DEVELOPMENT OF INTERSECTION MODEL AND TRAFFIC SIMULATION WITH SUMO FOR TRAFFIC OPTIMIZATION USING RIENFORCEMENT LEARNING

Divyanshu Upadhyay

17064007

Interim Project Report submitted to

Indian Institute of Technology (BHU) Varanasi

In partial fulfilment for the award of the degree

of

Bachelor of Technology
In Civil Engineering

by

Divyanshu Upadhyay
17064007

under the guidance of

DR. Abhisek Mudgal
(Assistant Professor)
Department of Civil Engineering

DEPARTMENT OF CIVIL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI

JUNE 2020

DECLARATION

I certify that

- the work contained in this report is original and has been done by me under the guidance of my supervisor.
- the work has not been submitted to any other Institute for any degree or diploma.
- I have followed the guidelines provided by the Institute in preparing the report.
- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Divyanshu Upadhyay
17064007

ACKNOWLEDGEMENT

Before getting into the thickest of things, I would like to thank the personalities who were part of my project in numerous ways, those who gave me outstanding support in completing the project from scratch.

It has indeed been a great privilege for me to have Dr. Abhisek Mudgal, Department of Civil Engineering, Indian Institute of Technology (BHU) Varanasi as my mentor for this project. His sincere help and constant encouragement are the motive behind this project. I am indebted to him for his timely and valuable advice.

I am highly grateful to Prof. Prabhat kumar Singh Dixit, Head, Department of Civil Engineering, Indian Institute of Technology (BHU) Varanasi for providing necessary facilities and encouraging me during the course of work. I would also like to thank the research scholars and technical supervisors for their guidance, support and useful critiques towards the project.

My grateful thanks are also extended to my friends for their help in keeping my progress on schedule.

Finally, I wish to thank my brother and parents for their support and encouragement throughout my study.

Divyanshu Upadhyay
Roll No.: 17064007
B.Tech Part - III
Department of Civil Engineering
Indian Institute of Technology (BHU) Varanasi

CONTENTS

ABSTRACT

Due to fast economic development in India, light-duty vehicle transportation for personal use is experiencing radical growth and this led to traffic congestion which is becoming more and more serious in many cities In India, until last two decades, the on-road emissions and their ill effects were not taken seriously, thus, no real effort was carried in transportation emission modelling area. These problems have negative effects on people's living and health. In urban areas, traffic or vehicle emissions are significant pollution sources of the urban atmospheric environment. Generally speaking, signalised intersections are considered as pollution hot spots. Many scholars used different approaches to discuss the mechanism of the formation and dissipation of traffic jam for signalised or unsignalized intersections at a microscopic or macroscopic level and to propose some strategies for reducing the size of traffic jam and the time for jam dissipation.

In this report we discuss about traffic simulation using SUMO, the Simulator of Urban MObility, a software tool. Using SUMO, a model of intersection is made for applying reinforcement learning techniques to traffic light policies with the aim of increasing traffic flow through intersections. We will simulate and evaluate different policies against them. We compare various policies including fixed cycles, longest queue first (LQF), and the reinforcement learning technique Q-learning. We evaluate these policies on a varying type of intersections as well as networks of traffic lights.

INTRODUCTION

1. MOTIVATION

The motivation for this document is to use SUMO for comparing various policies including fixed cycles, longest queue first (LQF), and the reinforcement learning technique Q-learning and try to conclude an optimised case by which we can reduce the size of traffic jam and the time for jam dissipation. For doing this we have used SUMO, the Simulator of Urban MObility, a software tool being developed by the DLR - Institute of Transportation Systems of DLR, the National Aeronautics and Space Research Centre of the Federal Republic of Germany in Berlin. SUMO is an open source (licensed under the GPL), highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks. The simulation platform offers many features, like microscopic simulation, online interaction and the simulation of multimodal traffic. Time schedules of traffic lights can be imported or generated automatically by SUMO and there are no artificial limitations in the network size and the number of simulated vehicles. SUMO uses its own file formats for traffic networks, but it is able to import maps encoded in many publicly available formats like OpenStreetMap, VISUM, VISSIM and NavTeq.

2. OBJECTIVE

The objective of this document is to develop a model of intersection using SUMO traffic simulator. This model is created using the data extracted from OSM database. After validation of model we will apply several algorithms to find the case by which we can reduce the traffic congestion and delay. Since, these intersections are the major cause of traffic emission so this will help us optimisation of traffic emission as well. Also comparing various algorithm, we draw several conclusions about which traffic signal methodology could be best used at the intersection.

THEORY AND LITERATURE REVIEW

1. Understanding SUMO

A. Requirements for Running a SUMO Simulation

There are two ways for running a SUMO simulation. The first one is through the command sumo and the second one through the command sumo-gui. Both of them are command line programs. The difference between them is that sumo runs the simulation in background, without a graphical user interface. In this case, all the simulation runs in background. Using the sumo-gui program, a graphical user interface, showing the running simulation is presented to the user, which can interact with the simulation through this user interface.
There are basically two different pieces of information necessary in order to start a SUMO simulation:

- A Network Topology
- A Traffic Pattern Demand

A Network Topology comprises a network of roads, railways, pedestrian ways, Aquatic routes or other means of moving cars, buses, trams, trucks, trains, boats or people. A Traffic Pattern Demand comprises the cars, buses, trams, trucks, trains, boats or people moving around, in a given pattern along the network.
These two requirements can be unified in terms of a configuration, which needs to be defined in order to run a simulation. A configuration can be defined in SUMO in an XML configuration file.
Besides a Network File, describing the network topology (*.net.xml) and one or more Route Files, describing the traffic pattern in terms of vehicles (*.rou.xml), you might also specify Additional Files (*.add.xml).
Almost every file used in SUMO package is encoded in XML. Most SUMO files (road network descriptions, route and/or demand descriptions, infrastructure descriptions, etc.) are SUMO-specific, not following any standard. For some of the file types used by SUMO, an xsd (XML Schema Definition) exists.

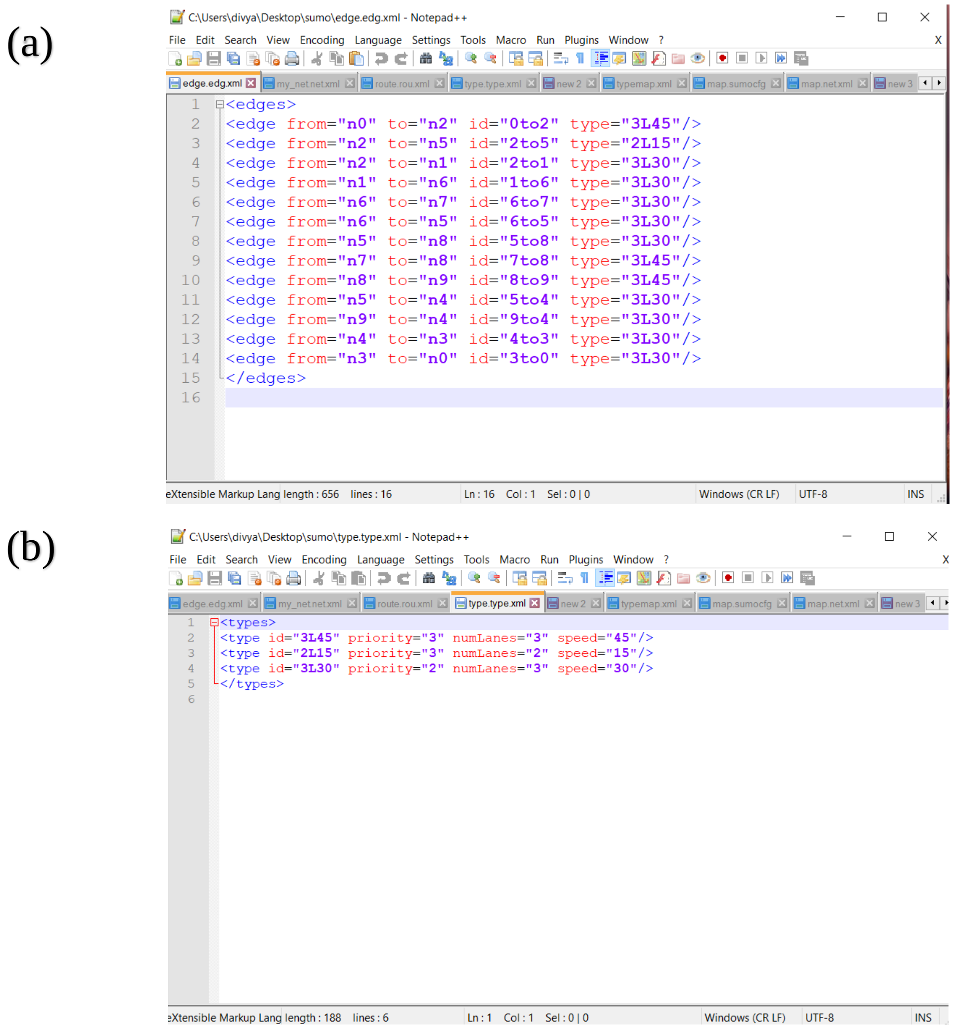
B. Network Topology

A SUMO network file (.net.xml) describes the traffic-related part of a map, the roads and intersections where the simulated vehicles run along or across. At a coarse scale, a SUMO network is a directed graph. Nodes, usually named junctions in SUMO-context, represent intersections, and edges roads or streets. Note that edges are unidirectional. Specifically, the SUMO network contains the following information:

- a collection of edges representing segments of roads, streets, aquatic routes, railways or pedestrian pathways. An edge may have multiple lanes, including the position, shape and speed limit of every lane,
- a collection of nodes or junctions, together with the many requests specifying the right of way regulation of each junction
- a set of traffic light logics referenced by junctions,
- a set of connections between lanes at junctions, describing how from specific lanes in junction, it is possible to turn into other lanes going out of the junction.

Also, depending on the used input formats and set processing options, one can also find

- districts,
- roundabout descriptions.


(a)
C:\Users\divya\Desktop\sumo\edge.edg.xml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
edge.edg.xml
my_net.net.xml
route.rou.xml
type.type.xml
new 2
typemap.xml
map.sumocfg
map.net.xml
new 3
eXtensible Markup Lang length : 656 lines : 16 Ln : 16 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

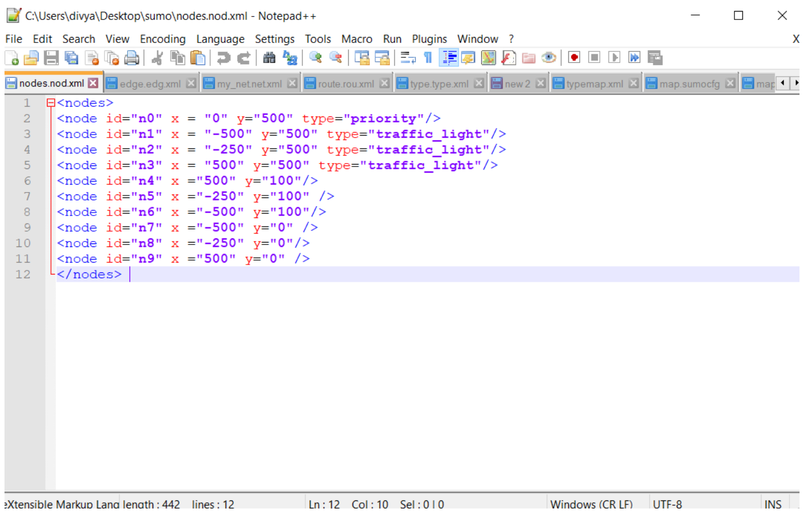
```
<edges>
<edge from="n0" to="n2" id="0to2" type="3L45"/>
<edge from="n2" to="n5" id="2to5" type="2L15"/>
<edge from="n2" to="n1" id="2to1" type="3L30"/>
<edge from="n1" to="n6" id="1to6" type="3L30"/>
<edge from="n6" to="n7" id="6to7" type="3L30"/>
<edge from="n6" to="n5" id="6to5" type="3L30"/>
<edge from="n5" to="n8" id="5to8" type="3L30"/>
<edge from="n7" to="n8" id="7to8" type="3L45"/>
<edge from="n8" to="n9" id="8to9" type="3L45"/>
<edge from="n5" to="n4" id="5to4" type="3L30"/>
<edge from="n9" to="n4" id="9to4" type="3L30"/>
<edge from="n4" to="n3" id="4to3" type="3L30"/>
<edge from="n3" to="n0" id="3to0" type="3L30"/>
</edges>
```

(b)
C:\Users\divya\Desktop\sumo\type.type.xml - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
edge.edg.xml
my_net.net.xml
route.rou.xml
type.type.xml
new 2
typemap.xml
map.sumocfg
map.net.xml
new 3
eXtensible Markup Lang length : 188 lines : 6 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

```
<types>
<type id="3L45" priority="3" numLanes="3" speed="45"/>
<type id="2L15" priority="3" numLanes="2" speed="15"/>
<type id="3L30" priority="2" numLanes="3" speed="30"/>
</types>
```

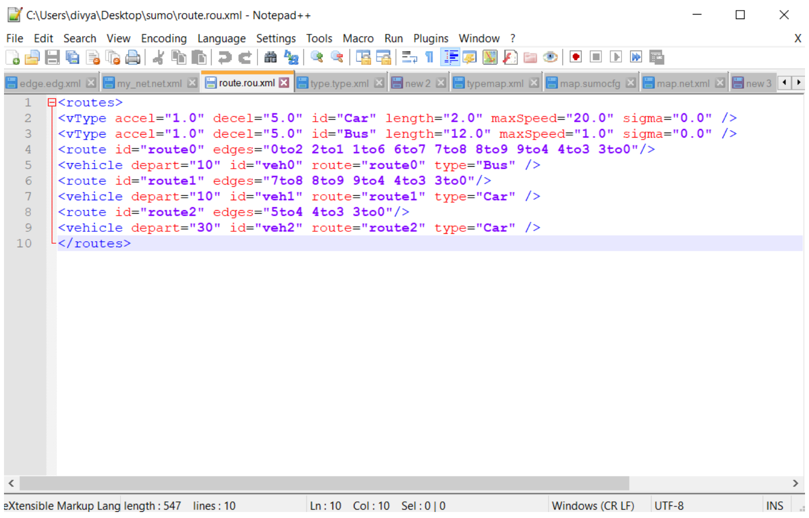
Fig. 1. (a) basic example of edge file (b) basic example of type file

(a)



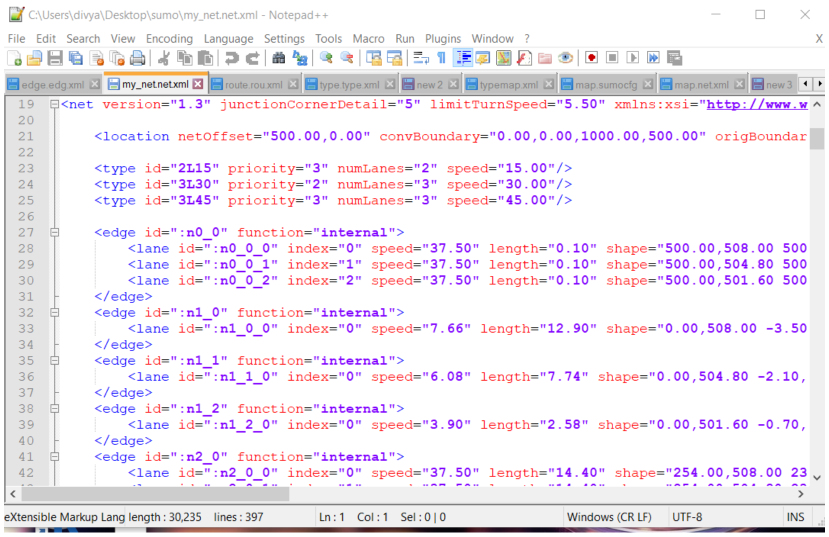
```
<nodes>
<node id="n0" x = "0" y="500" type="priority"/>
<node id="n1" x = "-500" y="500" type="traffic_light"/>
<node id="n2" x = "-250" y="500" type="traffic_light"/>
<node id="n3" x = "500" y="500" type="traffic_light"/>
<node id="n4" x ="500" y="100"/>
<node id="n5" x ="-250" y="100" />
<node id="n6" x ="-500" y="100"/>
<node id="n7" x ="-500" y="0" />
<node id="n8" x ="-250" y="0"/>
<node id="n9" x ="500" y="0" />
</nodes>
```

(b)



```
<routes>
<vType accel="1.0" decel="5.0" id="Car" length="2.0" maxSpeed="20.0" sigma="0.0" />
<vType accel="1.0" decel="5.0" id="Bus" length="12.0" maxSpeed="1.0" sigma="0.0" />
<route id="route0" edges="0to2 2to1 1to6 6to7 7to8 8to9 9to4 4to3 3to0"/>
<vehicle depart="10" id="veh0" route="route0" type="Bus" />
<route id="route1" edges="7to8 8to9 9to4 4to3 3to0"/>
<vehicle depart="10" id="veh1" route="route1" type="Car" />
<route id="route2" edges="5to4 4to3 3to0"/>
<vehicle depart="30" id="veh2" route="route2" type="Car" />
</routes>
```

(c)



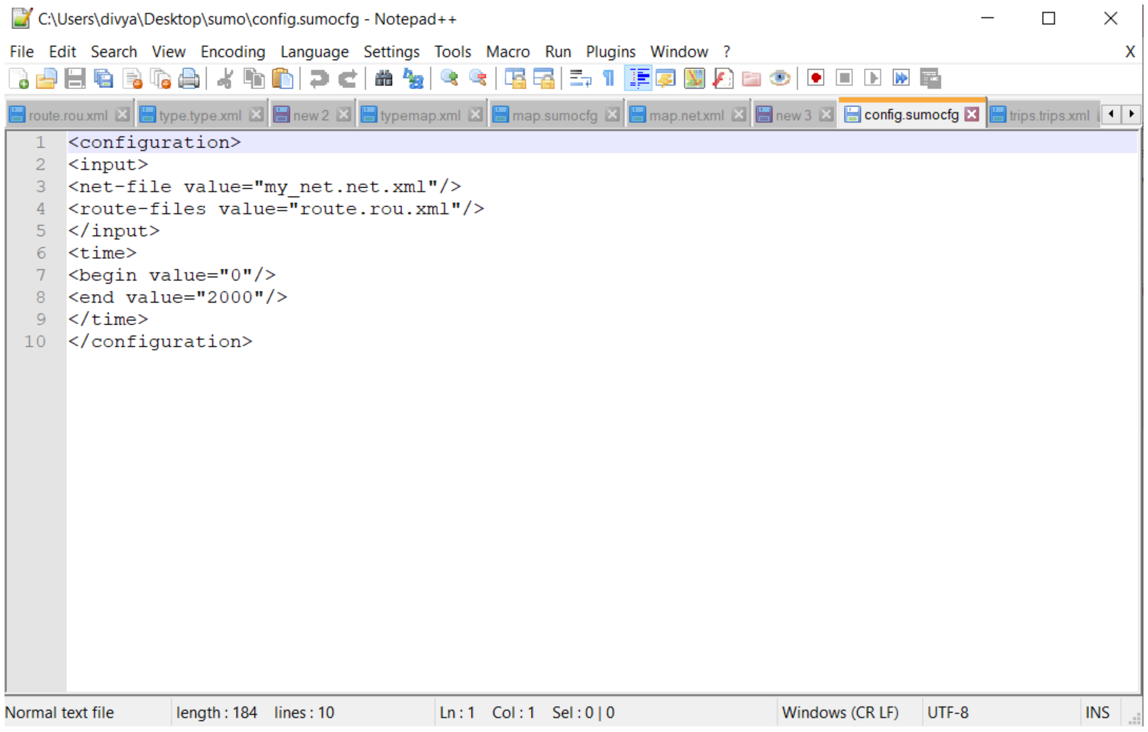
```
<net version="1.3" junctionCornerDetail="5" limitTurnSpeed="5.50" xmlns:xsi="http://www.w

    <location netOffset="500.00,0.00" convBoundary="0.00,0.00,1000.00,500.00" origBoundar

    <type id="2L15" priority="3" numLanes="2" speed="15.00"/>
    <type id="3L30" priority="2" numLanes="3" speed="30.00"/>
    <type id="3L45" priority="3" numLanes="3" speed="45.00"/>

    <edge id=":n0_0" function="internal">
        <lane id=":n0_0_0" index="0" speed="37.50" length="0.10" shape="500.00,508.00 500
        <lane id=":n0_0_1" index="1" speed="37.50" length="0.10" shape="500.00,504.80 500
        <lane id=":n0_0_2" index="2" speed="37.50" length="0.10" shape="500.00,501.60 500
    </edge>
    <edge id=":n1_0" function="internal">
        <lane id=":n1_0_0" index="0" speed="7.66" length="12.90" shape="0.00,508.00 -3.50
    </edge>
    <edge id=":n1_1" function="internal">
        <lane id=":n1_1_0" index="0" speed="6.08" length="7.74" shape="0.00,504.80 -2.10,
    </edge>
    <edge id=":n1_2" function="internal">
        <lane id=":n1_2_0" index="0" speed="3.90" length="2.58" shape="0.00,501.60 -0.70,
    </edge>
    <edge id=":n2_0" function="internal">
        <lane id=":n2_0_0" index="0" speed="37.50" length="14.40" shape="254.00,508.00 23
```

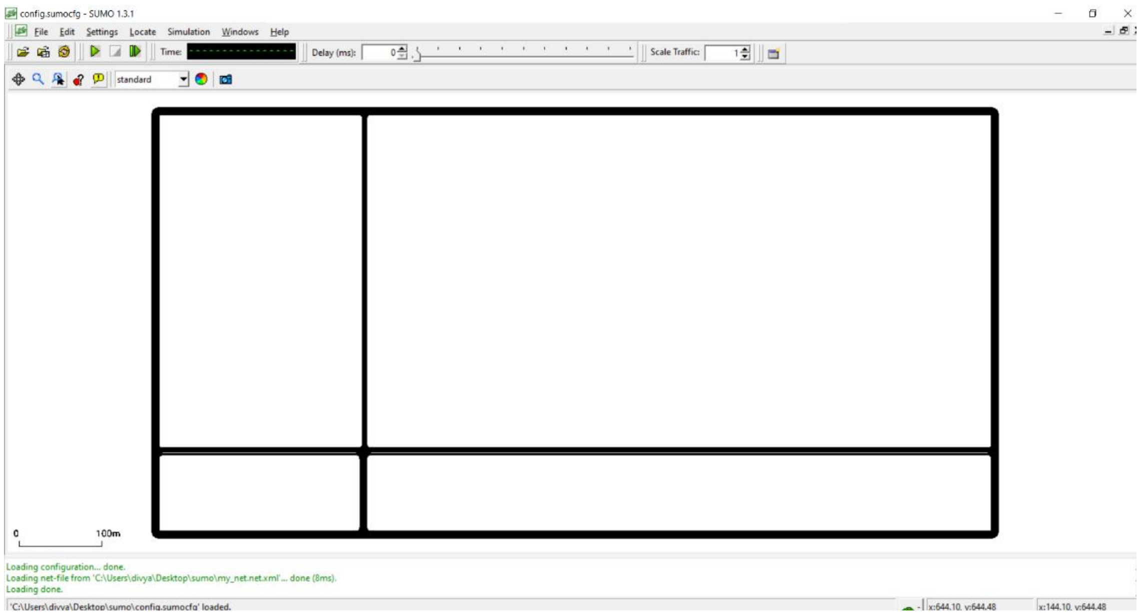
Fig. 2. (a) xml node file (b) xml route file (c) net xml file generated


C:\Users\divya\Desktop\sumo\config.sumocfg - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
route.rou.xml
type.type.xml
new 2
typemap.xml
map.sumocfg
map.net.xml
new 3
config.sumocfg
trips.trips.xml

```
<configuration>
<input>
<net-file value="my_net.net.xml"/>
<route-files value="route.rou.xml"/>
</input>
<time>
<begin value="0"/>
<end value="2000"/>
</time>
</configuration>
```

```
Normal text file    length : 184    lines : 10    Ln : 1    Col : 1    Sel : 0 | 0    Windows (CR LF)    UTF-8    INS
```

Fig. 3. The configuration file for the above-mentioned example.


config.sumocfg - SUMO 1.3.1
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms):
Scale Traffic:
standard
0
100m

```
Loading configuration... done.
Loading net-file from 'C:\Users\divya\Desktop\sumo\my_net.net.xml'... done (8ms).
Loading done.
'C:\Users\divya\Desktop\sumo\config.sumocfg' loaded.    x:644.10, y:644.48    x:144.10, y:644.48
```

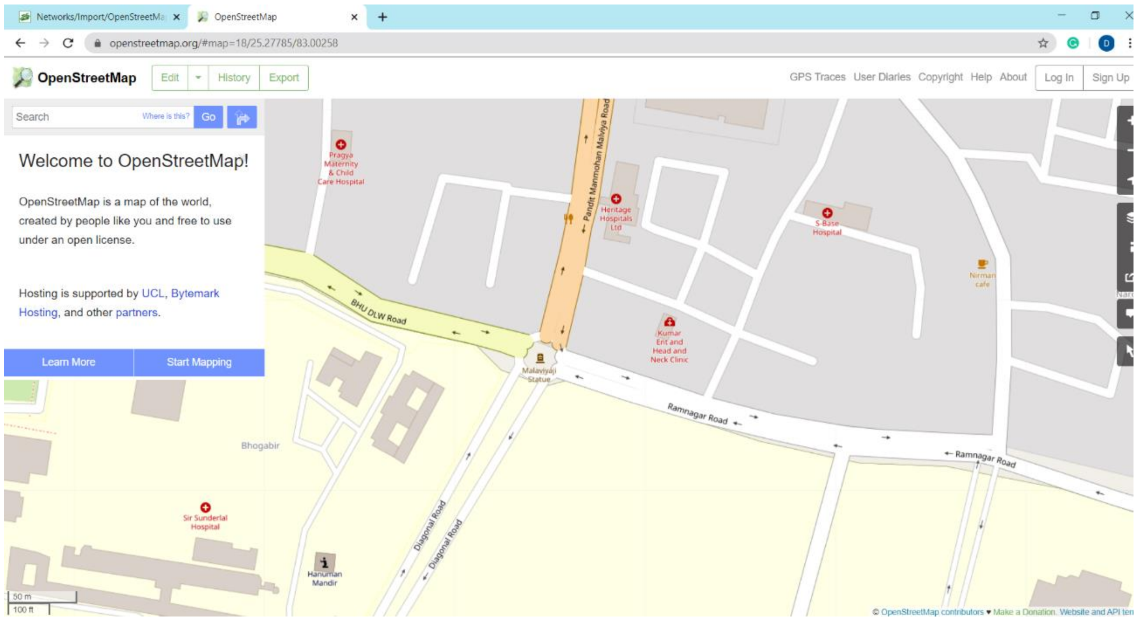
Fig. 4. The result sumo file for the above-mentioned example.

C. Acquiring a map from OpenStreetMap

There are many ways in which a map can be obtained from OpenStreetMap. The simplest way of doing it is using a Web browser and accessing the web page at:

www.openstreetmap.org

From this web page, you might be able to select a portion of the map you wish do download, and export it. An example of this solution can be seen on figure



Welcome to OpenStreetMap!

OpenStreetMap is a map of the world, created by people like you and free to use under an open license.

Hosting is supported by UCL, Bytemark Hosting, and other partners.

Figure 5: Browser interface for the OpenStreeMap web page

A second possibility is using some sort of standalone application enabled to use the OpenStreetMap database. There are many available programs like this at the Internet. One of the most complete ones is JOSM, the Java OpenStreetMap Editor, which can be downloaded for free at:

https://josm.openstreetmap.de/

An example of the JOSM graphical user interface can be shown in figure

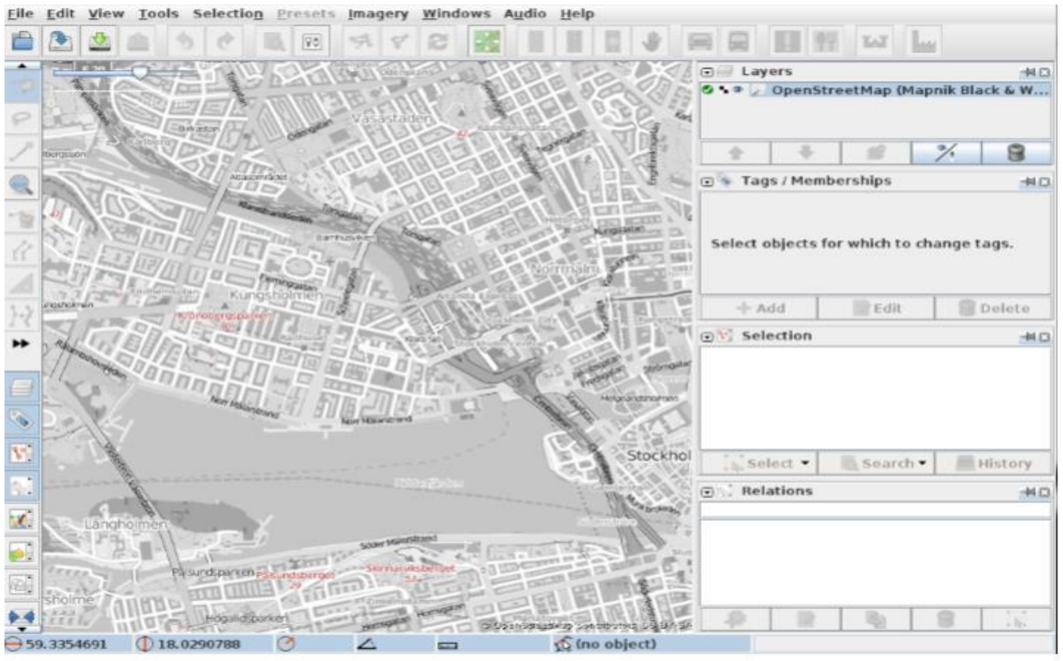

File Edit View Tools Selection Presets Imagery Windows Audio Help
Layers
OpenStreetMap (Mapnik Black & W...
Tags / Memberships
Select objects for which to change tags.
Add
Edit
Delete
Selection
Select
Search
History
Relations
59.3354691
18.0290788
(no object)

Figure 6: Graphical User Interface of the software JOSM

D. Converting from an OSM file to a SUMO .net.xml file

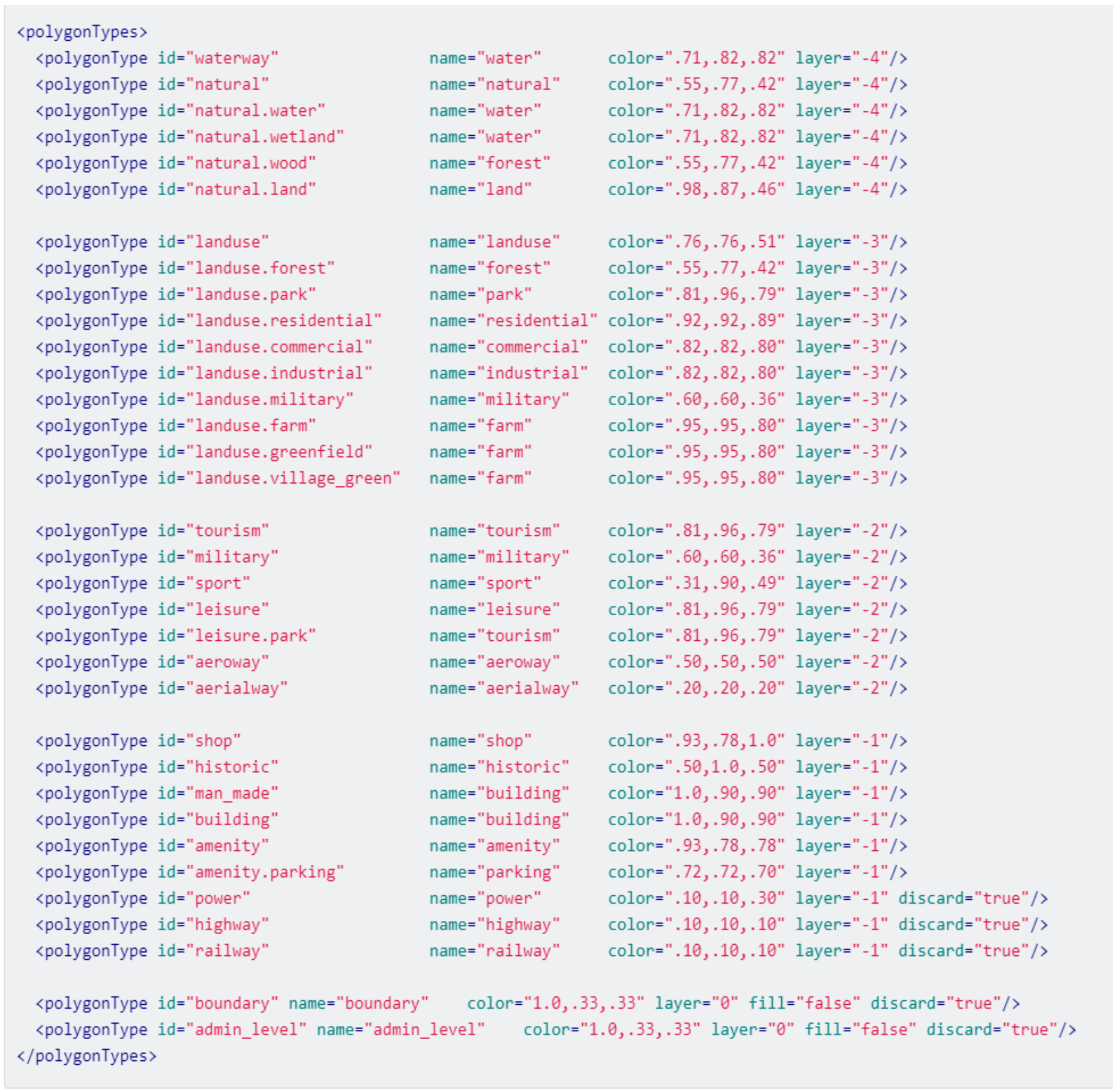
With all these different options, the result will be an XML file, usually named .osm or .osm.xml. You can then use the NETCONVERT tool to convert the OSM file to the .net.xml file format, in order to use it for the SUMO simulation. The simplest way of doing it is just a simple command:

```
netconvert --osm .osm -o .net.xml
```

For a realistic simulation, though, this might be not enough. Sometimes, information such as speed limit is missing in the OSM file and must be inferred from the abstract type of the road (i.e. motorway). Different simulation scenarios require different modes of traffic and thus different parts of the traffic infrastructure to be imported. Usually, conversion options must be specified, as e.g.:

```
--geometry.remove --roundabouts.guess --ramps.guess --junctions.join --
tls.guess-signals --tls.discard-simple --tls.join
```

Besides that, there is also a lot of information in OSM files, like POI (Points of Interest), buildings drawings, regions, etc, which are usually simply discarded while creating the SUMO .net.xml file. It is possible, though, to use type files to convert this information to useful information within SUMO. For example, all this graphic information not important for simulation can be showed in sumo-gui. You may use the POLYCONVERT tool to import polygons from OSM-data and produce a Sumo-polygon file. An example of a type file can below:



```
<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="power"                   name="power"       color=".10,.10,.30" layer="-1" discard="true"/>
  <polygonType id="highway"                 name="highway"     color=".10,.10,.10" layer="-1" discard="true"/>
  <polygonType id="railway"                 name="railway"     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>
```

Figure 7: Example of osmPolyconvert.typ.xml file for OpenStreetMap type conversion

2. Algorithm

2.1. Q-Learning

We use Q-learning with function approximation to learn the best traffic signal actions. Before we detail our specific problem formulation, we first describe the Q-learning algorithm for a general Markov Decision Process (MDP). Under a MDP with fixed transition probabilities and rewards, the Bellman equation (Equation1) gives the optimal policy.

$$Q(s,a) = R(s') + \gamma \max_{a'} Q(s',a') \qquad (1)$$

If the Q function can be correctly estimated, then a greedy policy becomes the optimal policy, and we can choose actions according to Equation 2.

$$\pi(s) = \arg\max_{a} Q(s,a) \qquad (2)$$

We use the function approximation abstraction from (Mnih et al., 2015). From our simulator we extract values for the state features, action, reward, and next state features: (s,a,r,s'). Note that the s variables are not states, but features of states. Thus, Q(s,a) represents the approximate Q value for that state and action. Each (s,a,r,s') tuple is a training example for our Q function. In order to satisfy the Bellman equation (Equation 1), we minimize the loss between our current Q value and our target Q value over parameters θ subject to regularization λ (Equation 3).

$$\min_{\theta} \left(||r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)|| + \lambda ||\theta|| \right) \quad (3)$$

2.2. Q functions

The heart of Q-learning is the Q function used for estimation. In the naive case, the Q function can simply be a lookup table that maps states and actions to Q values, in which case Q-learning is essentially the same as value iteration. However, Q-learning lets us generalize this framework to function approximations where the table of states and actions cannot be computed.
Every part of Equation 3 is differentiable, so if our Q function is differentiable with respect to its parameters, we can run stochastic gradient descent to minimize our loss. For our implementation, we use stochastic gradient descent on a linear regression function. We also performed SGD with a simple one-layer neural network.

APPROACH AND METHODOLOGY

1. Simulator

We have built a model of intersection similar to what it is in real world, and to evaluate our policies, we followed other researchers in using SUMO (Simulation of Urban MObility), an open-source industry-standard software package for modelling road networks and traffic flow. In particular, we used SUMO version 0.24. SUMO allowed us to build different types of road networks, add cars to our simulation and determine their routes, and add sensors for the traffic lights. The SUMO package also comes with TraCI, a Python API that allows a user to get information from a traffic simulation and modify the simulation as it runs in response to this information. We built an architecture that made use of the TraCI interface to get information about queue sizes, carbon emissions, sensor data, and traffic light states, in order to successfully deploy our algorithms.

2. Setup

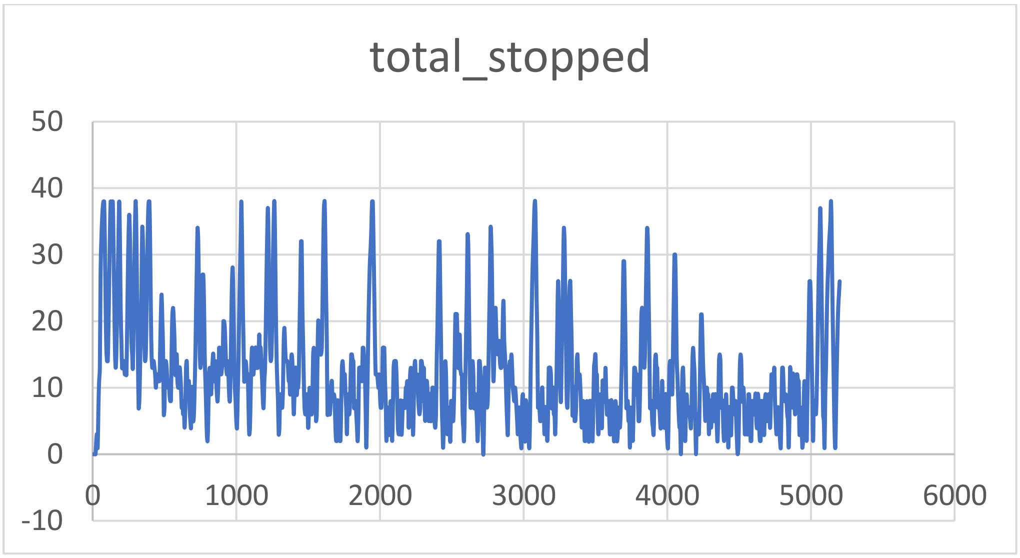
We have chosen an intersection (say BHU Main Gate) and generated a model by extracting data from OSM and then use poly SUMO file to get a model of that intersection. We have attached a RandomTrip.py file to see the flow of random traffic over the intersection. Now our aim is to compare the various algorithm to get the optimised result. For this we will implement four algorithms and compare them against each other. There algorithms are as follows

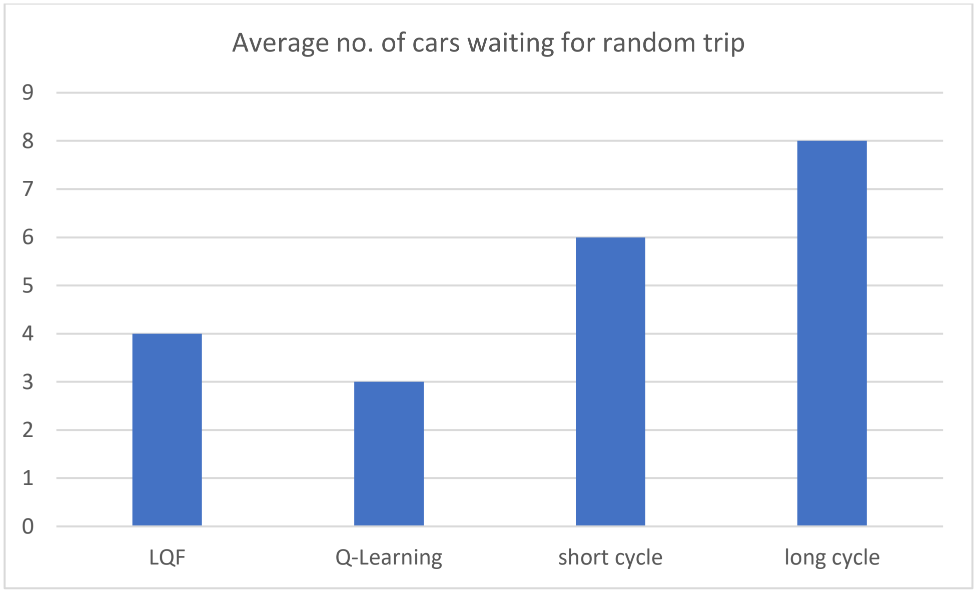
- Q-learning approach
- Short Cycle
- Long Cycle
- Longest Queue First (LQF)

We have already learned about Q-learning approach in previous section. Short Cycle changes the phase every 15 seconds in a round-robin fashion. Long Cycle changes the phase every 45 seconds in a round-robin fashion. LQF chooses to let the direction with the highest number of cars be green. Although LQF is a greedy algorithm, it is given more information than Q-learning, which makes it a reasonable target for comparison.

RESULTS AND DISCUSSION

We have made a model of intersection similar to what it is in real. Now using this model, we simulate data extracted for intersection. After that we will optimise the traffic congestion and traffic delay at the intersection. For doing so we will compare four different algorithms with each other to get the best possible case. These four algorithms consist of Q-learning approach, short cycle, long cycle and Longest Queue First (LQF) techniques. By reducing the traffic congestion and delay we can also reduce the level of emission there, since, we know that intersections are the emission hotspots. Intersection used in this document is BHU main gate intersection which face high congestion during peak hours. So, by concluding a result through simulation will help us decide the phase timing and the methodology used for traffic signal in a better way.





This chart is comparison of different algorithms for the random trip data we will also do this for our BHU gate intersection data and will compare the results in similar manner.

FUTURE WORK

The project has been completed till making the model of the intersection (BHU Main Gate) using SUMO traffic simulator with the help data extracted from OpenStreetMap Data and then we have taken some trails using random trips. Our future work includes taking real time data of that intersection and then using several algorithms to optimise the traffic at that intersection. This includes comparison of various algorithm to conclude an optimised case by which we can reduce the size of traffic jam and the time for jam dissipation. We will look at the CO2 emissions data provided by SUMO. And will try to optimise for traffic emission as well if possible.

REFERENCES

[1] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529– 533, 2015

[2] Krajzewicz, Daniel, Erdmann, Jakob, Behrisch, Michael, and Bieker,Laura. Recent development and applications of SUMO-Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 5 (3&4):128–138, December 2012.

[3] SUMO Web Site - http://sumo.dlr.de

[4] The SUMO Documentation - http://sumo.dlr.de/wiki/Main_Page

[5] Jayakrishnan, R, Mattingly, Stephen P, and McNally, Michael G. Performance study of scoot traffic control system with non-ideal detectorization: field operational test in the city of anaheim. In 80th Annual Meeting of the Transportation Research Board, 2001.

[6] Environmental Protection Agency (EPA): 'Carbon pollution from transportation – transportation and climate change', 2017. Available at http:// www.epa.gov/air-pollution-transportation/carbon-pollution-transportation, 10 January 2017

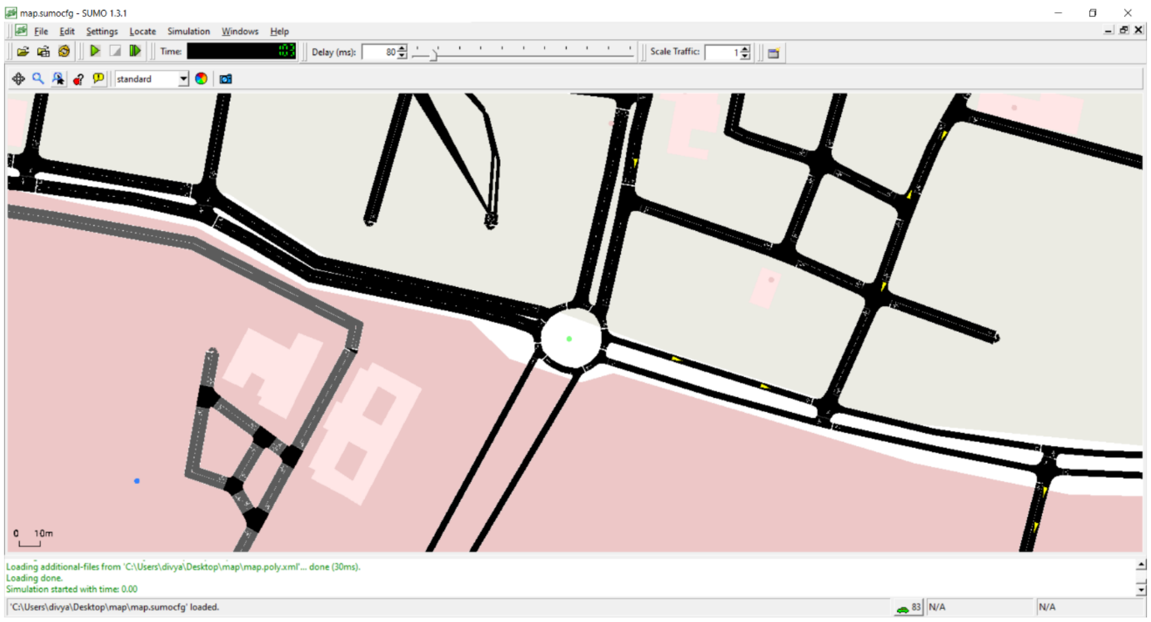
[7] Long, J.C., Gao, Z.Y., Zhao, X.M., et al.: 'Urban traffic jam simulation based on the cell transmission model', Netw. Spat. Econ., 2011, 11, (1), pp. 43–64

[9] Li, H.Q., Lu, H.P., Liu, Q.: 'Research on transition algorithms for timing plans of arterial time in-day traffic control', J. Wuhan Univ. Technol. (Transp. Sci. Eng.), 2008, 32, (5)

[10 Arel, Itamar, Liu, Cong, Urbanik, T, and Kohls, AG. Reinforcement learning-based multi-agent system for network traffic signal control. Intelligent Transport Systems, IET, 4(2):128–135, 2010

APPENDIX

SUMO model of BHU main gate intersection and the code of net file:



```
Loading additional-files from 'C:\Users\divya\Desktop\map\map.poly.xml'... done (30ms).
Loading done.
Simulation started with time: 0.00
'C:\Users\divya\Desktop\map\map.sumocfg' loaded.
```

Net File:

https://docs.google.com/document/d/1KSsmilG7jGKeOgJM2R1Ld1LSH2QgZaeSNU2Brs3oXYw/edit?usp=sharing