VEHICULAR MOBILITY MODELLING

MOBMOD COURSE - FALL 2023
PROF. JÉRÔME HÄRRI AND ALI NADAR

OBJECTIVES

This lab aims at understanding the particular characteristics of large-scale vehicular traffic modeling with the help of the traffic simulator SUMO.
We will investigate:

1. How to import a real street layout from OSM using SUMO.
2. How to configure large-scale mobility between Origin/Destination sectors.
3. How to analyze the traffic output: mobility as well as carbon footprint.

This lab is based on Ubuntu and SUMO 0.31.0, and it's available at
`/datas/teaching/courses/MobMod/Fall2017/Lab3`

GETTING STARTED

We are going to use **SUMO `1.15.0`** and you can find it in **`/packages/sumo_1.15.0`**.

It does not require to be installed: all the executables are going to be in **`sumo_1.15.0/bin`**.

SUMO is a complex simulator and its documentation can be found at
http://sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki.

In this lab we are going to use:

- **`netconvert`**: http://sumo.dlr.de/wiki/NETCONVERT
- **`netedit`**: http://sumo.dlr.de/wiki/NETEDIT
- **od2trips**: http://sumo.dlr.de/wiki/OD2TRIPS
- **duarouter**: http://sumo.dlr.de/wiki/DUAROUTER
- **`sumo`**: http:/sumo.dlr.de/wiki/SUMO and **`sumo-gui`**: http://sumo.dlr.de/wiki/SUMO-GUI

The analysis of the simulation results can be done with a tool of your choice.

SUMO OVERVIEW

SUMO and the SUMO tools can be used with configuration files or directly from command line. The configuration files have a common structure and they are XML files. In order for SUMO to work, it requires in input the definition of a network and the definition of a traffic demand (and it can be customized with additional xml files). The results of a simulation are generated (on demand) and are in XML format too.

The best way to know what kind of parameters are available in SUMO and all the tools are:

- **`program --help`**
- **`program --save-template program.cfg`**

In the Lab3 folder you can find all the configuration files that are we going to use in this lab, commented.

HOW TO IMPORT A REAL STREET LAYOUT FROM OSM USING SUMO TOOLS

The network topology in SUMO is defined with nodes and edges in an XML file. SUMO provides tools to import networks in various formats.

OPENSTREETMAP AND NETCONVERT

We are interested in maps exported from OpenStreetMap (https://www.openstreetmap.org). OSM is built by a community of mappers that contribute and maintain the data. It contains a lot of local knowledge about roads, trails, cafés, railway stations, and much more, all over the world. Last but not least, it's open data.

In order to convert from OSM to SUMO format, we are going to use **netconvert**. The detailed explanation of the process is available at http://sumo.dlr.de/wiki/Networks/Import/OpenStreetMap.

One of the tools that SUMO provides in order to quick-start the generation of a realistic traffic scenario, it's called OSMWebWizard and a tutorial is available at http://sumo.dlr.de/wiki/Tutorials/OSMWebWizard. It generates many intermediate files during the execution, and it can be useful to understand the process.

NETEDIT

netedit is a graphical editor for SUMO, it's quite complex to use, so we advise to backup your files before modifying them with it. An overview of the functionality is available at http://sumo.dlr.de/wiki/NETEDIT.

HOW TO CONFIGURE A TRAFFIC DEMAND USING SUMO TOOLS

The traffic demand is mainly defined using trips and flows. A trip is associated to a vehicle and it has an origin, a destination and a departure time. Flows are defined once for many vehicles, over a period of time, with origin and destination. Trips and flows are used to generate routes for each vehicle in the simulation.

PUBLIC TRANSPORTS

With **netcovert** is possible to extract public transports information from OSM. Once the information is extracted, it can be converted to flows using **sumo_1.15.0/tools/ptlines2flows.py**. The explanation is available at http://sumo.dlr.de/wiki/Tutorials/PT_from_OpenStreetMap.

TRAFFIC DEMAND BASED ON TRAFFIC ASSIGNMENT ZONES (TAZ)

Among the tools provided by SUMO, we can find **od2trips** http://sumo.dlr.de/wiki/OD2TRIPS. Starting from an Origin/Destination matrix, that can be based on edges, or TAZs, it computes the trips. The explanation on how to do it is available at http://sumo.dlr.de/wiki/Demand/Importing_O/D_Matrices. A TAZ, or district, is define as follow:

```
<tazs>
   <taz id="<TAZ_ID>" edges="<EDGE_ID> .. <EDGE_ID>" shape="<COORD> .. <COORD>"/>
   <taz id="<TAZ_ID>" edges="<EDGE_ID> .. <EDGE_ID>" shape="<COORD> .. <COORD>"/>
   ...
</tazs>
```

The shape of a TAZ can be derived from polygons using **netedit**. Starting from a TAZ definition with empty edges but containing their shape, and by using **sumo_1.15.0/tools/edgesInDistricts.py** (showed among other tool for TAZs: http://sumo.dlr.de/wiki/Tools/District#edgesInDistricts.py) it is possible to have a fully functional TAZ definition.

Note: when using the trip definition generated with **od2trips** directly with the advanced routing in SUMO, it's possible to have a routing error. This is due to the random choice of origin and destination edge. In this specific case, the easy solution is to generate the routes with **duarouter** and the parameter **ignore-errors** set to true.

SUMO SIMULATION

Once we have a trips file or a route file, we can run it with SUMO and SUMO-GUI. SUMO requires in input the network and trips/routes and it provides various output files. The more important for us are:

- summary-output: saves the aggregated vehicle departure info
- tripinfo-output: saves each vehicle trip info

The **sumo-gui** program can be launched with the following script:

```
#!/bin/bash
export SUMO_HOME=/packages/sumo_1.15.0
export SUMO_TOOLS=$SUMO_HOME/tools
$SUMO_HOME/bin/sumo-gui -c sumo.cfg
```

RESULT EVALUATION

SUMO output is saved in XML format. Among the tools provided by SUMO, you can find a XML to CSV converter: **sumo_1.15.0/tools/xml/xml2csv.py -i file.xml -o file.xml**

Once you have the results in CSV format, you can evaluate them with your tool of choice.

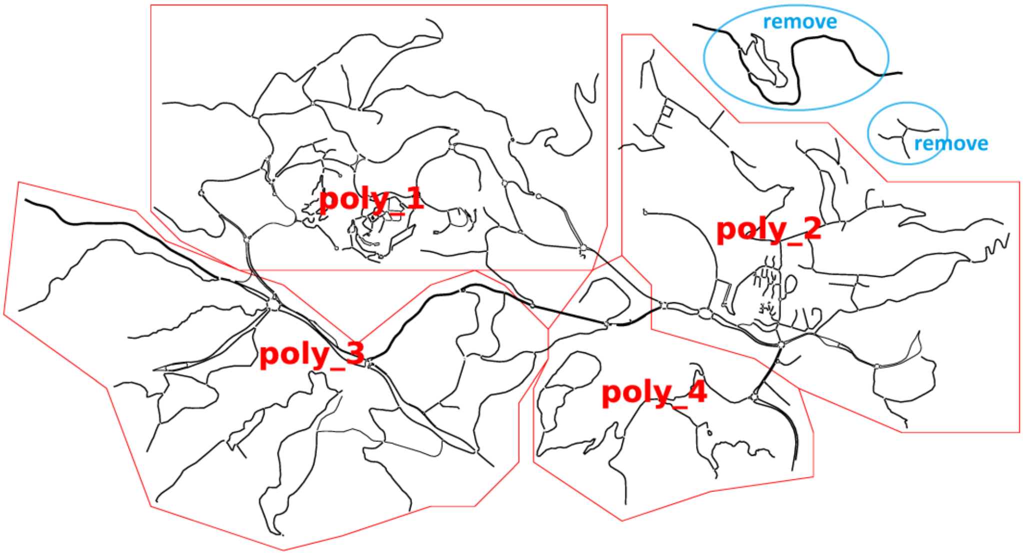
SUMO provides some visualization tools too:

```
sumo_1.15.0/tools/visualization/plot_summary.py -i summary.xml --measure=tag
sumo_1.15.0/tools/visualization/plot_tripinfo_distributions.py -i tripinfo.xml --
measure=tag
```

ASSIGNMENT

Generate the Sophia Antipolis SUMO Traffic Scenario.

1. Create the `netconvert` configuration file to import the **map.sophia.antipolis.osm** file.
    a. In order to simplify the scenario, import only the **passenger** mobility using the **keep-edges.by-vclass** option, remove the **isolated edges**, and the **internal links**.
    b. Export the public transports information with **ptstop-output** and **ptline-output**.
    c. Use only **actuated** traffic lights. Remember to remove the **turnarounds**.
2. For the public transports, generate the bus flows using `ptlines2flows.py` and the files previously created with the **ptstop-output** and **ptline-output** parameters.
3. Using `netedit`

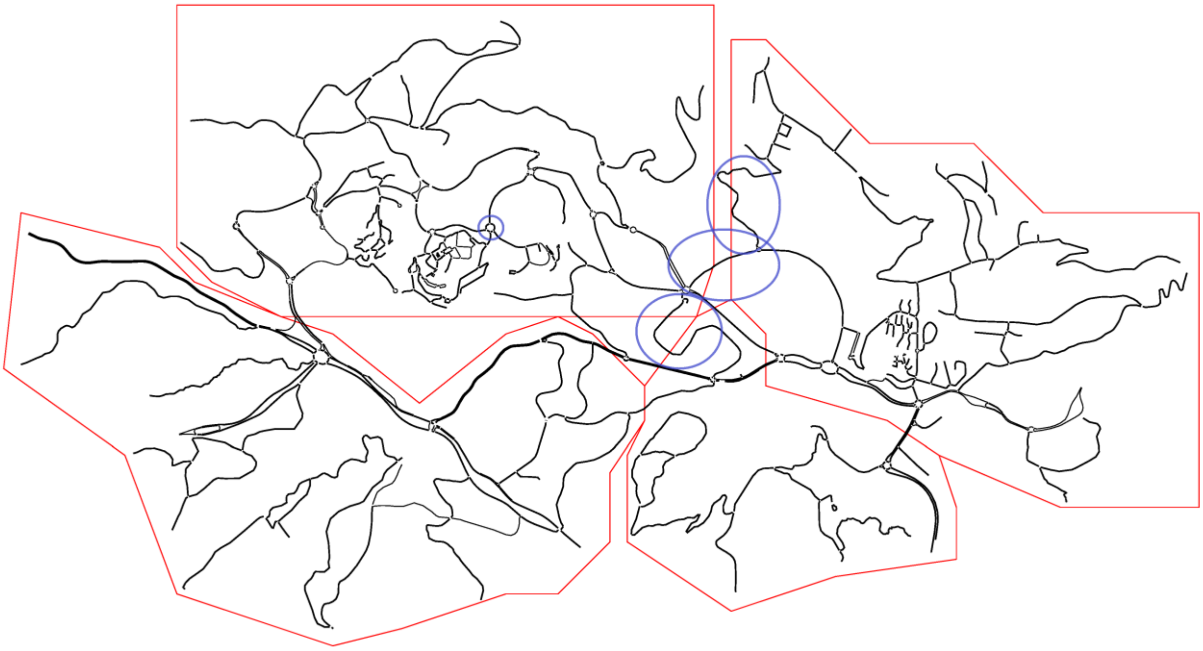


    a. Remove the streets that are isolated.
    b. Generate the polygons associated to the TAZ in red.
4. From the polygons description, create the empty TAZ file, using only the id and the shape attributes.
5. Generate the TAZ file from the empty TAZ previously created and `edgesInDistricts.py`.
6. Generate flows and trips using `od2trips` and the following OD matrix:

From TAZ	To TAZ	Vehicles
1	2	0.14
1	3	0.16
1	4	0.6
2	1	0.86
2	3	0.75
2	4	0.28
3	1	0.86
3	2	0.5
3	4	0.09
4	1	0.47
4	2	0.83
4	3	0.62

    a. With time interval from 0.00 to 1.00 and a factor of 800.0, tse the O format presented in http://sumo.dlr.de/wiki/Demand/Importing_O/D_Matrices#The_O-format to define the OD matrix.

7. Generate the routes from the trips using **`duarouter`**. Use the option to ignore the errors, because not all the trip generated with **`od2trips`** are possible to use.
8. Generate the SUMO configuration file containing the **ptstops** definition, the **buses** and the **passengers routes**. Analyze the mobility using the usual **summary** and **tripinfo** results.
    a. There is a problem in this network (easily visible if you increase the size of the vehicles, and color them by time loss). Where and what is the problem?
9. Using **`netedit`** fix the network (blue ellipses):



10. Repeat the analysis in 8. Is the problem solved? What are the differences? Support your analysis with graphs.