as TraCI. TraCI uses transmission control protocol based client/server acrhitechture. It permits to recover values of simulated objects and logically control their behaviour on-line. SUMO acts as a server when TraCI is intiated. SUMO doesn't not control the road traffic simulations after TraCI is initiated , it waits for the external application to come over and take control of the simulation.

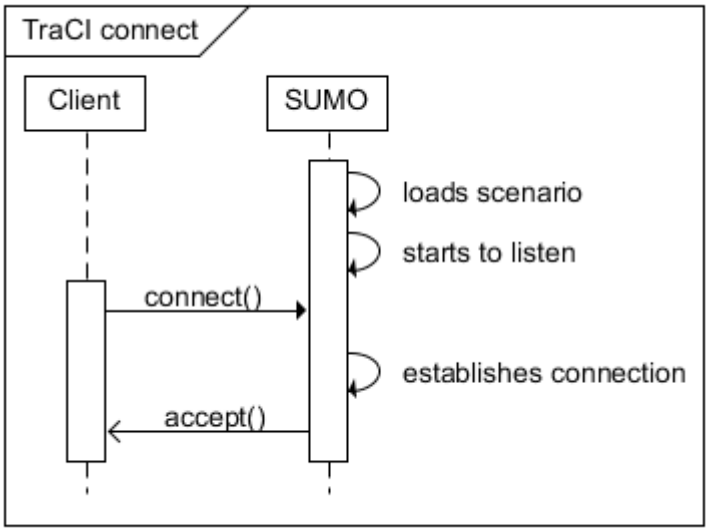


Figure 46:Figure showing the communication between the Client and SUMO during simulation

The above picture show the communication between the SUMO acting as a server with a client software. When the TraCI is initiated SUMO loads the scenario and waits for the client software then after the client software is connected the simulation can started by the play button in SUMO-GUI. The are many categories in TraCI package. Each package has a different function. They are edge, gui, Indcution loop, junction, lane, lane area, multientry and exit, person, polygon, route, simulation, traffic light, vehicle, vehicle type, connection, constants, domain, exceptions and storage. Each category has various functions and they can be used in the python program according to their functionality.

3.7 TraCI traffic light

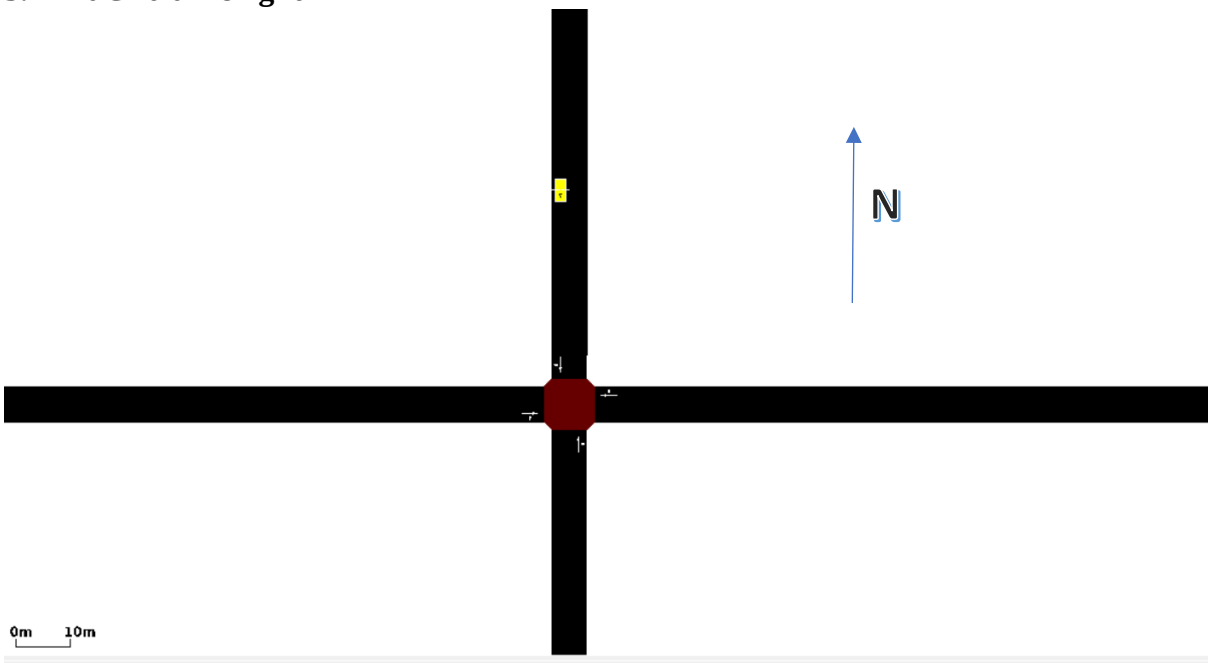
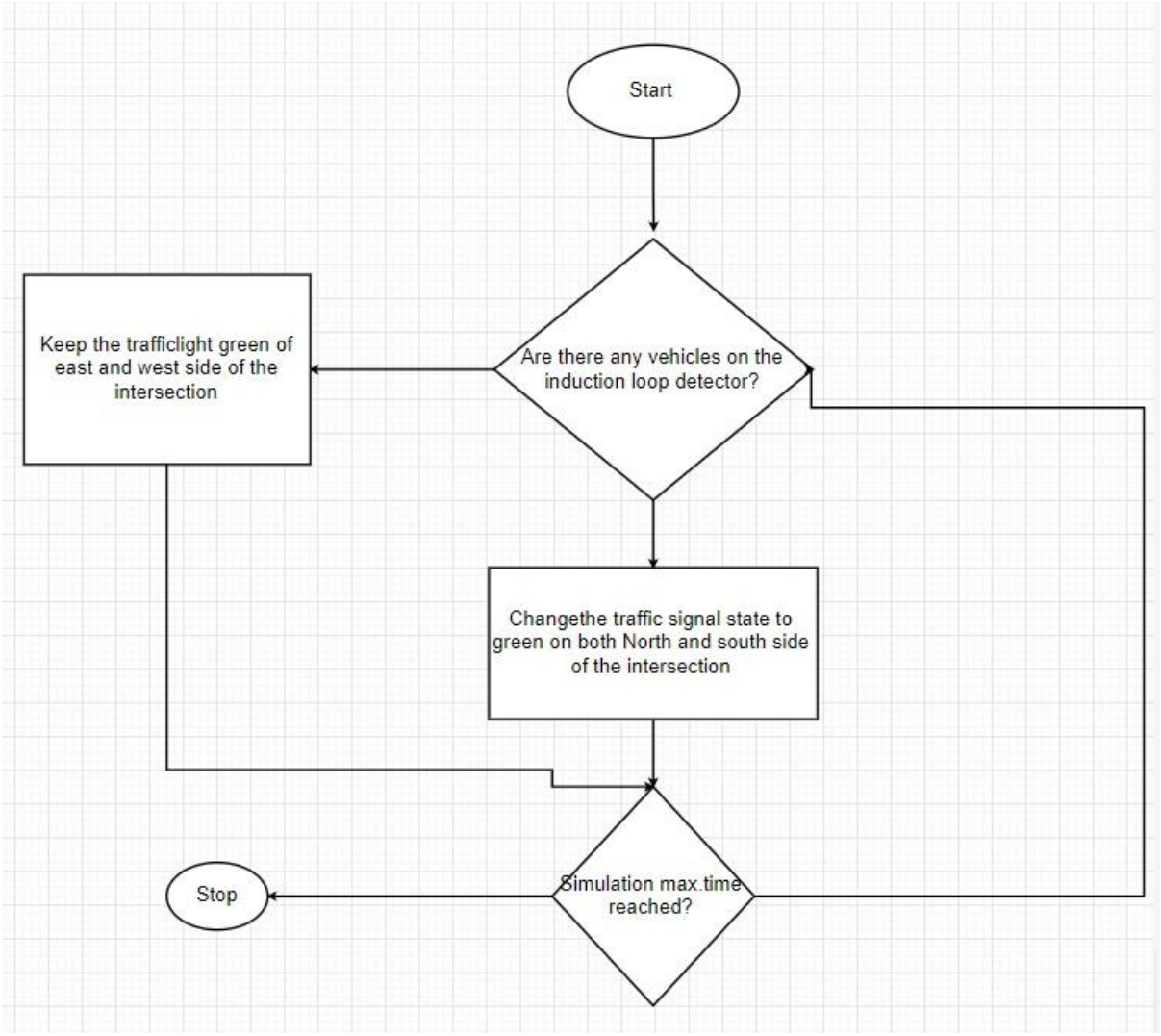


Figure 47: Traffic light with priority

The above figure shows the intersection with a induction loop detector place on the north bound approach of the intersection. This intersection is used to detect the vehicles passing from north to south through the intersection. A python script has been written for this intersection to regulate the traffic signal. This python program opens the TraCI terminal in SUMO. Then detects if there are any vehicles on the induction loop. If there are no signals passing from north to south, then the East and west signals of the intersection remain green. As soon as the induction loop detects a vehicle, the python program that has been developed changes the state of the traffic signal so that the traffic signals to North and south remain green. As the phase finishes the traffic signal return to the previous state which is green signal on east and west approaches. TraCI means traffic control interface. Python program communicates with SUMO via TraCI. The communication between SUMO and python is bidirectional.


Start
Are there any vehicles on the induction loop detector?
Keep the trafficlight green of east and west side of the intersection
Changethe traffic signal state to green on both North and south side of the intersection
Simulation max.time reached?
Stop

4.Traffic signal Synchronization

Traffic signal Synchronization is a technique to match the green times of the traffic signals along the street. This method is used where there is a high volume of traffic. Drivers loose patience when they have to stop at every intersection on the main street when there is few or no traffic on the side street. All the signals in the intersection should have the same cycle length.

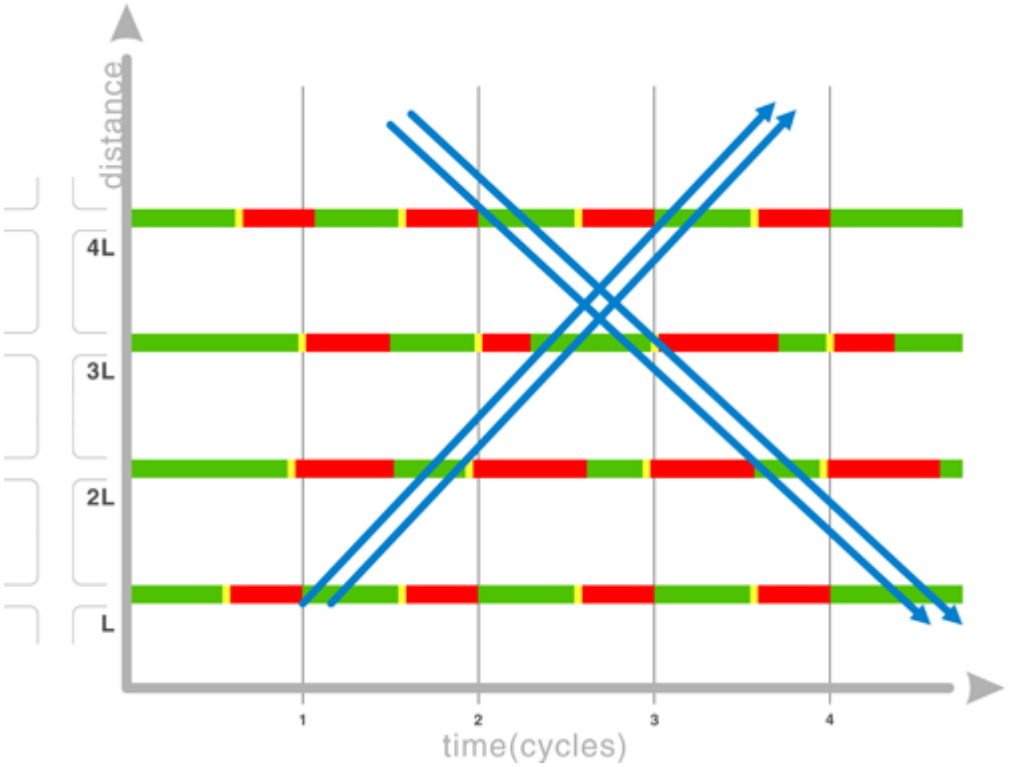


Figure 48:Trafffic signal synchronization along four signals

The above picture represents the traffic movement along the synchronized intersections on a time space graph. The blue lines show the traffic movement along the street. As soon the vehicle approaches the signal 2L the state of the traffic signal is already so no time is wasted at this intersection and the next two signals. Synchronization does not change the green time of the intersection.

4.1 A scenario with four intersections

A network was developed in NETEDIT with four intersections with a gap of 320 metres. All the four-traffic signal have the same cycle length. The active lane area detector of 300 metres length in front of the traffic signal gneJ12 as shown in the image below.

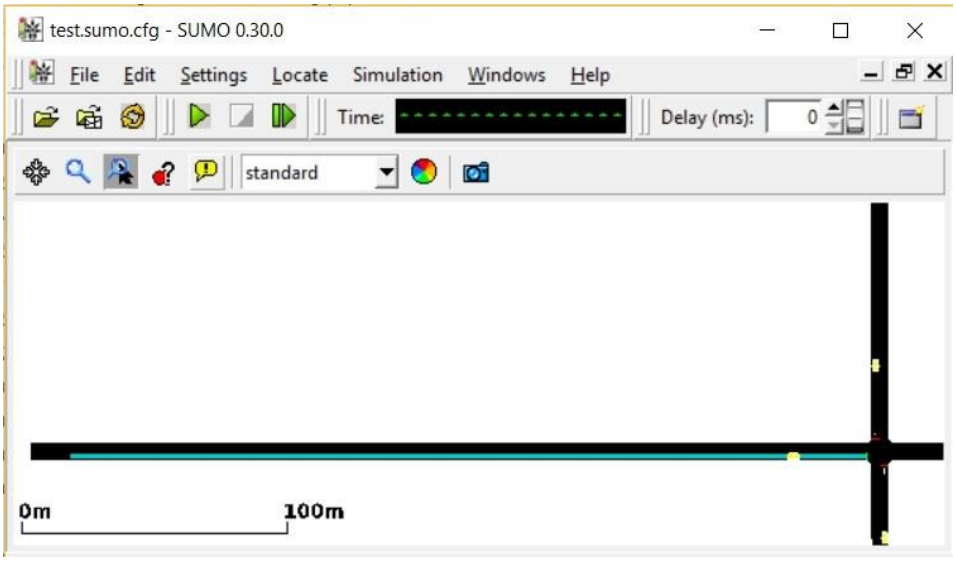


Figure 49: Lane area detector

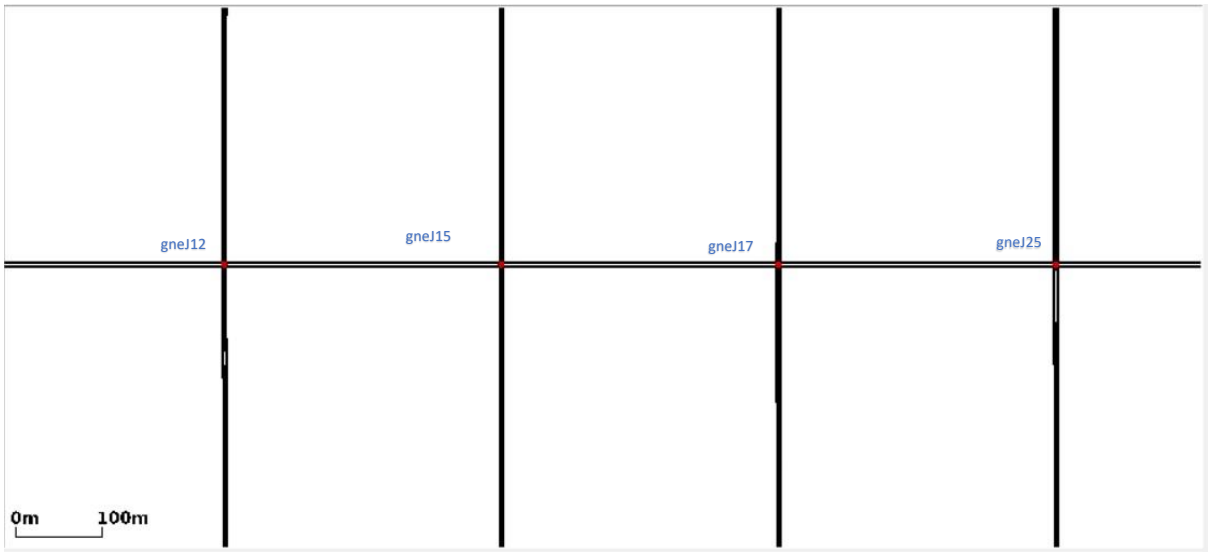
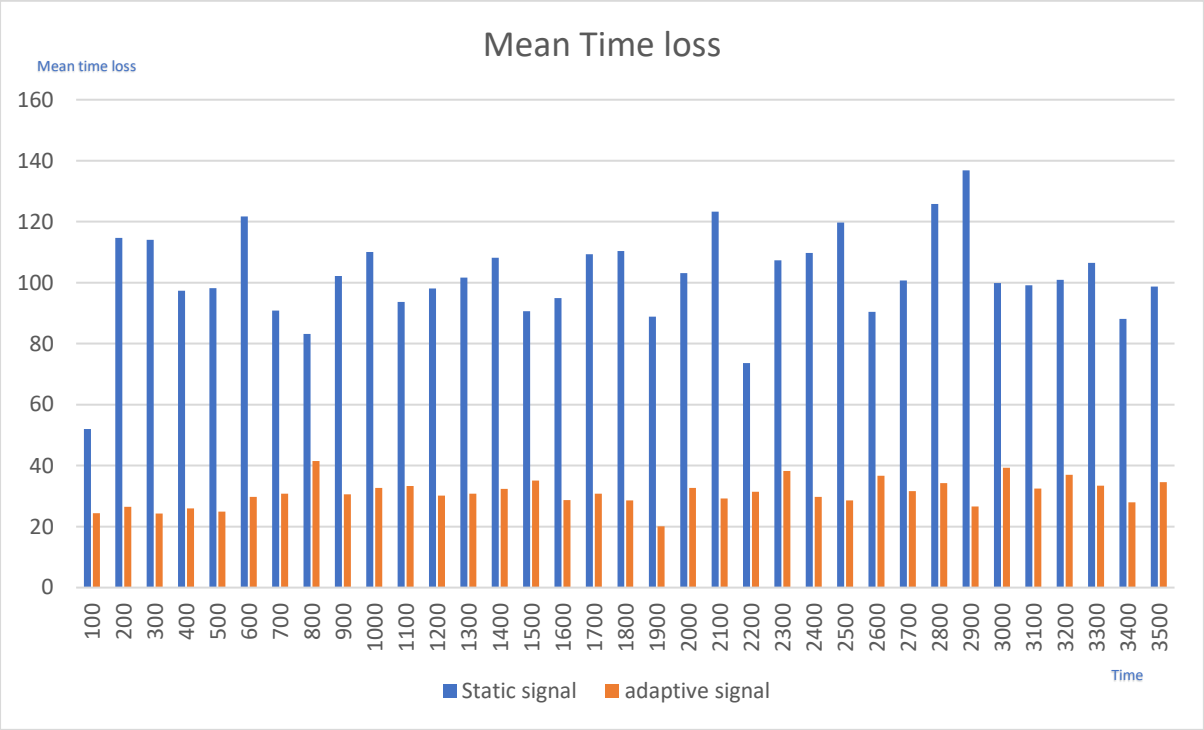


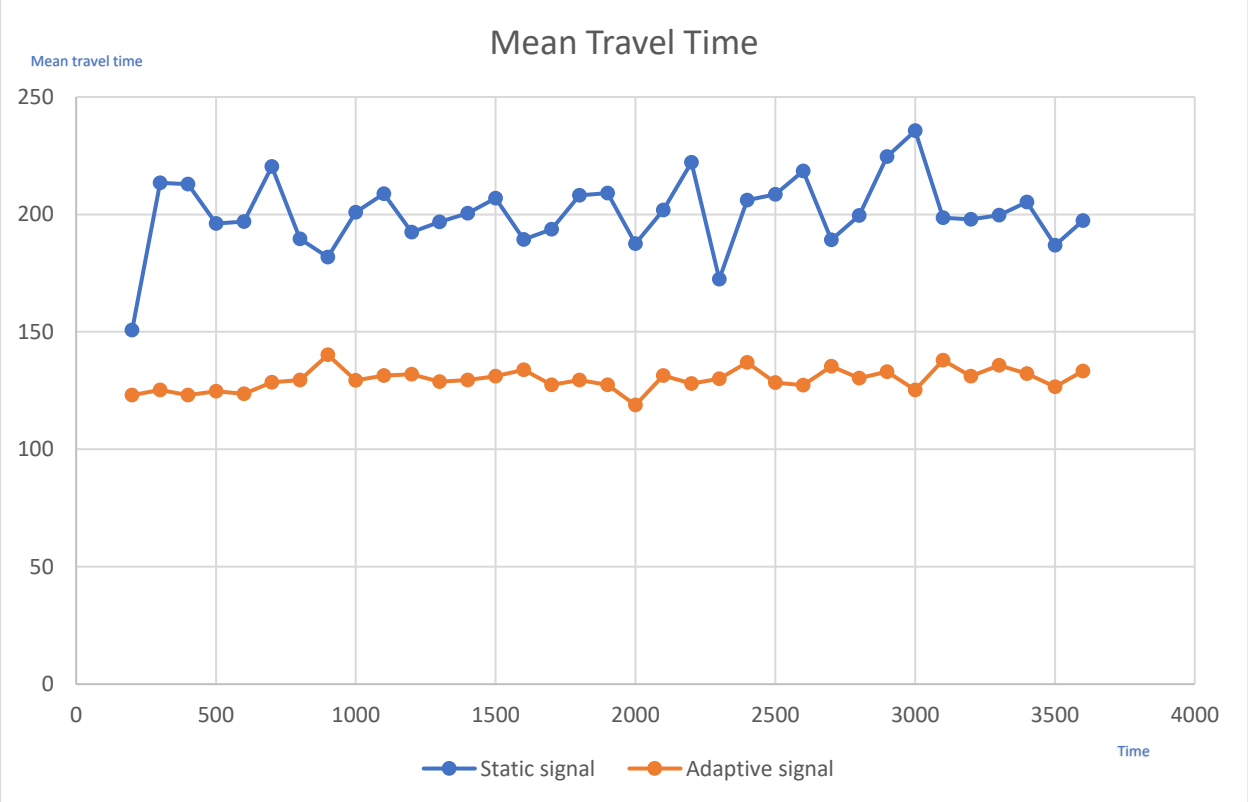
Figure 50: Network with four traffic signals

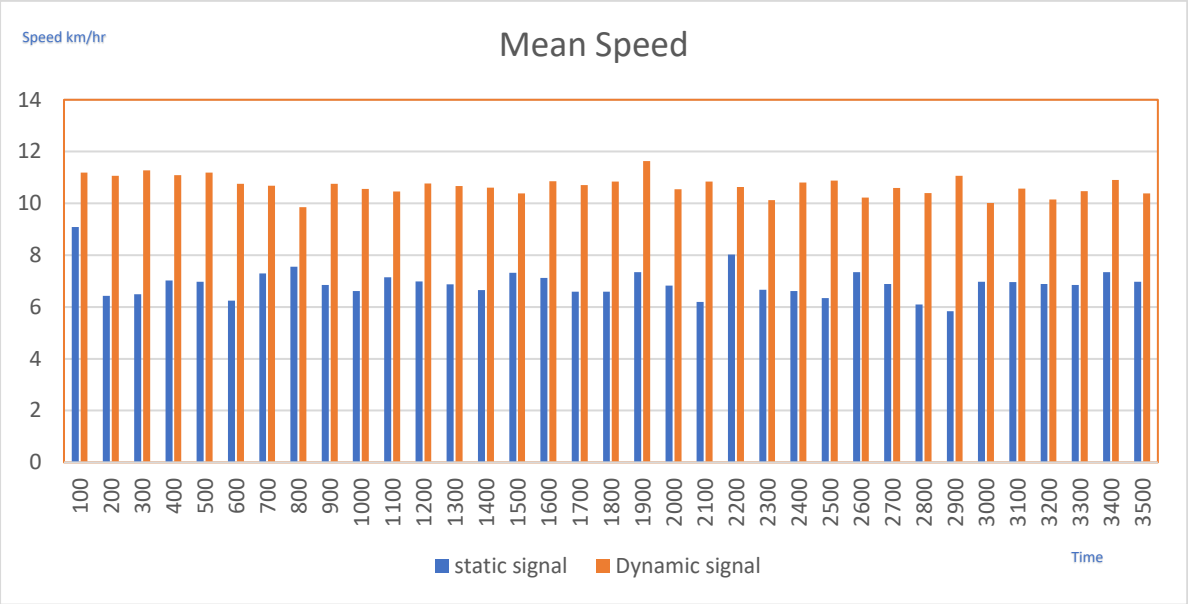
```
<    <tlLogic id="gneJ12" type="actuated" programID="0" offset="0">
        <param key="max-gap" value="13.0"/>
        <param key="detector-gap" value="2.0"/>
        <param key="show-detectors" value="true"/>
        <param key="detectorRange" value="300" />
        <param key="file" value="NULL"/>
        <param key="freq" value="300"/>
        <phase duration="38" minDur="15" maxDur="38" state="rGrG"/>
        <phase duration="4" state="yyyy"/>
        <phase duration="24"  state="GrGr"/>
        <phase duration="4" state="yryr"/>
    </tlLogic>
```

The signal gneJ12 is an actuated signal. The other signals are coordinated with gneJ12 signal. The coordination is done by the python communicating with SUMO via TraCI. Max-gap is the gap that the two consecutive vehicles. If max gap is greater than 3 metres and the time is greater

than the minimum phase duration the traffic light changes the phase. If the SUMO detects the continuous flow it keeps the traffic phase until maximum phase duration. The detectorRange is the distance before the traffic signal until which the detector is active. Here in this network the detectorRange is 300 metres. Lane area detectors are always used for actuated traffic light. Lane area detector is responsible giving the data about the vehicles that are approaching the traffic signal.







The simulation results show that there is 30% reduction in travel time, 60% reduction in time lost and 50% increase in mean speed. To the above scenario three additional schemes were tested on this network. Those include

A: Traffic signals with fixed time cycle

B: Traffic signals based on delay

C: Traffic signal based on continuous flow

D: Synchronized adaptive traffic signal with active detectors

Scheme	A	B	C	D
Mean Travel Time(s)	180	170	176	146
Mean Time Loss(S)	81	71	95	51
Mean Speed(m/s)	7.69	8.17	7.1	10.6
Co_2 Released(g)	382	365	378	311

Table 16 Comparison between different traffic schemes

As seen in the above table scheme D is the best performer in all the key performances shown above

4.2 A scenario with five intersections

An ideal network for testing the synchronization of signal is developed which contains five consecutive signalized intersections with equal space between them. The first signal from the east is an actuated signal which observes the traffic flow passing through the intersection and the other intersection are synchronized dynamically according to this actuated signal.

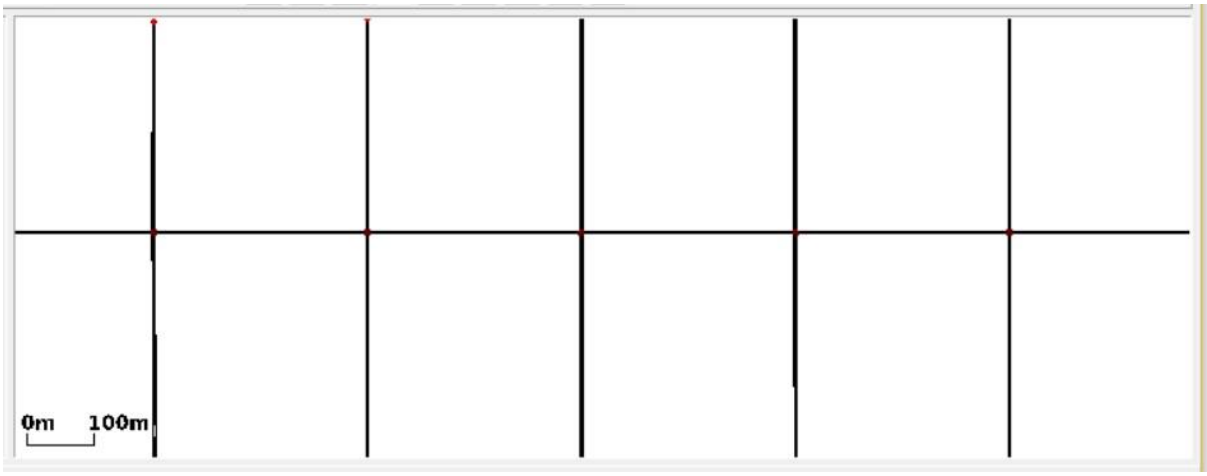


Figure 51:Network with five equidistant intersection

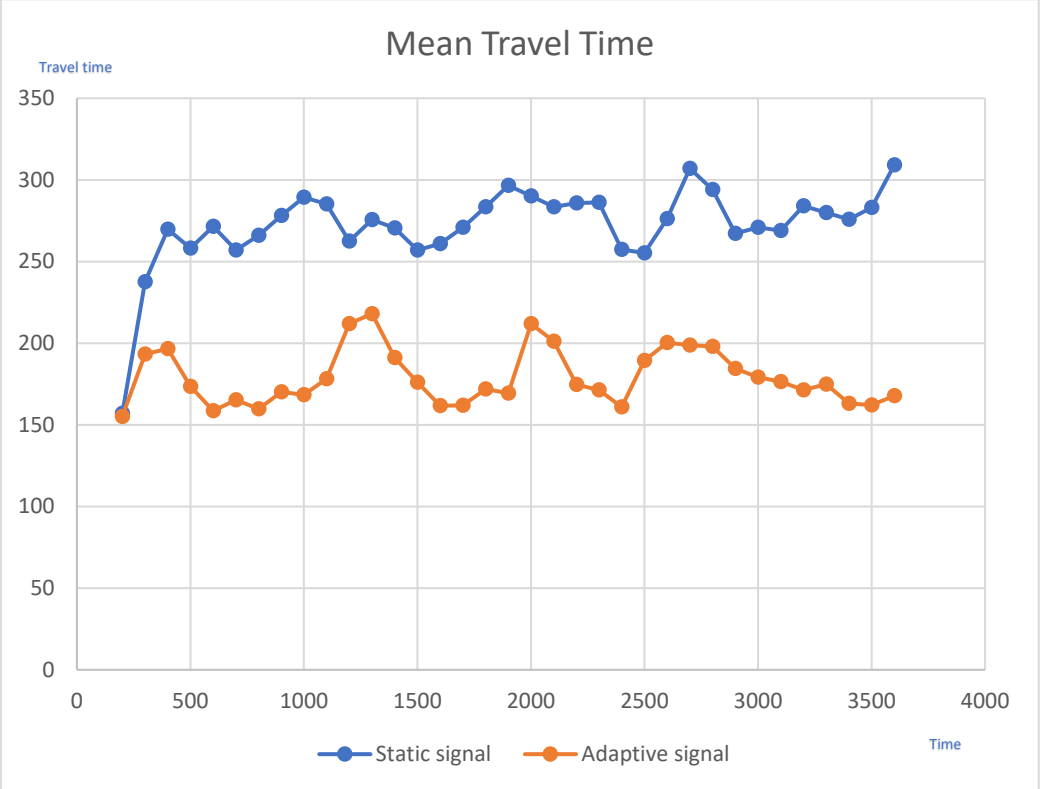


Figure 52 Mean travel time

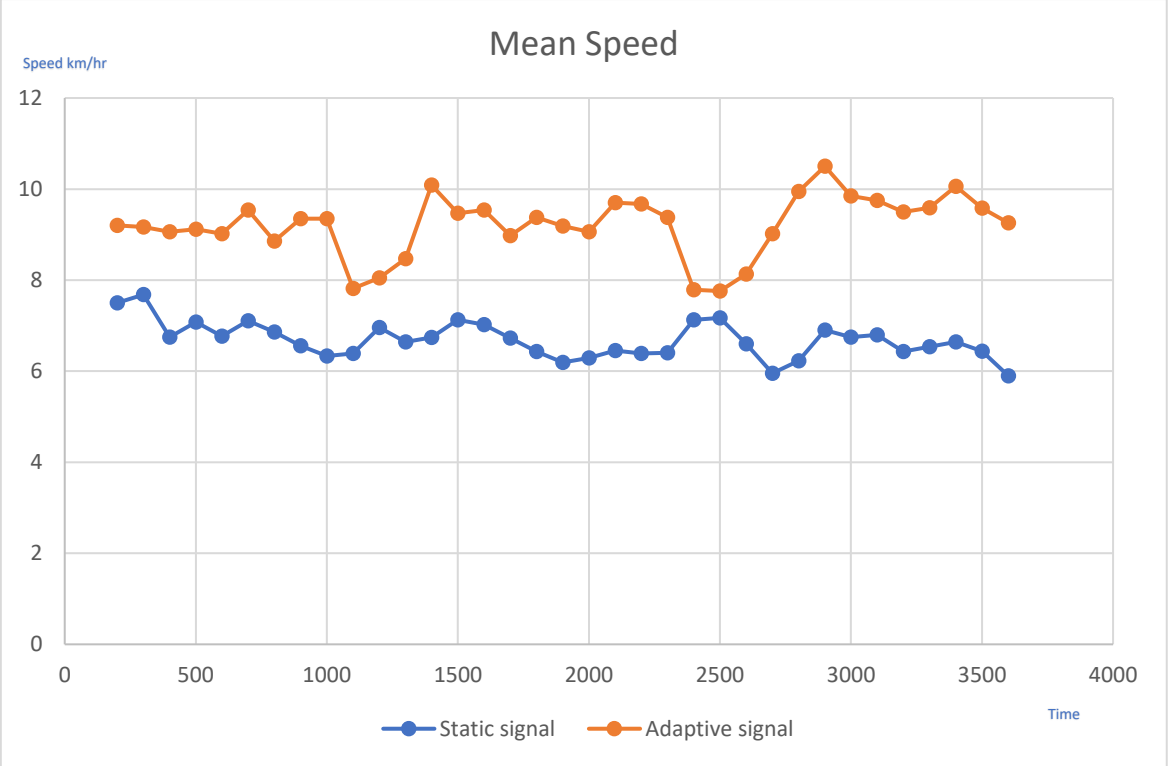


Figure 53 Mean speed

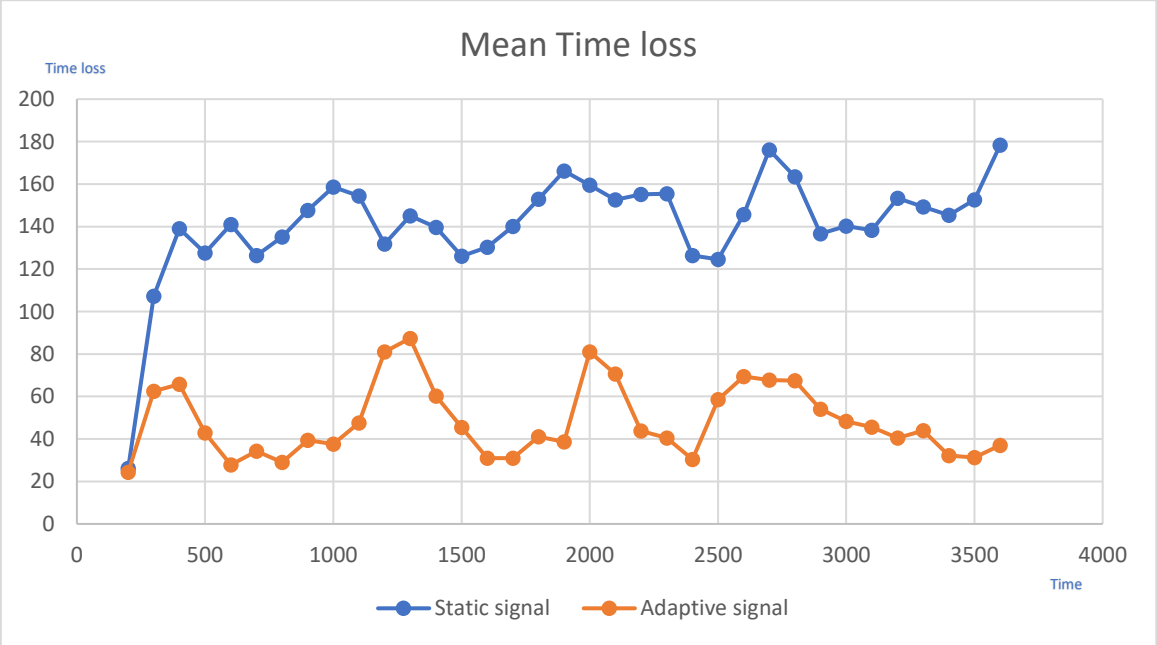


Figure 54 Mean time loss

Scheme	A	B	C	D
Mean Travel Time	218	199	210	179
Mean Time Loss	102	79	90	58
Mean Speed	7.01	8.1	7.6	10.2
CO_2 emissions	463	427	453	381

Table 17Comparison between different traffic control schemes

In terms of every KPI the scheme A performed poorly with respect to other schemes. The D scheme outperforms every other scheme mentioned in the table above.

4.3 A scenario with six intersections

This scenario has the same characteristics as the previous scenario, but this network has six signalized intersections. All the intersections on the network are equally spaced.

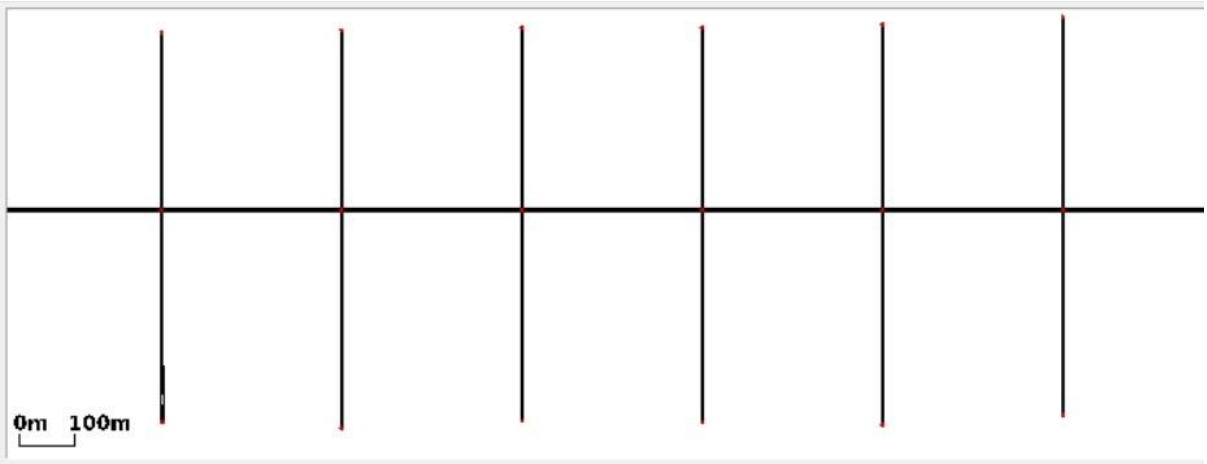


Figure 55:Network with six equidistant intersections

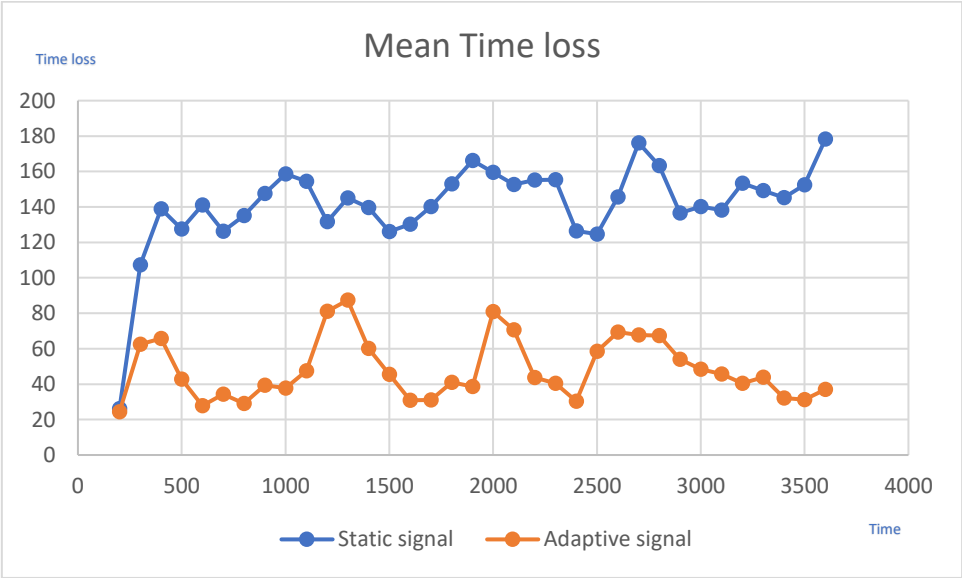


Figure 56 Mean time loss

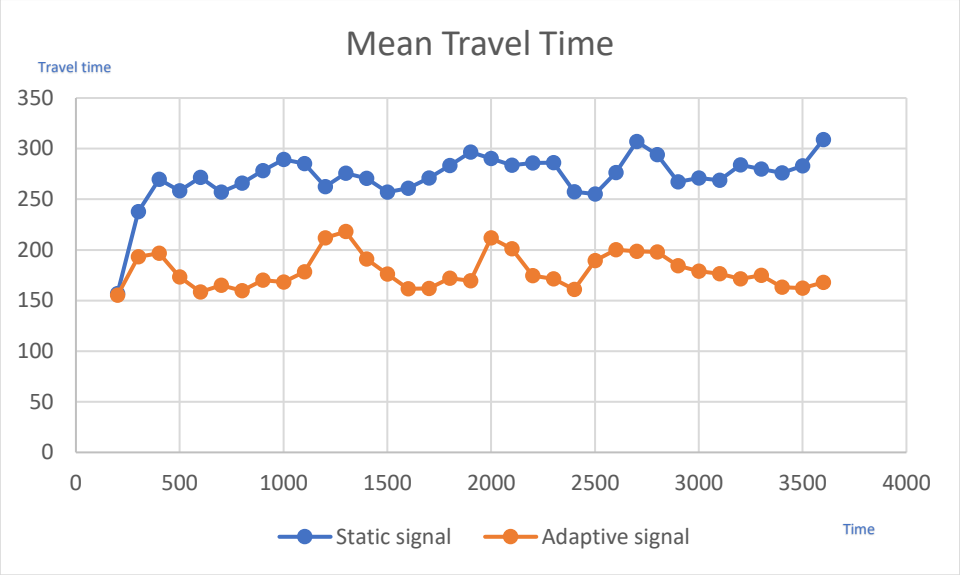


Figure 57 Mean travel time

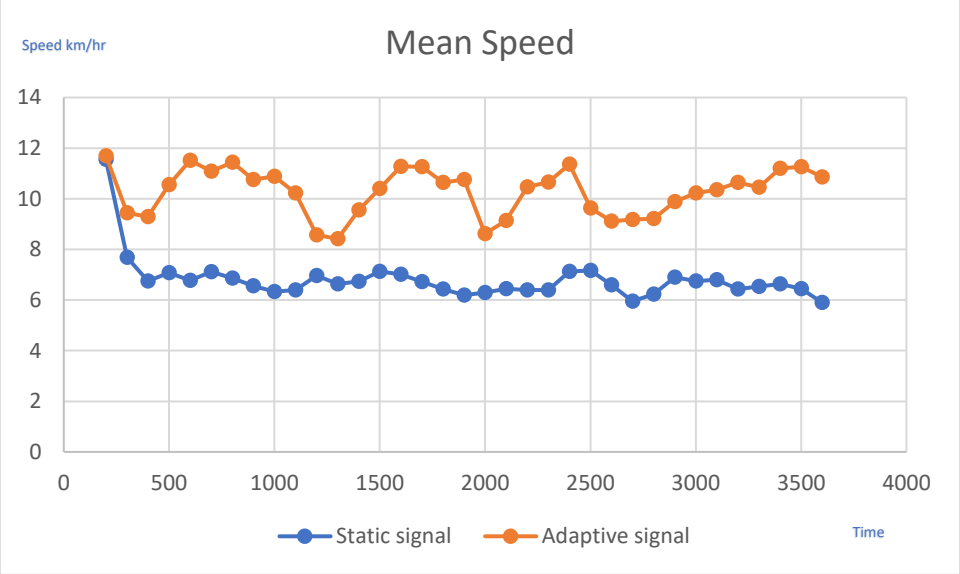


Figure 58 Mean speed

Scheme	A	B	C	D
Mean Travel Time	266	227	230	179
Mean Time Loss	139	97	103	65
Mean Speed	6.4	9.6	9.3	10.2
CO_2 emissions	565	494	514	442

Table 18: Comparison between different traffic schemes

4.4 Adaptive traffic signal with regulating maximum speed of the street.

In this simulation environment, the traffic signals are adaptive with the maximum velocity of the arteries are dynamic. The green time of the traffic signal is divided into two section out of which the first section is fixed and the second section is dynamic. The figure below shows the timing of the green phase in SUMO.

```
I    <phase duration="15" state="rgrG"/>
II   <phase duration="10" minDur="1" maxDur="15" state="rGrG"/>
     <phase duration="4" state="rYrY"/>
     <phase duration="24" state="GrGr"/>
     <phase duration="4" state="YrYr"/>
```

Figure 59: Phase timing

During the second phase the traffic signal is adaptive. The simulation is initiated by TraCI and the python program regulates the maximum velocity of the street. The initial maximum velocity is 15 m/s during the phase II the maximum velocity of the street is extended to 20 m/s, when the phase II finishes the maximum velocity is return to initial value which is 15 m/s . The output is taken from the simulation by using a multi-entry-exit detector. The results are shown in the figures below. The simulation was run for a duration of 3600 seconds

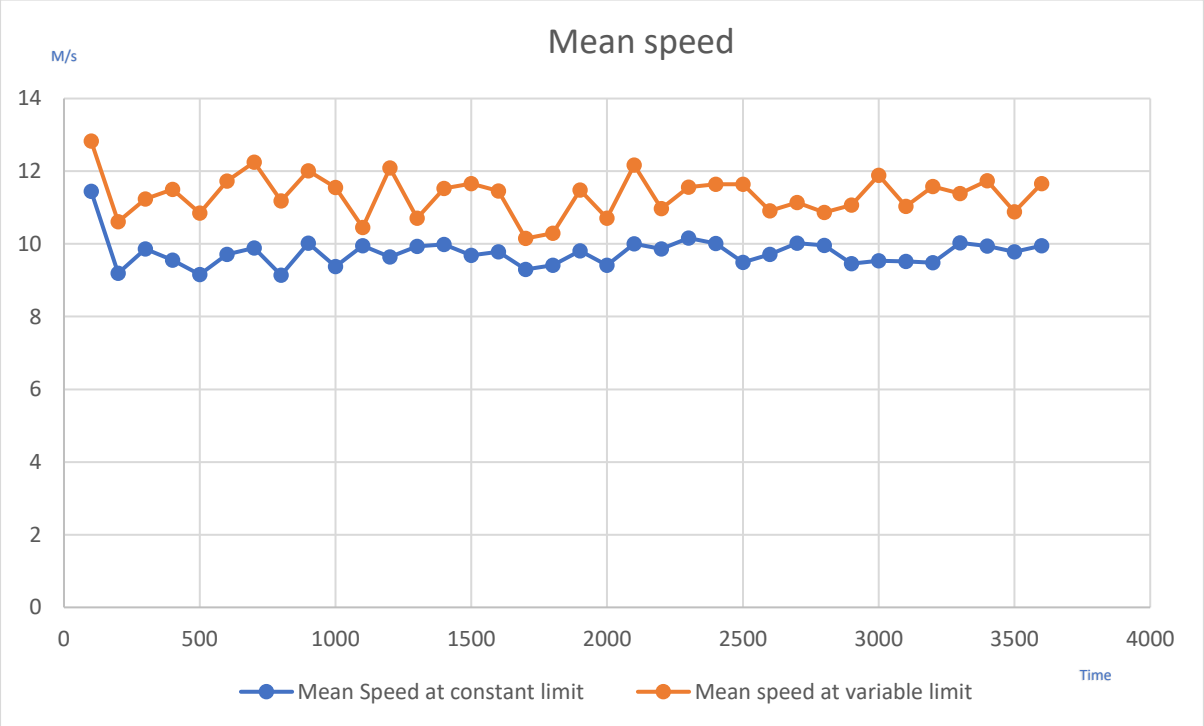
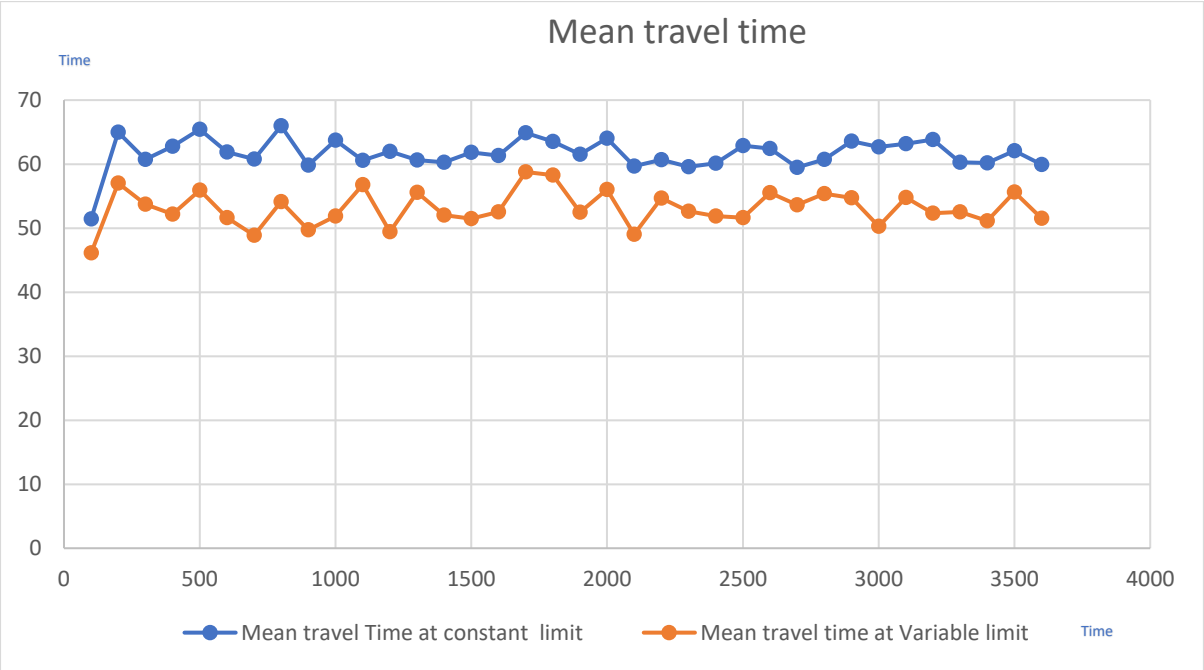


Figure 60 Mean Speed

As seen in the above figure the mean speed is always higher when there is a variable limit.

Fixed time signal with constant speed limit			Adaptive signal with variable speed limit	
sections	I	II	I	II
Control	Fixed	Fixed	Fixed	Flow responsive
Maximum velocity(m/s)	15	15	15	20
Phase time(sec)	15	15	15	1-15

Table 19



The meant travel time with the variable limit is always lower than the mean travel time at constant limit. Now during the simulation an alternative approach was chosen to increase the safety of the passengers because the approach speed at the intersection is high when the light changes to amber because of the increased speed limit. The phase timing is divided into three sections and they are shown in the picture below.

```
I    <phase duration="10" state="rgrG"/>
II   <phase duration="5" minDur="1" maxDur="15" state="rGrG"/>
III  <phase duration="5" state="rgrG"/>
     <phase duration="4" state="rYrY"/>
     <phase duration="24" state="GrGr"/>
     <phase duration="4" state="YrYr"/>
```

In this scheme the first and third sections are fixed timings and the second section is adaptive which is based on the traffic flow. The velocity of the streets is increased to 20 m/s when the phase is adaptive and then decreased to 15 m/s during the phase III. The results are shown in the pictures below. Multi entry exit detectors were used to generated output in this simulation.

Fixed time signal with variable speed limit			Adaptive signal with variable speed limit and safety		
sections	I	II	I	II	III
Control	Fixed	Flow responsive	Fixed	Flow responsive	Fixed
Maximum velocity(m/s)	15	15	15	20	15
Phase time(sec)	15	1-15	10	1-15	5

Table 20

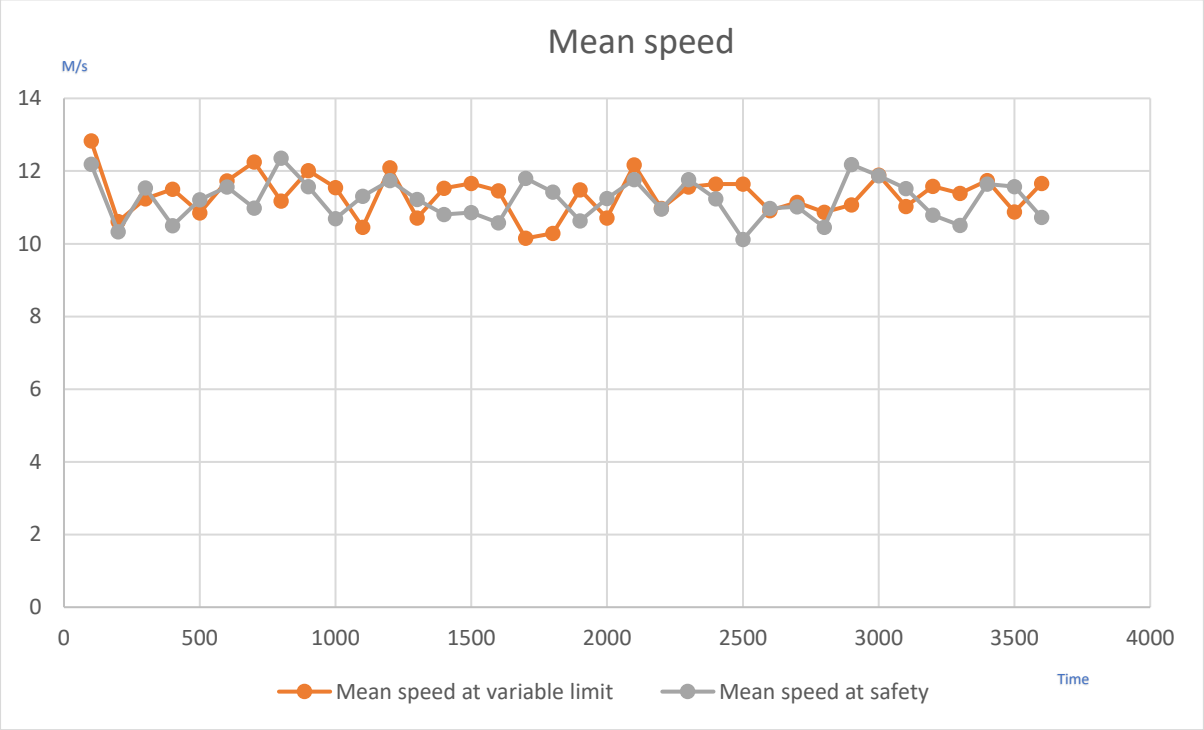


Figure 61 Mean speed

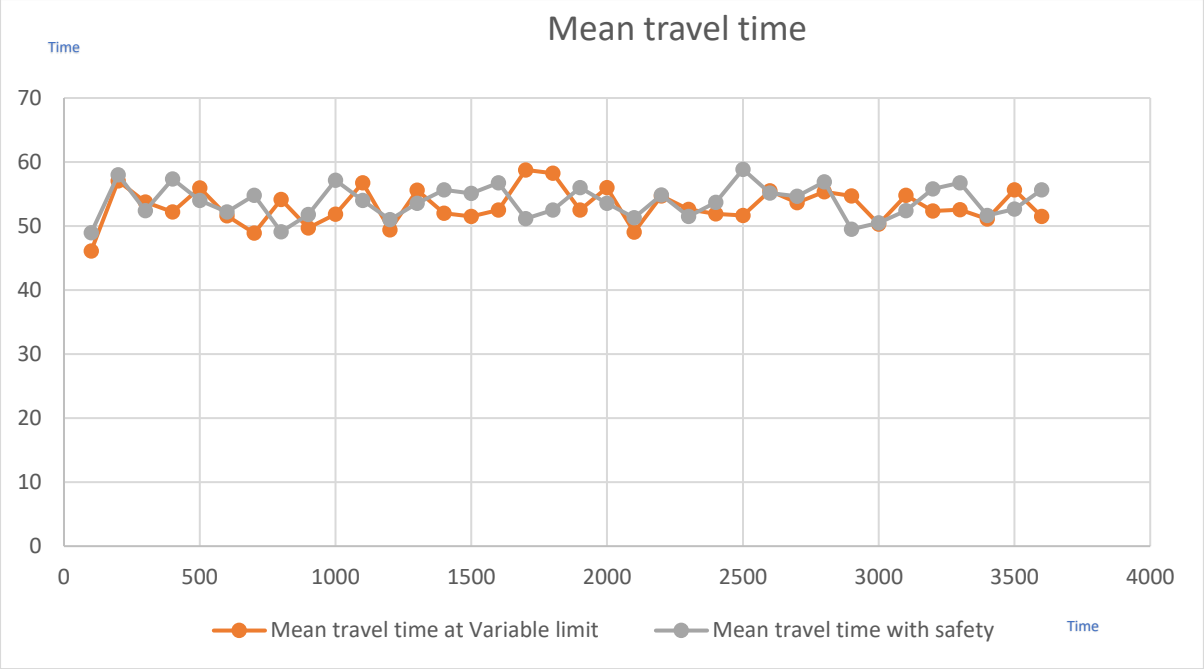


Figure 62 Mean travel time

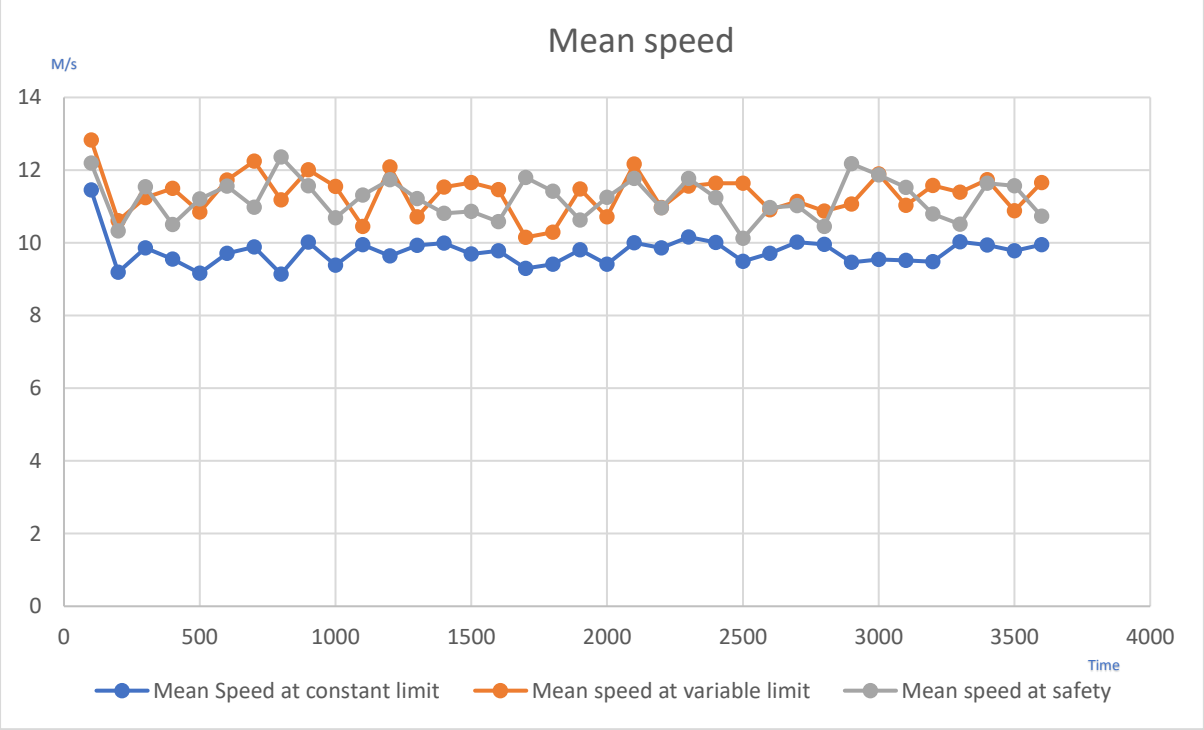


Figure 63 Comparison of mean speed between three schemes

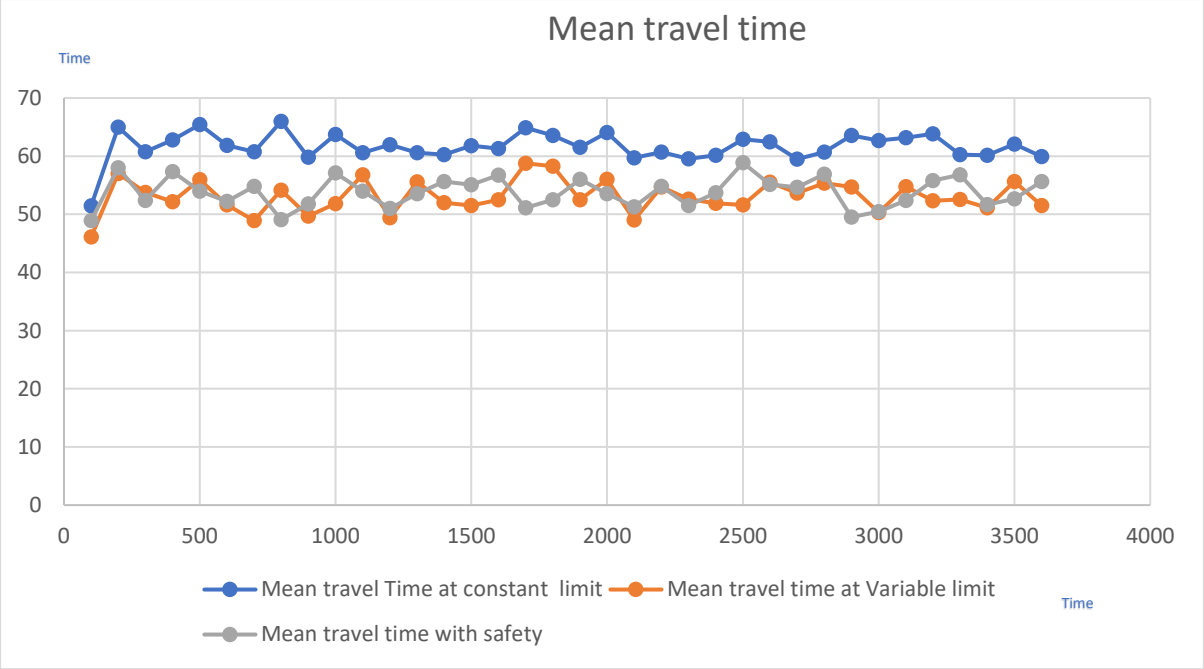


Figure 64 comparison of mean travel time between three schemes

The traffic scheme with additional safety phase did not show much difference so it is better to utilize this scheme.

KPI	Signal with constant speed limit	Signal with variable limit	Signal with variable limit and safety
Mean travel time(s)	61	53	54
Mean time loss(s)	20	18	19
Mean speed(m/s)	9.7	11.3	11.2
Mean co_2 emissions	171	157	160

Table 21

4.5 Adaptive signal based on the location of the vehicle

This adaptive signal changes the green time according to the location of the vehicle on the street if the infrastructure detects the vehicle at 400m from the intersection then the time green time is extended by 15 seconds so that the vehicle can pass the intersection freely. If there is no vehicle from 400m to 300m then the traffic signal control system checks whether there are any vehicle between 200m and 300m from the intersection. If there are any vehicles on the street the time is extended by 10 seconds. If there are no vehicle from 200m to 400m away from the intersection, then the system check whether there are any vehicle from 100m to 200m if there are any vehicle in this section the green time is extended by two seconds. Note that there are three sections in the green time phase, out of which the first and the third segments are fixed timings at which the maximum velocity on the street is maintained at 15m/s. The seconds segment is adaptive and the maximum velocity during this phase is increased to 20m/s. During the phase III if the timing of the adaptive phase is extended to 15 seconds the maximum velocity of the street was maintained at 20m/s otherwise it was reduced to 15m/s second.

Acceleration(a)=1m/s^2

Deceleration(d)=5m/s^2

Initial velocity(u)=15m/s

Final velocity(v)v=20m/s

Distance between the vehicle and the traffic signal=400m

Distance travelled during acceleration(S)= $\frac{V^2-u^2}{2a}$

S=87.5m

Distance covered during cruising=300m

Time required for the vehicle to cross the intersection from 400m is 21sec

Time required by the vehicle at 300m from the intersection is 17 sec

Time required by the vehicle at 200m from the intersection is 7 secs

Using the calculations mentioned above the lane area detectors were activated on the network from a distance of 200m to 400m from the intersection.

Fixed time signal			Adaptive signal based on vehicle's location		
segments	I	II	I	II	III
Control	Fixed	Fixed	Fixed	Actuated	Fixed
Maximum velocity(m/s)	15	15	15	20	15
Phase time	15	15	10	1-15	5

Table 22

In the table shown below one can see the three sections. The segments II and III are divivded into sections 1,2 and three. If the Number of vehicles on the lane is less than N_{min} the signal is turned to red. If the number of vehicles is greater than N_{min} the green time is extended while simultaneously checking the position of the vehicles on the lane. In the section 2 the traffic signal is extended according to the position of the vehicles. The maximum velocity during section 1 and seciton two is always 20m/s, where the maximum velocity during section 3 will be 20m/s if the section 2 is extended otherwise the maximum velocity will be reduced to 15m/s

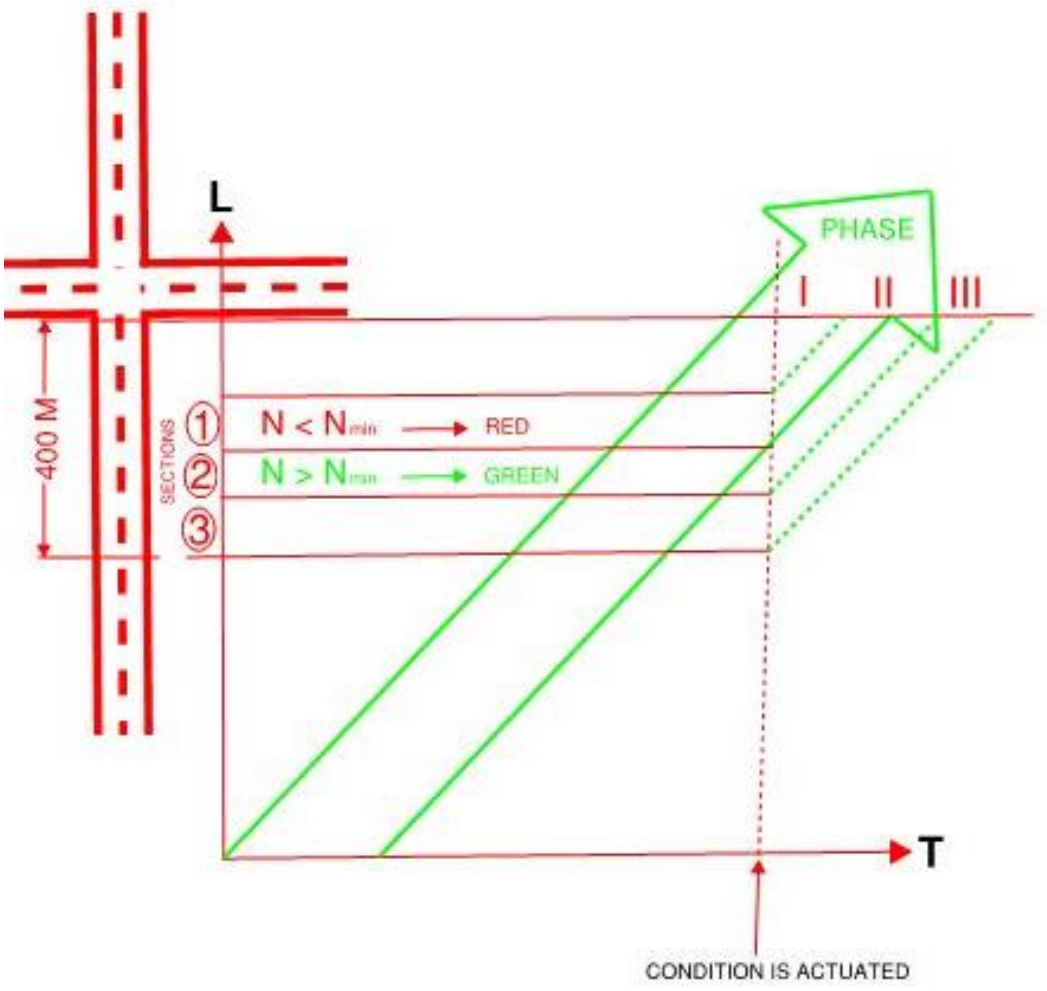
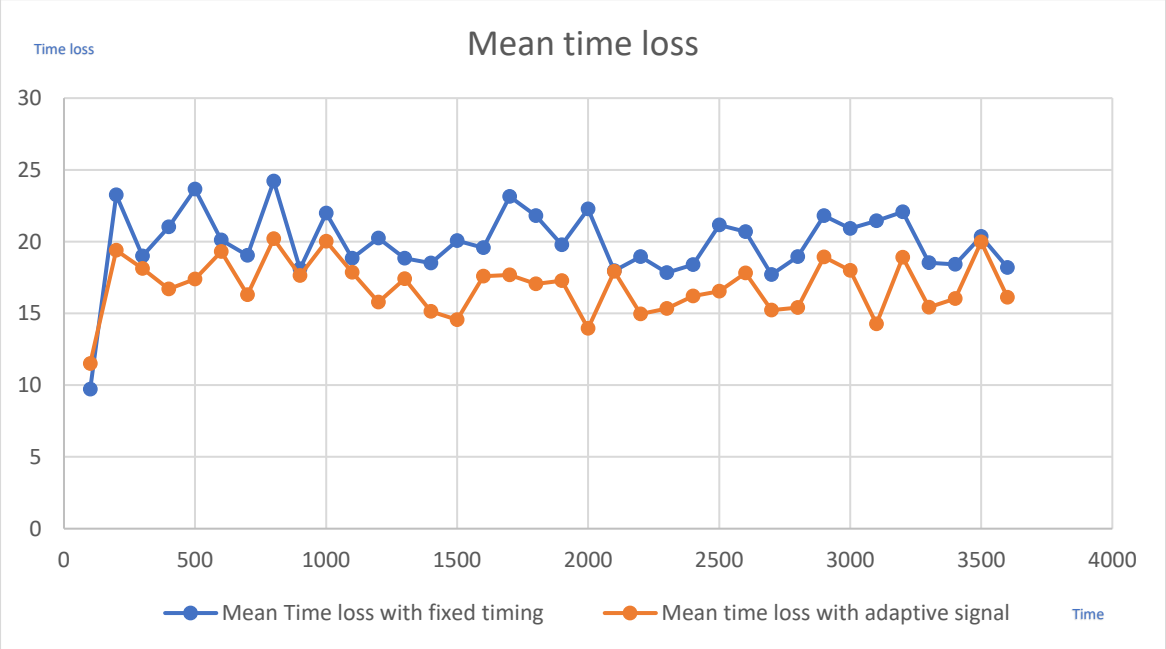
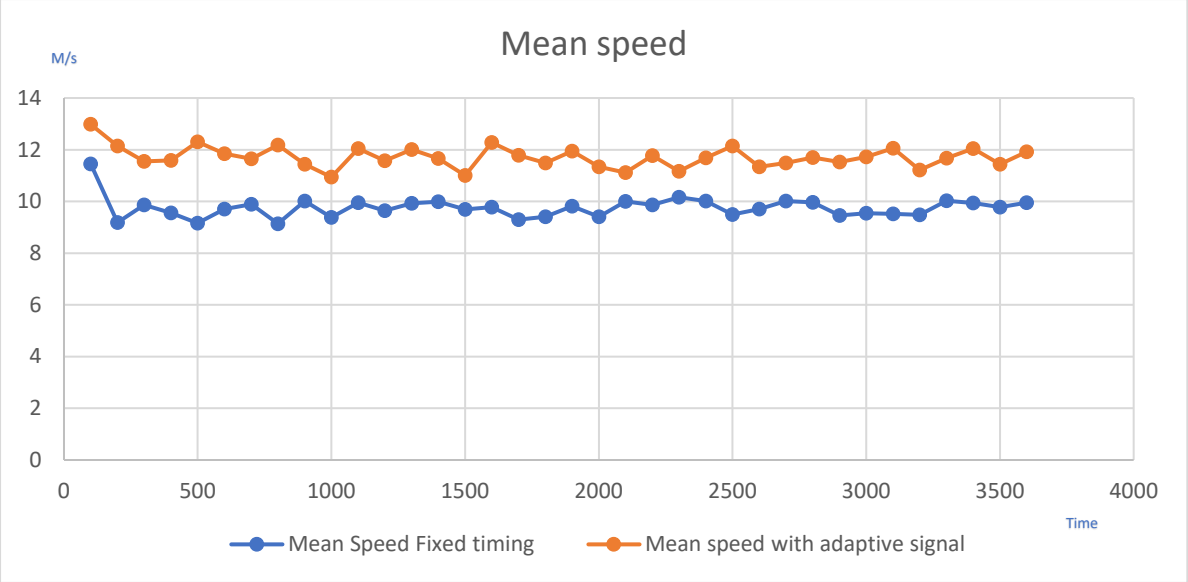
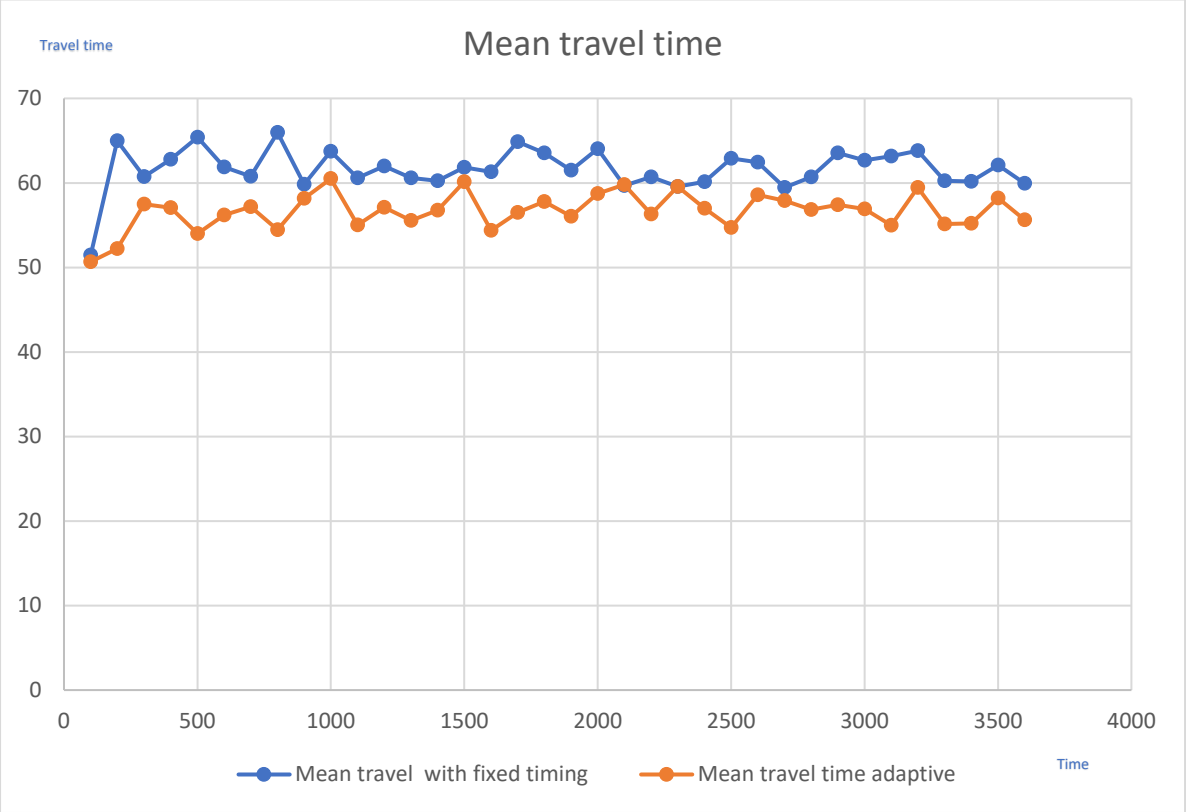


Figure 65:Adaptive traffic signal space time diagram







KPI	Static Signal	Adaptive signal with variable speed limit
Mean travel time(s)	61	56
Mean speed(m/s)	9.7	11.7
Mean time loss(s)	20	16
Mean Co_2 emissions(g)	171	163

Table 23

4.6 Application on viale dell'Oceano Pacifico

Figure 66:Network converted from the OSM map data

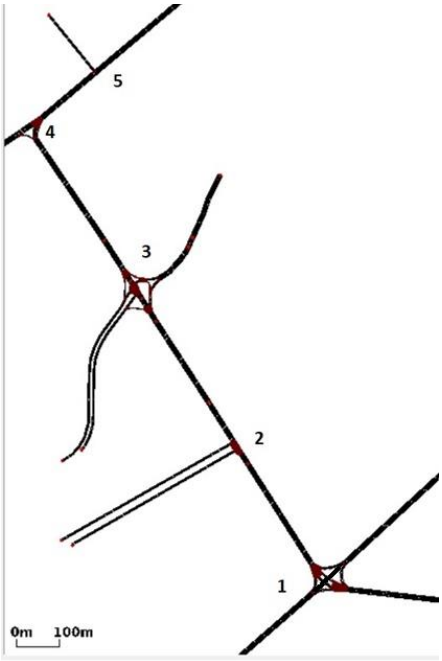


Figure 67: Filtered network from OSM data.

The network converted from OSM has many other street this causes the SUMO software to lag a little bit, so the network was filtered by deleting the other lanes manually in the NETEDIT. The result is a just the Viale dell'Oceano Pacifico with the three intersections. The Figure 61 shown below demonstrates the filtered network.

Number	Intersection	Distance(m)
1	Viale dell'Oceano Pacifico-Via Cristoforo Colombo	0
2	Viale dell'Oceano Pacifico-Viale Giorgio Ribotta	450
3	Viale dell'Oceano Pacifico- Viale della Grande Muraglia	900
4	Viale dell'Oceano Indiano- Viale Egeo	1400
5	Viale Egeo-Via monte dell Finocchio	1900

Table 24:Table showing the distance starting from intersetion1

Mean time loss by a vehicle is 30seconds. The time required to pass the five intersections is 110 seconds.

4.7 Issues found during simulation

The map imported from the OpenStreetMap is not accurate. SUMO considers a large intersection as two intersections. Signal imported from OSM to SUMO do not have proper cycle times. During the simulation of car following models such Pwagner and Wiedemann there were some collisions occurred. Upon referring to the SUMO site forum. The collisions occurred were due to some errors in the simulation software. While testing the python algorithms in SUMO. There were some complications with the logics in the python algorithm as SUMO did not flag any errors and signals were in a constant phase. So, care should be taken while writing the logic in python. There were some problems with multi entry exit detectors due some vehicles exiting the network on other arteries. So, care should be taken while placing the entry exit detectors and the researcher should make sure that there are no other arteries between the entry and exit detector. The traffic lights logic cannot be used as an additional file as this gave some error. The simulation by TraCI does not respect the time described in the configuration file. The user has to specify the detectors in an additional xml file with all the characteristics of the detector in it. Instant Induction loops cannot be used by the TraCI. During the simulation some cars were involved in some collision, the SUMO website describes that the SUMO software uses collision free model, but the presence of some bugs or intentional configurations. SUMO teleports the vehicles if the vehicle has waited too long or involved in a collision. The teleported vehicle will be inserted as soon as there is a place available on the network. A vehicle may be teleported multiple times during the simulation, however the vehicle maintains the average speed on the network during the teleportation. SUMO cannot provide the offset, so the user has to calculate the offset manually. NETEDIT always creates the junctions as a type of priority, this causes vehicles to make unnecessary stops at those junctions. One can overcome this issue by modifying the type of the junction to internal, so vehicles maintain their velocity while passing this intersection.The emissions values generated by the full ouput shows that the vehicles emit 0mg during idling time, but this is not true as idling engines do create pollution. Converting the network from OSM is quite easy as it uses juts the NETCONVERT tool but correcting this network would take a lot of time as an example a network with 1000 junctions takes less than a minute using the SUMO converting tools and after that correcting would require at least three days.

5.Conclusion and future work

In chapter two several car following models and lane changing models were tested in SUMO. Krauss car following model in SUMO has the most variations of it in SUMO. These studies may help others in selecting the right car following model in SUMO according to ones criteria. In chapter 5 the methodology used for traffic signal synchronization was described. The signal synchronization with adaptive traffic signal decreases a considerable amount of time loss on urban road arteries. Care should be taken while writing the algorithm in python as the while loop in the program. After studying and analysing several research articles on SUMO. I have concluded that many of the researchers do not share the inner working in building up the networks in SUMO and on TraCI in SUMO. This report would be an excellent guide to whoever starting to simulate in SUMO as this gives an idea on which car following model and what parameters can be used to get the desired result. This study suggests that the use of V2I communication to detect vehicles on the network and make necessary modifications to the traffic light according to the flow. There is a significant reduction in average waiting time, increase in average speed is being exposed by the simulation results. The simulation results of the scenario with six intersections show that there is an average 30% reduction in travel time. There is a 50% increase in mean speed and 60% decrease in time loss along through these intersections. There is a significant improvement in traffic conditions of urban arteries which utilized adaptive traffic signal with synchronization. Further research should be carried out to better understand this system in Python. There is a huge demand for traffic simulation now more than ever due to increased adaptation by various research institutions and government organizations. The simulations results obtained from these simulations can be compared with the real traffic to calibrate the parameters of the model. SUMO is a open source software and it’s uses are unlimited because this simulation software can be couple with any other software to perform the needs of the user. This other software include MATLAB, omnet++, Ns2, Unity3D, rFpro etc. Further research is required to integrate these methods into the existing traffic networks.

6.Figures

7.Tables

8.References

1. Self-organizing traffic signals using secondary extension and dynamic coordination - Burak Cesme, Peter G. Furth
2. A Design Procedure for Road Artery Signal Synchronization with Real-Time Bus Priority- Chiara Colombaroni, Gaetano Fusco, and Andrea Gemma
3. Simulation and Evaluation of a Public Transport Priority Methodology- Malandraki , Papamichail, Papageorgiou M, Dinopoulou V.b
4. Optimization Model of Transit Signal Priority Control for Intersection and Downstream Bus Stop - Rui Li , Changjiang Zheng and Wenquan Li
5. Transportation Planning Based on GSM Traces: A Case Study on Ivory Coast- Mirco Nanni , Roberto Trasarti, Barbara Furletti, Lorenzo Gabrielli, Peter Van Der Mede, Joost De Bruijn, Erik De Romph, and Gerard Bruil
6. Simulation of Transit Signal Priority Using the NTCIP Architecture Hongchao Liu
7. Planning and Deploying Transit Signal Priority in Small and Medium-Sized Cities: Burlington, Vermont, Case Study Kleoniki Vlachou
8. Development of model-based transit signal priority control for local arterials- Yongjie Lina , Xianfeng Yang , Lei Jia , and Nan Zou
9. Evaluation of Different Vehicle Following Models under Mixed Traffic Conditions Venkatesan Kanagaraj1 , Gowri Asaithambi, C. H. Naveen Kumar, Karthik K. Srinivasan, R. Sivanandan.
10. Preparing Data for Urban Traffic Simulation using SUMO Jos´e Capela Dias, Pedro Henriques Abreu, Daniel Castro Silva, Gustavo Fernades, Penousal Machado, and Ant´onio Leit˜ao A Comparative Study of Urban Road Traffic Simulators Mustapha Saidallah, Abdeslam El Fergougui and Abdelbaki Elbelrhiti Elalaoui
11. Impact of Transit Signal Priority on Level of Service at Signalized Intersections: Alexander Skabardonis, Eleni Christofa
12. Evaluating the Transit Signal Priority Impacts along the U.S. 1 Corridor in Northern Virginia: Vaibhavi Kamdar
13. Optimization of Traffic Signals on Urban Arteries through a Platoon Based Simulation Model Chiara Colombaroni, Gaetano Fusco, Andrea Gemma
14. Traffic Control Optimization for Multi-Modal Operations in a Large-Scale Urban Network Aleksandar Stevanovic*, Jelka Stevanovic, Cameron Kergaye, and Peter Martin
15. Evaluation of Different Vehicle Following Models under Mixed Traffic Conditions Venkatesan Kanagaraj1 , Gowri Asaithambi, C. H. Naveen Kumar, Karthik K. Srinivasan, R. Sivanandan.
16. Krauss, S.: Microscopic Modelling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics. Ph.D. Thesis, University of Cologne, Cologne, Germany (1997)
17. Development of model-based transit signal priority control for local arterials Yongjie Lina *, Xianfeng Yangb , Lei Jiaa , and Nan Zoua.

18. Recent Development and Applications of SUMO – Simulation of Urban MObility Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker.
19. http://www.SUMO.dlr.de/pydoc/TraCI.html
20. C. Diakaki, V. Dinopoulou, K. Aboudolas, M. Papageorgiou, E. Ben Shabat, E. Seider, and A. Leibov (2003). Extensions and new applications of the traffic-responsive urban control strategy: coordinated signal control for urban networks, Transportation Research Record, no. 1856, 202– 211.
21. Djordjevic, Boban & Krmac, Evelin. (2016). KEY PERFORMANCE INDICATORS FOR MEASURING THE IMPACTS OF ITS ON TRANSPORT.

Appendix

Type map

```
<polygonTypes>
   <polygonType id="waterway"              name="water"      color=".71,.82,.82" layer="-4"/>
   <polygonType id="natural"               name="natural"    color=".55,.77,.42" layer="-4"/>
   <polygonType id="natural.water"         name="water"      color=".71,.82,.82" layer="-4"/>
   <polygonType id="natural.wetland"       name="water"      color=".71,.82,.82" layer="-4"/>
   <polygonType id="natural.wood"          name="forest"     color=".55,.77,.42" layer="-4"/>
   <polygonType id="natural.land"          name="land"       color=".98,.87,.46" layer="-4"/>
   <polygonType id="landuse"               name="landuse"    color=".76,.76,.51" layer="-3"/>
   <polygonType id="landuse.forest"        name="forest"     color=".55,.77,.42" layer="-3"/>
   <polygonType id="landuse.park"          name="park"       color=".81,.96,.79" layer="-3"/>
   <polygonType id="landuse.residential"   name="residential" color=".92,.92,.89" layer="-3"/>
   <polygonType  id="landuse.commercial"   name="commercial"  color=".82,.82,.80" layer="-3"/>
   <polygonType id="landuse.industrial"    name="industrial"  color=".82,.82,.80" layer="-3"/>
   <polygonType id="landuse.military"      name="military"    color=".60,.60,.36" layer="-3"/>
   <polygonType id="landuse.farm"          name="farm"       color=".95,.95,.80" layer="-3"/>
   <polygonType id="landuse.greenfield"    name="farm"       color=".95,.95,.80" layer="-3"/>
   <polygonType id="landuse.village_green" name="farm"       color=".95,.95,.80" layer="-3"/>

   <polygonType id="tourism"               name="tourism"    color=".81,.96,.79" layer="-2"/>
   <polygonType id="military"              name="military"   color=".60,.60,.36" layer="-2"/>
   <polygonType id="sport"                 name="sport"      color=".31,.90,.49" layer="-2"/>
   <polygonType id="leisure"               name="leisure"    color=".81,.96,.79" layer="-2"/>
   <polygonType id="leisure.park"          name="tourism"    color=".81,.96,.79" layer="-2"/>
```

```
   <polygonType id="aeroway"             name="aeroway"     color=".50,.50,.50" layer="-2"/>
   <polygonType id="aerialway"           name="aerialway"   color=".20,.20,.20" layer="-2"/>
   <polygonType id="shop"                name="shop"        color=".93,.78,1.0" layer="-1"/>
   <polygonType id="historic"            name="historic"    color=".50,1.0,.50" layer="-1"/>
   <polygonType id="man_made"            name="building"    color="1.0,.90,.90" layer="-
1"/>
   <polygonType id="building"            name="building"    color="1.0,.90,.90" layer="-1"/>
   <polygonType id="amenity"             name="amenity"     color=".93,.78,.78" layer="-1"/>
   <polygonType id="amenity.parking"     name="parking"     color=".72,.72,.70" layer="-
1"/>
      <polygonType id="highway"          name="highway"     color=".10,.10,.10" layer="-
1" discard="true"/>


   <polygonType id="boundary" name="boundary"          color="1.0,.33,.33"  layer="0"
fill="false" discard="true"/>
   <polygonType id="admin_level" name="admin_level"    color="1.0,.33,.33"  layer="0"
fill="false" discard="true"/>
</polygonTypes>
```

Configuration

```
   <input>
      <net-file value="file.net.xml"/>
       <route-files value="file.rou.xml"/>
   </input>
   <time>
      <begin value="0"/>
      <end value="3600"/>
   </time>


   <output>
      <queue-output value="output.xml"/>
   </output>
/configuration>
```

OSM

Then the user has to navigate to the directory where the network file is downloaded. Then the following command netconvert is used.

The syntax for the following command is shown below

Netconvert --osm-files file.osm -o file.net.xml

In the above syntax the file.osm is converted to file.net.xml.The next step is to use a function known as polyconvert which imports geometrical shapes so that it can be viewed by the SUMO-gui. In this step we import polygons and their types of the structures in the map based on their color.

The above polygon types has to imported into a notemap++ file and to be saved in .xml format as file.xml.

Then the foll/owing syntax can be used to convert

Polyconvert --net-file file.net.xml --osm-files file.osm --type-file typemap.xml -o file.poly.xml

This created the polygon file which can be used to visualize infrastructure and habitation in the network

Python c:/SUMO/map/randomtrips.py -n file.net.xml -e 100 1

The simulation generated 1 to 100 trips.

Python c:/SUMO/map/randomtrips.py -n file.net.xml -r file.rou.xml -e 100 1

Then the above syntax gives an output file of file.poly.xml. There is file in the tools folder of the SUMO directory where you can find the randomtrips.py which generated random trips for our simulation. This file can be copied into the directory where the file.osm is located then the following command has to be used

Traffic light

The traffic signal logic for the simple node is given below.

```
<tlLogic id="gneJ1" type="static" programID="0" offset="0">
    <phase duration="48" state="GGrrGGrr"/>
    <phase duration="4" state="yyrryyrr"/>
    <phase duration="48" state="rrGGrrGG"/>
    <phase duration="4" state="rryyrryy"/>
</tlLogic>
```

The route file for the lane changing mode LC2013l is described below

<routes>

<vType accel="1.5" decel="3.5" id="Car" length="4.0" mingap="3" emergencyDecel="9.5" maxSpeed="13.0" tau="1" lcKeepRight="1" lcCooperative="2" lcStrategic="0.5" laneChangeModel="LC2013"/>

<flow id="type1" color="1,1,0" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 "/>

</flow>

<flow id="type2" color="1,1,0" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16"/>

</flow>

<flow id="type3" color="0,1,0" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE15 gneE16 gneE17 gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17"/>

</flow>

<flow id="type4" color="1,0,1" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE16 gneE17 gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17 gneE7"/>

</flow>

<flow id="type5" color="1,1,0" begin="0" end= "36060" vehsPerHour="100" type="Car">

<route edges="gneE17 gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 "/>

</flow>

<flow id="type6" color="1,0,0" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE7 gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 gneE9"/>

</flow>

<flow id="type7" color="1,0.5,0.5" begin="0" end= "3600" vehsPerHour="100" type="Car">

<route edges="gneE8 gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 gneE9 gneE11"/>

</flow>

<flow id="type8" color="0.5,1,0" begin="0" end= "36000" vehsPerHour="100" type="Car">

<route edges="gneE9 gneE11 gneE12 gneE13 gneE14 gneE15 gneE16 gneE17 gneE7 gneE8 gneE9 gneE11 gneE12"/>

</flow>

</routes>

Intersection1

The demand for the intersection 1 is given in SUMO as observed from the field visits. The code below describes the parameters and the demand of the traffic flow.

<routes>

<vType accel="1.0" decel="5.0" id="Car" length="3.0" maxSpeed="13.0" sigma="0.5" />

<flow id="type1" color="1,1,0" begin="0" end= "7200" vehsPerHour="318" type="Car">

<route edges="gneE14 -119306556#2"/>

</flow>

<flow id="type2" color="1,1,0" begin="0" end= "7200" vehsPerHour="152" type="Car">

<route edges="gneE14 335286725#0"/>

</flow>

<flow id="type3" color="1,1,0" begin="0" end= "7200" vehsPerHour="73" type="Car">

<route edges="-335286725#0 gneE8"/>

</flow>

<flow id="type4" color="1,1,0" begin="0" end= "7200" vehsPerHour="330" type="Car">

<route edges="-335286725#0 -119306556#2"/>

</flow>

<flow id="type5" color="1,1,0" begin="0" end= "7200" vehsPerHour="170" type="Car">

<route edges="-335286725#0 -119306556#2"/>

</flow>

<flow id="type6" color="1,1,0" begin="0" end= "7200" vehsPerHour="247" type="Car">

<route edges="119306556#2 gneE8"/>

</flow>

<flow id="type7" color="1,1,0" begin="0" end= "7200" vehsPerHour="352" type="Car">

<route edges="119306556#2 335286725#0"/>

</flow>

<flow id="type8" color="1,1,0" begin="0" end= "7200" vehsPerHour="601" type="Car">

```
    <route edges="119306556#2 335286725#0"/>
    </flow>
</routes>
```

The traffic signal logic written in the SUMO is described below. The phases and their timings imitate the traffic signal observed

```
    <tlLogic id="Viale_oceano_pacifico-viale_Avignone" type="static" programID="0" offset="0">
        <phase duration="48" state="GGrrrrrGGG"/>
        <phase duration="4" state="YYrrrrryyy"/>
        <phase duration="68" state="GGGGgrrrrr"/>
        <phase duration="4" state="yyyyyrrrrr"/>
        <phase duration="24" state="rrrGGGGrrr"/>
        <phase duration="4" state="rrryyyyrrr"/>
    </tlLogic>
```

Output of Induciton loop detector

A	B	C	D	E	F	G	H	I
begin	end	id	nVehContrib	flow	occupancy	speed	length	nVehEntered
0	100	e1Detector_gneE15_0_0	12	432	3.82	12.56	4	12
100	200	e1Detector_gneE15_0_0	26	936	8.26	12.59	4	26
200	300	e1Detector_gneE15_0_0	37	1332	12.1	12.32	4	38
300	400	e1Detector_gneE15_0_0	49	1764	17.41	11.44	4	49
400	500	e1Detector_gneE15_0_0	50	1800	22.85	9.08	4	50
500	600	e1Detector_gneE15_0_0	56	2016	18.25	12.35	4	56
600	700	e1Detector_gneE15_0_0	50	1800	15.93	12.57	4	51
700	800	e1Detector_gneE15_0_0	44	1584	14.79	12.22	4	46
800	900	e1Detector_gneE15_0_0	45	1620	18.29	10.12	4	44
900	1000	e1Detector_gneE15_0_0	48	1728	23.55	8.44	4	48
1000	1100	e1Detector_gneE15_0_0	58	2088	19.83	11.78	4	58
1100	1200	e1Detector_gneE15_0_0	59	2124	19.55	12.1	4	59
1200	1300	e1Detector_gneE15_0_0	57	2052	18.51	12.33	4	57
1300	1400	e1Detector_gneE15_0_0	44	1584	16.69	10.97	4	46
1400	1500	e1Detector_gneE15_0_0	53	1908	16.96	12.51	4	53
1500	1600	e1Detector_gneE15_0_0	46	1656	17.75	10.79	4	47
1600	1700	e1Detector_gneE15_0_0	51	1836	16.21	12.59	4	51
1700	1800	e1Detector_gneE15_0_0	51	1836	16.9	12.16	4	51
1800	1900	e1Detector_gneE15_0_0	48	1728	17.64	11.33	4	49
1900	2000	e1Detector_gneE15_0_0	55	1980	17.2	12.56	4	54
2000	2100	e1Detector_gneE15_0_0	47	1692	17.73	10.9	4	47
2100	2200	e1Detector_gneE15_0_0	50	1800	20.27	10.35	4	50
2200	2300	e1Detector_gneE15_0_0	52	1872	19.06	11.43	4	53
2300	2400	e1Detector_gneE15_0_0	51	1836	19.52	11.08	4	51
2400	2500	e1Detector_gneE15_0_0	49	1764	18.7	10.74	4	48
2500	2600	e1Detector_gneE15_0_0	54	1944	18.57	11.88	4	55
2600	2700	e1Detector_gneE15_0_0	57	2052	18.1	12.6	4	57

Edge

There are various functions in this class. All these functions give the information about the road and the vehicles on the road.

1.adaptTraveltime(self, edgeID, time, begin=None, end=None)

This function is used for re routing for the given edge.It is necessary to specify the begin and end of this time so this function will be applied only in that time otherwise the function will apply all the time.

Syntax:

TraCI.edge.adaptTraveltime(edgeID, begin=None, end=None)

2.getAdaptedTraveltime(self, edgeID, time)

This function gives the travel time in seconds used on an edge at a given time used for rerouting in double.
Syntax:

TraCI.edge.getAdaptedTraveltime()

3. getLaneNumber(self, edgeID)
This function returns the number of lanes of this edge as an integer.
Syntax:

TraCI.edge.getLaneNumber()

4. getLastStepHaltingNumber(self, edgeID)
This function returns the number of halting vehicle on given edge in the last time step as an integer.
Halt is a speed less than 0.1 m/s.
Syntax:
TraCI.edge.getLastStepHaltingNumber("edgeID")

5. getLastStepMeanSpeed(self, edgeID)
This function returns the average speed of the vehicles in m/s on a given edge in the last time step as double.
Syntax:
TraCI.edge.getLaststepMeanSpeed("edgeID")

6. getLastStepOccupancy(self, edgeID)

yields the occupancy in percentage for the last time step on the given edge as a double.
Syntax:
TraCI.edge.getLastStepOccupancy("edgeID")

7. getLastStepVehicleNumber(self, edgeID)

Returns the total number of vehicles for the last time step on the given edge.
Syntax:
TraCI.edge.getLastStepVehicleNumber()
8. getWaitingTime(self, edgeID)
This command gives the sum of the waiting time of all vehicles currently on the prescribed edge as a double value.
Syntax:
TraCI.edge.getWaitingTime("edgeID")

Inductionloop

1.getLaneID(self, loopID)

This function can be used to the ID of the lane on which the induction loop detector is placed upon.

Syntax:

TraCI.inductionloop.getLaneID(“LaneID”)

2.getLastStepMeanSpeed(self, loopID)

This returns the value of mean speed of m/s of vehicles on the induction loop detector in the last time step.

Syntax:

TraCI.inductionloop.getLastStepMeanSpeed()

3. getLastStepVehicleNumber(self, loopID)

This function returns the number of vehicles that were on the specified induction loop detector in the last time step.

4. getTimeSinceDetection(self, loopID)

This function returns the time in seconds since last detection.

Syntax:

TraCI.inductionloop.getTimeSinceDetection(“loopID”)

5. getPosition(self, loopID)

This command returns the measured position from beginning of the lane in meters.

Syntax:

TraCI.inductionloop.getPosition(“loopID”)

The above described commands can used also for the lane area detectors except the following command.

Lane

1.getLength(self, laneID)

Gives the length of the lane area detector in double

Syntax:

TraCI.lane.getLength(“laneID”)

2.. getMaxSpeed(self, laneID)

getMaxSpeed(string)
This command returns a value in double the maximum allowed speed on the lane in m/s.
Syntax:
TraCI.lane.getMaxSpeed(“laneID”)

3.setMaxSpeed(self, laneID, speed)

setMaxSpeed(string, double)
This command can be used to set a fresh maximum speed on the specified lane
Syntax:
TraCI.lane.setMaxSpeed("laneID")
4.getEdgeID(self, laneID)
getEdgeID(string)
This command returns the id of the edge that the given lane belongs to.
Syntax:
TraCI.lane.gerEdge("laneID")

Simulation

1. getArrivedNumber(self)

getArrivedNumber() -> integer

Returns the number of vehicles which arrived (have reached their destination and are remove d from the road network) in this time step.

2. getCurrentTime(self)

getCurrentTime() -> integer

Returns the current simulation time in ms. Hence care should be taken while writing the python program as the simulation time is displayed in seconds.

3. getDeltaT(self)
getDeltaT()
Returns the length of one simulation step in milliseconds

4. getDepartedNumber(self)
getDepartedNumber()
Returns the number of vehicles which departed (were inserted into the road network) in this ti me step.

5.getMinExpectedNumber(self)
getMinExpectedNumber() -> integer

Returns the number of vehicles which are in the net plus theones still waiting to start. This nu mber may be smaller thanthe actual number of vehicles still to come because of delayedroute file parsing. If the number is 0 however, it is
guaranteed that all route files have been parsed completely
and all vehicles have left the network.

Traffic Light

1. getCompleteRedYellowGreenDefinition(self, tlsID)

This function can be used to set the set the traffcc definitions in SUMO

2. getControlledLanes(self, tlsID)
getControlledLanes(string) -> c

This function gives the list of all the lanes controlled by traffic light.

3. getPhase(self, tlsID)
getPhase(string)
Returns the phase of the traffic signal in integer form.

4. getPhaseDuration(self, tlsID)
getPhaseDuration(string) -> integer

Retuns the duration of the current phase in integer.

5. getRedYellowGreenState(self, tlsID)
getRedYellowGreenState(string) -> string

This function return the state of the traffic light in the following format as Ggyyrro, here the lower case letters mean that the vehicles approaching the intersection of that lane has to decelerate. The image below shows the possible states of the traffic light those can be described in SUMO.

Syntax:
TraCI.trafficlights.getRedYellowGreenState("tlsID")

6.setCompleteRedYellowGreenDefinition(self, tlsID, tls)

setCompleteRedYellowGreenDefinition(string,) -> None

7.setLinkState(self, tlsID, tlsLinkIndex, state)

setLinkState(string, string, int, string) -> None
Sets the state for the given tls and link index. The state must be one
of rRgGyYoOu for red, red-
yellow, green, yellow, off, where lower case letters mean that the stream has to decelerate.
The link index is shown the gui when setting the appropriate junctino
visualization optin.

8.setPhase(self, tlsID, index)

setPhase(string, integer) -> None

.

9.setPhaseDuration(self, tlsID, phaseDuration)

setPhaseDuration(string, integer or float) -> None

Set the phase duration of the current phase in seconds.

10.setProgram(self, tlsID, programID)

setProgram(string, string) -> None

Sets the id of the current program.

11.setRedYellowGreenState(self, tlsID, state)

setRedYellowGreenState(string, string) -> None

Sets the named tl's state as a tuple of light definitions from
rugGyYuoO, for red, red-
yellow, green, yellow, off, where lower case letters mean that the stream has to decelerate.

Vehicle

1. getWaitingTime(self, vehID)

getWaitingTime() -> double
The waiting time of a vehicle is defined as the time (in seconds) spent with a
speed below 0.1m/s since the last time it was faster than 0.1m/s.
(basically, the waiting time of a vehicle is reset to 0 every time it moves).
A vehicle that is stopping intentionally with a <stop> does not accumulate waiting time.

2. getSpeed(self, vehID)

getSpeed(string)
The speed of the vehicle in the last will be returned by this function

3.setDecel(self, vehID, decel)

setDecel(string, double) -> None

This command is used to set the maximum deceleration of the vehicle in m/s^2.

4. setMaxSpeed(self, vehID, speed)

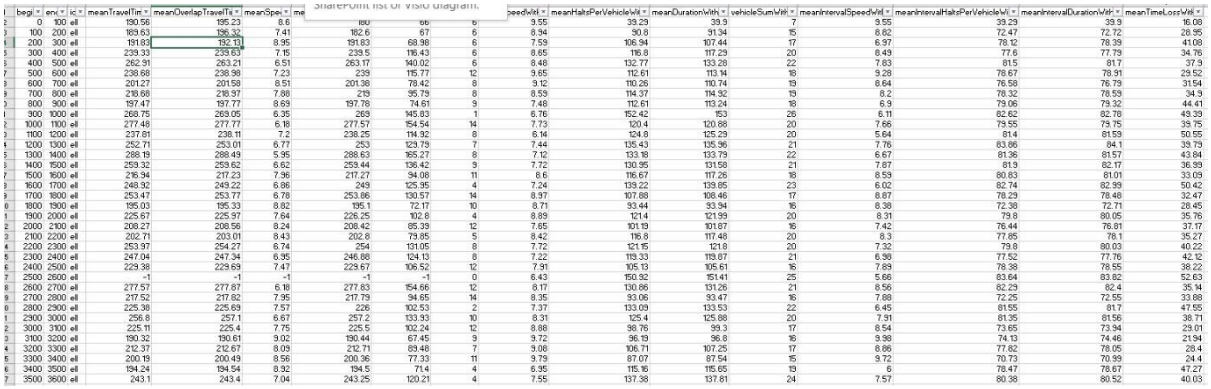
setMaxSpeed(string, double) -> None

This instruction sets the maximum speed in m/s for the specified vehicle.The value is taken as a double.

Syntax:

TraCI.vehicle.setMaxSpeed(“vehID”, speed)

Output of multi entry-exit detector



	begi	en	ic	meanTravelTim	meanOverlapTravelTi	meanSpe	me			speedWit	meanHaltsPerVehicleWi	meanDurationWith	vehicleSumWit	meanIntervalSpeedWit	meanIntervalHaltsPerVehicleWi	meanIntervalDurationWit	meanTimeLossWit
2	0	100	e8	190.56	195.23	8.6	180	66	6	9.55	39.29	39.9	7	9.55	39.29	39.9	16.08
3	100	200	e8	189.63	136.32	7.41	182.6	67	6	8.94	30.8	91.34	15	8.82	72.47	72.72	28.95
4	200	300	e8	191.83	192.13	8.95	191.83	68.98	6	7.59	106.94	107.44	17	6.97	78.12	78.39	41.08
5	300	400	e8	239.33	239.63	7.15	239.5	116.43	6	8.65	116.8	117.29	20	8.49	77.6	77.79	34.76
6	400	500	e8	262.91	263.21	6.51	263.17	140.02	6	8.48	132.77	133.28	22	7.83	81.5	81.7	37.9
7	500	600	e8	238.68	238.98	7.23	239	115.77	12	9.65	112.61	113.14	18	9.28	78.67	78.91	29.52
8	600	700	e8	201.27	201.58	8.51	201.38	78.42	8	9.12	110.26	110.74	19	8.64	76.58	76.79	31.54
9	700	800	e8	218.68	218.97	7.88	219	95.79	8	8.55	114.37	114.92	19	8.2	78.32	78.59	34.9
10	800	900	e8	197.47	197.77	8.83	197.76	74.61	9	7.48	112.61	113.24	18	6.9	79.06	79.32	44.41
11	900	1000	e8	268.75	269.05	6.35	269	145.83	1	6.76	152.42	153	26	6.11	82.62	82.76	43.39
12	1000	1100	e8	277.48	277.77	6.18	277.57	154.54	14	7.73	120.4	120.88	20	7.66	79.55	79.75	33.75
13	1100	1200	e8	237.81	238.11	7.2	238.25	114.92	8	6.14	124.8	125.29	20	5.64	81.4	81.59	50.55
14	1200	1300	e8	252.71	253.01	6.77	253	129.79	7	7.44	135.43	135.96	21	7.76	83.86	84.1	33.79
15	1300	1400	e8	288.19	288.49	5.95	288.63	165.27	8	7.12	133.18	133.79	22	6.67	81.36	81.57	43.84
16	1400	1500	e8	259.32	259.62	6.62	259.44	136.42	9	7.72	130.95	131.58	21	7.87	81.9	82.17	36.99
17	1500	1600	e8	216.94	217.23	7.36	217.27	94.08	11	8.6	116.67	117.26	18	8.59	80.83	81.01	33.09
18	1600	1700	e8	248.92	249.22	6.86	249	125.95	4	7.24	139.22	139.85	23	6.02	82.74	82.99	50.42
19	1700	1800	e8	253.47	253.77	6.78	253.86	130.57	14	8.97	107.88	108.46	17	8.87	78.29	78.48	32.47
20	1800	1900	e8	195.03	195.33	8.82	195.1	72.17	10	8.71	93.44	93.94	16	8.38	72.38	72.71	28.45
21	1900	2000	e8	225.67	225.97	7.64	226.25	102.8	4	8.89	121.4	121.99	20	8.31	79.8	80.05	35.76
22	2000	2100	e8	208.27	208.56	8.24	208.42	85.39	12	7.65	101.19	101.87	16	7.42	76.44	76.81	37.17
23	2100	2200	e8	202.71	203.01	8.43	202.8	79.85	5	8.42	116.8	117.48	20	8.3	77.85	78.1	35.27
24	2200	2300	e8	253.97	254.27	6.74	254	131.05	8	7.72	121.15	121.8	20	7.32	79.9	80.03	40.22
25	2300	2400	e8	247.04	247.34	6.35	246.88	124.13	8	7.22	119.33	119.87	21	6.38	77.52	77.76	42.12
26	2400	2500	e8	229.38	229.69	7.47	229.67	106.52	12	7.91	105.13	105.61	16	7.89	78.38	78.55	38.22
27	2500	2600	e8	-1	-1	-1	-1	-1	0	6.43	150.92	151.41	25	5.66	83.64	83.82	52.63
28	2600	2700	e8	277.57	277.87	6.18	277.83	154.66	12	8.17	130.86	131.26	21	8.56	82.23	82.4	35.14
29	2700	2800	e8	217.52	217.82	7.95	217.79	94.65	14	8.35	93.06	93.47	16	7.88	72.25	72.55	33.88
30	2800	2900	e8	225.38	225.69	7.57	226	102.53	2	7.37	133.09	133.53	22	6.45	81.55	81.7	47.55
31	2900	3000	e8	256.8	257.1	6.67	257.2	133.93	10	8.31	125.4	125.88	20	7.91	81.35	81.56	38.71
32	3000	3100	e8	225.11	225.4	7.75	225.5	102.24	12	8.88	98.76	99.3	17	8.54	73.65	73.94	29.01
33	3100	3200	e8	190.32	190.61	9.02	190.44	67.45	9	9.72	96.19	96.8	16	9.98	74.13	74.46	21.94
34	3200	3300	e8	212.37	212.67	8.09	212.71	89.46	7	9.08	106.71	107.25	17	8.86	77.82	78.05	28.4
35	3300	3400	e8	200.19	200.49	8.56	200.36	77.33	11	9.79	87.07	87.54	15	9.72	70.73	70.89	24.4
36	3400	3500	e8	194.24	194.54	8.92	194.5	71.4	4	6.95	115.16	115.65	19	6	78.47	78.67	47.27
37	3500	3600	e8	243.1	243.4	7.04	243.25	120.21	4	7.55	137.38	137.81	24	7.57	80.38	80.52	40.03

Python script for SUMO

```
from __future__ import absolute_import
from __future__ import print_function

import os
import sys
import optparse
import subprocess
import random

try:
    sys.path.append(os.path.join(os.path.dirname(
        __file__), '..', '..', '..', '..', "tools"))
    sys.path.append(os.path.join(os.environ.get("SUMO_HOME", os.path.join(
        os.path.dirname(__file__), "..", "..", "..")), "tools"))
    from sumolib import checkBinary
except ImportError:
    sys.exit(
        "please declare environment variable 'SUMO_HOME' as the root directory of your sumo installation

import traci


def run():
    """execute the TraCI control loop"""
    step = 0

    traci.trafficlights.setPhase("gneJ1", 2)
    while traci.simulation.getMinExpectedNumber() > 0:
        traci.simulationStep()
        if traci.trafficlights.getPhase("gneJ1") == 2:

            if traci.inductionloop.getLastStepVehicleNumber("0") > 0:

                traci.trafficlights.setPhase("gneJ1", 3)
            else:

                traci.trafficlights.setPhase("gneJ1", 2)
        step += 1
    traci.close()
    sys.stdout.flush()


def get_options():
    optParser = optparse.OptionParser()
    optParser.add_option("--nogui", action="store_true",
                         default=False, help="run the commandline version of sumo")
    options, args = optParser.parse_args()
    return options

if __name__ == "__main__":
    options = get_options()
    if options.nogui:
        sumoBinary = checkBinary('sumo')
    else:
        sumoBinary = checkBinary('sumo-gui')


    traci.start([sumoBinary, "-c", "test.sumo.cfg",
                             "--tripinfo-output", "tripinfo.xml"])
    run()
```

Adaptive traffic signal detecting the density of vehicles on the edge

```
import traci
# No.of phases can be described here
MAINGREEN = "GrGr"
MAINYELLOW ="YrYr"
SIDEGREEN ="rGrG"
SIDEYELLOW ="ryry"
if not traci.isEmbedded():
        N = 1000
        pWE = 1./10
        pEW = 1./15
        pNS = 1./20
        pSN = 1./25
        routes = open("dy.rou.xml","w")
        log = open("log.txt", "w")
        print >> routes, """<routes>
        <vType id="Main" accel="1" decel="4.5" sigma="0.5" length="4" minGap="10" maxSpeed="13"/>
        <vType id="Side" accel="1" decel="4.5" sigma="0.5" length="4" minGap="10" maxSpeed="13"/>
        <route id="NB" edges="-gneE1 -gneE2" />
        <route id="SB" edges="gneE2 gneE1" />
        <route id="EB" edges="-gneE3 -gneE0" />
        <route id="WB" edges="gneE0 gneE3" />"""
        lastVeh = 0
        vehNr = 0
        for i in range(N):
             if random.uniform(0,1) < pWE:
                  print >> routes, '  <vehicle id="%i" type="Main" route="NB" depart="%i" />' % (vehNr, :
                  vehNr +=1
                  lastVeh=i
             if random.uniform(0,1) < pEW:
                  print >> routes, '  <vehicle id="%i" type="Main" route="SB" depart="%i" />' % (vehNr,i)
                  vehNr += 1
                  lastVeh = i
             if random.uniform(0,1) < pNS:
                  print >> routes, '  <vehicle id="%i" type="Side" route="EB" depart="%i" />' % (vehNr, :
                  vehNr += 1
                  lastVeh = i
             if random.uniform(0,1) < pSN:
                  print >> routes, '  <vehicle id="%i" type="Side" route="WB" depart="%i" />' % (vehNr, :
                  vehNr += 1
                  lastVeh = i
        print >> routes, "</routes>"
        routes.close()

        sumoBinary = 'sumo-gui'
        sumoConfig = "test.sumo.cfg"
        if len(sys.argv) > 1:
           retcode = subprocess.call("%s -c %s --python-script %s" % (sumoBinary, sumoConfig,  _file_), :
           sys.exit(retCode)
        else:
           sumoProcess = subprocess.Popen("%s -c %s" % (sumoBinary, sumoConfig), shell=True, stdout=sys.:
step = 0
max_d = 0
i=0
edge_density = {}
edge_Ids = traci.edge.getIDList()
traffic_lights=traci.trafficlights.getIDList()
traci.trafficlights.setRedYellowGreenState("gneJ1","rrrr")
while step == 0 or traci.simulation.getMinExpectedNumber()>0:
    traci.simulationStep()
    for i in xrange(15):
        vehicle_Ids=traci.edge.getLastStepVehicleIDs(edge_Ids[i])
        for vehicle in vehicle_Ids:
                time=traci.simulation.getCurrentTime()
                road=traci.vehicle.getRoadID(vehicle)
                typeID=traci.vehicle.gerTypeID(vehicle)
                speed=traci.vehicle.getMAxSpeed(vehicle)
                accln=traci.vehicle.getAccel(vehicle)
                decel=Traci.vehicle.gerDecel(vehicle)
        d_1=traci.edge.gerLastStepVehicleNumber("gneE0")
        d_2=traci.edge.gerLastStepVehicleNumber("gneE1")
        d_3=traci.edge.gerLastStepVehicleNumber("gneE2")
        d_4=traci.edge.gerLastStepVehicleNumber("gneE3")
        edge_density.append(d_1)
        edge_density.append(d_2)
        edge_density.append(d_3)
        edge_density.append(d_4)
        max_d = max(edge_density)

    if(max_d == d_1 or max_d == d_2):
        traci.trafficlights.setRedYellowGreenState("gneJ1", "yGyG")
    else:
        if(max_d == d_3 or max_d == d_4):
                traci.trafficlights.setRedYellowGreenState("gneJ1","GyGy")
    step += 1
traci.close()
sys.stdout.flush()
```

Adaptive signal based on the location of the vehicle

```
while traci.simulation.getMinExpectedNumber() > 0:
      traci.simulationStep()
      if traci.trafficlights.getPhase("gneJ12") == 0:
          traci.lane.setMaxSpeed("gneE5_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)
          traci.lane.setMaxSpeed("gneE8_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)
          x=traci.simulation.getCurrentTime()
          y=x+3000


      if traci.trafficlights.getPhase("gneJ12")==1:

          traci.lane.setMaxSpeed("gneE5_0",20)
          traci.lane.setMaxSpeed("-gneE5_0",20)
          traci.lane.setMaxSpeed("gneE8_0",20)
          traci.lane.setMaxSpeed("-gneE5_0",20)
          if traci.simulation.getCurrentTime()<y:
              if traci.lanearea.getLastStepVehicleNumber("e2Detector_gneE5_0_3")>0:
                  traci.trafficlights.setPhaseDuration("gneJ12",15)
              elif traci.lanearea.getLastStepVehicleNumber("e2Detector_gneE5_0_2")>0:
                  traci.trafficlights.setPhaseDuration("gneJ12",10)
              elif traci.lanearea.getLastStepVehicleNumber("e2Detector_gneE5_0_1")>0:
                  traci.trafficlights.setPhaseDuration("gneJ12",5)
      if traci.trafficlights.getPhase("gneJ12")==2:
          traci.lane.setMaxSpeed("gneE5_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",20)
          traci.lane.setMaxSpeed("gneE8_0",20)
          traci.lane.setMaxSpeed("-gneE5_0",15)
      if traci.trafficlights.getPhase("gneJ12")==3:
          traci.lane.setMaxSpeed("gneE5_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)
          traci.lane.setMaxSpeed("gneE8_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)

      if traci.trafficlights.getPhase("gneJ12")==4:
          traci.lane.setMaxSpeed("gneE5_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)
          traci.lane.setMaxSpeed("gneE8_0",15)
          traci.lane.setMaxSpeed("-gneE5_0",15)
```