ICT for Smart Mobility

Exercise – Traffic simulation with SUMO
Optimize traffic lights at an intersection with SUMO

In this practical activity you will use the transport and traffic simulator SUMO (https://sumo.dlr.de/docs/index.html).

NOTE: The goal of the lab is not to be proficient with SUMO and learn all its features. The goal is to understand its building blocks, the advantages of a simulator and its capabilities.

To install SUMO on your device (Windows/Linux/macOS): https://sumo.dlr.de/docs/Downloads.php
I suggest to install it through Python packaging index: `pip install eclipse-sumo` (Current version 1.19)

NOTE: you might have to manually set the environment variable **SUMO_HOME** to the base directory of the sumo installation. Setting environment variables is explained here: https://sumo.dlr.de/docs/Basics/Basic_Computer_Skills.html#configuring_path_settings

STEP A – Create and simulate your first intersection

To be able to perform the following steps, take a look at the following tutorial:
https://sumo.dlr.de/docs/Tutorials/Hello_World.html

1. In SUMO, first you need to create a road/street network. In SUMO a street network consists of nodes (junctions) and edges (streets connecting the junctions). Routes are defined by connecting edges and assigning vehicles that pass through them.

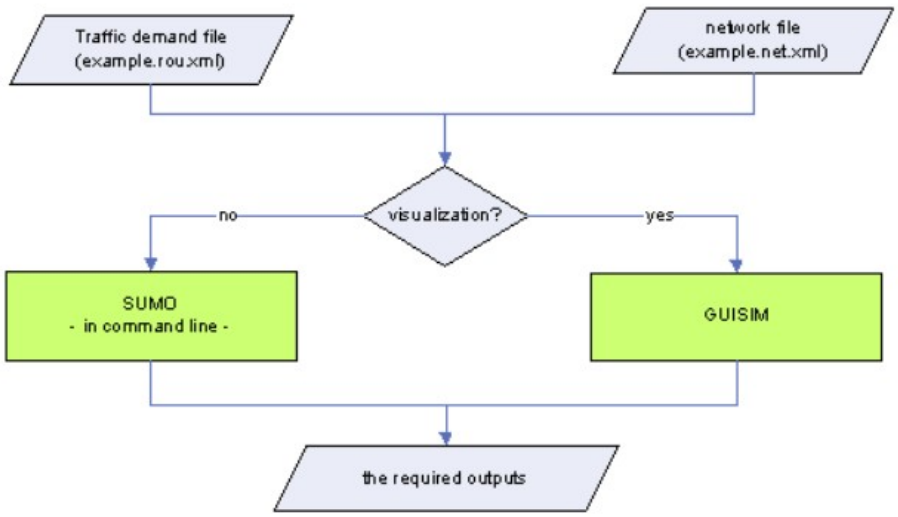
- Create an intersection with at least tree streets, possibly with multiple lanes and both edge directions.

Use **netedit** to create network. The network will be saved as a network file: **example.net.xml**

2. Secondly, you need to **create the demand**, i.e., defining vehicles performing routes. Use **netedit** also to create traffic demand file (**example.rou.xml**)

3. Now you are ready to simulate your scenario. You need a running configuration (**example.sumocfg**). To visualize the result, use **sumo-gui**. To save the metrics of interest, use **sumo** through command line (see step B)

If you want to have more details on each single step, take a look at this other following tutorial:
https://sumo.dlr.de/docs/Tutorials/quick_start.html



STEP B – Change parameters of your intersection

Now **change the configuration or parameters** of your intersection, **keeping the same demand**.

- First, take a look at the simulation at sumo-gui

As output, SUMO allows to generate a large number of different measures. All of them write the values they collect into files or a socket connection. Take a look at:
https://sumo.dlr.de/docs/Simulation/Output/index.html
https://sumo.dlr.de/docs/Simulation/Output/StatisticOutput.html

For example, try to run from the command line: `sumo -c example.sumocfg --duration-log.statistics --statistic-output example.txt`

- Keep track of some metrics, quantify the difference of the two configurations of the intersection that you prepared.

STEP C – Add traffic lights to the intersection

Now **add traffic lights** in the intersection. `Network→ set traffic light mode`

- Run a simulation with the new intersection and collect statistics (sumo and sumo-gui)
- Understand the different phases of the traffic light

See details about traffic lights in SUMO: https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html

STEP D – Optimize phase duration

- Improve the traffic light phase duration in the cycle.

In particular, focus on a metric to optimize (e.g., maximize `speed`) and find the best values of duration of green lights. You can find in `netedit`, by selecting `Phases` in `Edit Traffic Light`.

NOTE: For this exercise, it is enough to change it manually on netedit and collect the results.

NOTE: you can automatically generate random trips on a network by using the tool randomTrips.py
https://sumo.dlr.de/docs/Tools/Trip.html
For example, try to run from the command line:

```
python3 <your SUMO installation folder>/sumo/tools/randomTrips.py -n example.net.xml -o exampleFlow.xml --route-file exampleFlow.rou.xml
```

- What is the best value of the metric you obtained?
- What are the corresponding values in seconds of the phases?

For performing a complete analysis of many SUMO random simulations, take a look at the following tutorial: https://medium.com/data-science/how-to-simulate-traffic-on-urban-networks-using-sumo-a2ef172e564 (code at https://github.com/skandavivek/sumo-traffic-grids)
This uses Python to automatize run and results collection. However, in this case Python is not interacting with the simulations.

STEP E – OPTIONAL - Interact with the traffic signal with Python

- Create a **transit signal priority scheme** for your intersection, where one of the road is exclusive for public transport. Insert an **induction loop** to recognize entering vehicles. When a vehicle enters the induction loop, switch the signal immediately so the vehicle can cross the intersection without a stop.

You can also write your controller in Python. To control a running road traffic simulation with Python, you need to use the Traffic Control Interface (TraCI) . TraCI uses a TCP-based client/server architecture where SUMO acts as a server and the external script (the “controller”) is the client.

You can find an example of the solution here:
https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html

STEP F – OPTIONAL – Reinforcement Learning for traffic signal control

In step E, you designed a way to interact with the traffic signals. Using **Reinforcement Learning**, the algorithm should **learn the optimal policy to follow, depending on the current state of the system**.

A solution of the problem might be found at: https://github.com/suessmann/intelligent_traffic_lights

STEP G – OPTIONAL – Further readings and tutorials

- This a tutorial on how to automatically generate complex scenarios from real road network by using open street maps: https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html
  `python3 <your SUMO installation folder>/sumo/tools/osmWebWizard.py`
- Another framework to apply RL to SUMO: https://flow-project.github.io/ (https://github.com/flow-project/flow)