Using SUMO in a distributed manner to generate road traffic data.

Indrek Loolaid
Supervisor: Amnir Hadachi

Institute of Computer Science, University of Tartu, Estonia

Abstract. In this report I give an overview of how the road traffic simulation tool SUMO can be used to generate traffic data (to alleviate the situation of not having actual road traffic data at hand). More speciffically the aim is to run SUMO in a distributed manner. However SUMO itself is not a distributed application. So to overcome that obstacle the idea is to run several instances of SUMO (one instance for a given area) and set up communication between them so that vehicles can move from one area of simulation to another.

Keywords: road traffic simulation, distributed systems, SUMO, TraCI, OpenStreetMap

1 Introduction

Road traffic simulations can be useful for several purposes. For example simulations can give insight into how traffic is affected when certain changes in traffic routing are made. So changes to traffic management that affect the flow of traffic negatively can be avoided and conversely beneficial changes can be discovered.

There are three models of traffic simulation: macroscopic, mesoscopic and microscopic. A macroscopic model deals with aggregate properties of traffic such as average velocity and traffic density. A microscopic model simulates individual vehicles. A mesoscopic model is a mix between a microscopic and macroscopic model. It simulates individual vehicles but interactions between them are simulated in a macroscopic level. [1]

2 Overview of SUMO

SUMO [2] (Simulation of Urban MObility) is an open source road traffic simulation package that is intended to simulate traffic in an area the size of a city. The package contains the simulation program itself and several additional tools to help set up a road network and generate traffic for a simulation.

SUMO is a microscopic traffic simulator so it keeps track of individual vehicles during simulation. The simulation is time-discrete and space-continuous. The simulation is advanced in steps, the default length of a time step is one second, but that can be set as low as one millisecond. Internally a vehicle's position is determined by the lane it is on and distance from the start of that lane. A vehicle's speed is calculated using a car-following model. This means that when determining it's speed the speed of the leading vehicle and distance to it is also taken into account. [3]

I chose SUMO as the simulation software for this project because in addition to meeting the necessary criteria - open source, microscopic simulation, can use OpenStreetMap data - it had the most features and was very well documented when compared to some alternatives (e.g MATSim) I considered.

3 Road network

To run a simulation a network file is needed. A network file is simply an XML file that contains information about how junctions (nodes) are connected with roads (edges consisting of one or more lanes), how lanes are connected at junctions, what are the top speeds and shapes of lanes and also information about traffic light logic.

There are several ways to obtain such a network file, but most of them mean using the `netconvert` tool that comes with SUMO. One possibility is to provide SUMO-specific XML files as input for `netconvert`. Those inputs include an XML file to list all nodes, another one to list all edges, one for describing edge types (i.e what is the maximum speed, number of lanes, allowed/disallowed vehicle types etc. for a given edge type) and finally a file to describe how edges are connected in nodes (e.g to specify if left turns are allowed in a given junction or not). The edge type and connection files are optional and SUMO

uses default values if edges don't have any type information specified and guesses the connections at junctions if unspecified. [4]

It is also possible to generate a random road network using the `netgenerate` tool. However using a network that exists in the real world is likely to be more desirable. To that end `netconvert` can be used to import network files used by other simulation software such as VISUM, Vissim, MATSim etc. But probably the most useful feature is the ability to import road networks from OpenStreetMap (OSM). OSM is like the Wikipedia of maps, meaning that anyone can freely use and contribute to the geographic data that OSM provides [5]. This means that it's easy and cheap to obtain real road networks that can be used in traffic simulations.

To use an OSM map for creating a road network for SUMO it first has to be downloaded. That can be done manually from http://www.openstreetmap.org/ by either selecting a rectangular area from the map or specifying a bounding box with geographic coordinates. But it is also possible to obtain a map programmatically by sending an HTTP request to the OSM API [6,7]. SUMO even provides a Python script for making the API call.

The downloaded OSM map is simply an XML file that contains a list of data primitives - nodes (a point on the earth's surface), ways (an ordered list of nodes) and relations (a data structure that documents relations between other elements) [8,9]. This OSM map file can then be given as input to `netconvert` to transform it into a network file that SUMO can use [10].

4 Generating traffic

Once there is a network file, it's time to describe the traffic that will flow on that network. This is done in another XML file called the routes file. Three types of information is in a routes file - vehicle types, the routes vehicles can take and vehicles themselves [11].

A vehicle type defines the physical properties of a vehicle and the behaviour of the driver. Examples of physical properties include maximum speed, acceleration and vehicle emissions class. Driver behaviour can be affected for example with the so called sigma attribute which determines the imperfection of the driver. The value of sigma is between 0.0 and 1.0 where a lower value means "more perfect". Sigma value of 0.0 results in a vehicle that behaves deterministically.

Routes are a list of connected edges along which a vehicle can drive. And a vehicle is defined by what type it is, what route it takes and at what time step it enters the simulation. There are more parameters that can be specified for a vehicle, but we will no focus on them here.

Instead of defining each individual vehicle explicitly it is possible to define vehicle type and route probability distributions. So a vehicle type and route will be determined at simulation runtime with defined probabilities. And there is the possibility to indclude several vehicles taking the same route by defining flows. A flow determines a time period during which a certain number of vehicles (of the same type) are added into the simulation to take the specified route. Vehicles created by a flow are distributed equally over the given interval.

It is possible to write the XML to describe the traffic by hand, but it can be generated with the `netconvert` tool that comes with SUMO. To generate a routes file `activitygen` needs a network file and a statistics file describing the demographics of the city/area as input. The statistics file describes some general aspects of the city such as number of households, number of habitants and the probability of an adult owning a car and more specific details including (but not limited to) work hours, population and work position distribution, locations of schools and information about bus lines [12]. The created route file is not yet usable because the routes are defined only by the first and last edge so another tool - `duarouter` - must be used to calculate the exact path a vehicle will take.

5 Simulation data extraction and visualization

When the road network and traffic descriptions are ready the simulation can be run. For the simulation to be useful the user needs some kind of output. The most obvious output is the visualization generated by the SUMO GUI. But there are several other output formats that SUMO can provide about different aspects of the simulation. There's the option to write to a file the position of every vehicle at each time step where the position is identified either by xy-coordinates or by the position on a given lane on a given edge. And there's also the possibility to output aggregated information about vehicle trips (route length, travel time, time spent waiting etc), aggregated information about edges/lanes and information about traffic lights.

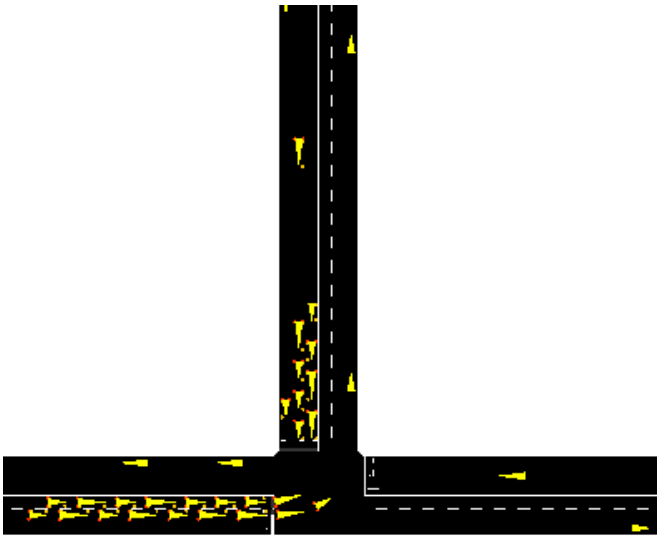

Fig. 1. Example of a simulation visualization by SUMO GUI

There are a few python scripts included in SUMO to help visualize the road network and plot some of the aggregated outputs mentioned before [13]. Figures 2 and 3 are examples of these kinds of visualizations.

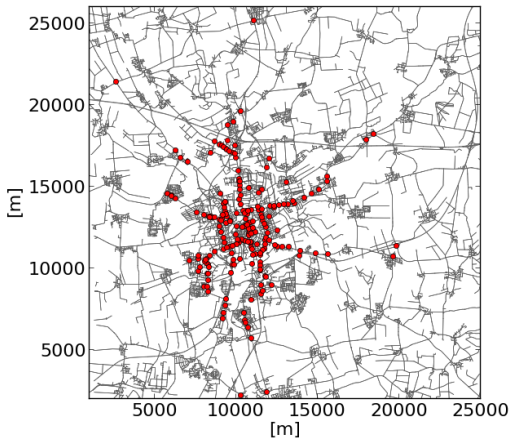

Fig. 2. All traffic light plotted on a network (http://sumo-sim.org/userdoc/images/Plot_net_trafficLights.png)

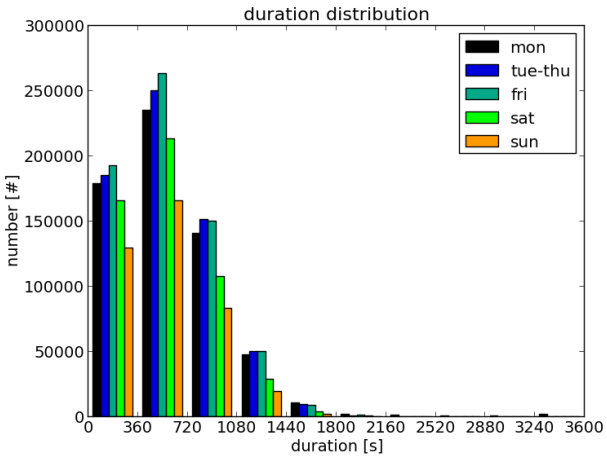

Fig. 3. Duration distribution of trips (http://sumo-sim.org/userdoc/images/Tripinfo_distribution_duration.png)

6 Running SUMO in a distributed way

To run a large scale simulations it would be necessary to distribute the simulation across several computational nodes. However SUMO can't run in a distributed environment on its own. One solution, as proposed in [14], would be to partition the area where traffic is to be simulated into several smaller regions, run a SUMO instance for each of these sub-regions and send messages between neighboring regions to move vehicles from one area to another.

Running the simulation this way means that the simulation state has to be changed while it's running (to introduce new vehicles coming from adjacent areas to a running simulation). This can be done with TraCI (Traffic Control Interface) [15]. TraCI uses a TCP based client/server architecture to access SUMO. In this case SUMO acts as a server and e.g a Python script acts as a client. The client can then use TraCI to ask for or change the state of the running simulation. There are many attributes that can be queried and changed but for the current purpose, being able to to list all vehicles, retrieve the attributes of a specific vehicle and add a new vehicle with given attributes are most important. We do not need to worry about removing vehicles, SUMO does this automatically when a vehicle reaches its destination.

When SUMO is run as a server (controlled by TraCI) then the simulation advances only when the command to simulate the next time step is given. This makes it quite easy to keep the distributed simulation in sync. A

client for a given area must connect to clients controlling the simulation of its neighbouring areas. Then after a simulation step is done the client must determine which vehicles crossed the border between areas and send respective notifications. In case no vehicle crossed a given border an empty message is still sent. When a client has received messages from all its neighbours it can add the specified vehicles to its simulation and advance the time step.

Then there's the problem of identifying which vehicles cross which area border. One way to to it might be to first identify which vehicles are on a route that takes it out of the current area. That means identifying routes with the endpoint at the border of the area and checking if a vehicles route is any of those. Then for those vehicles store its state in a temporary variable before advancing the simulation and when the given vehicle is removed from the next simulation step (by SUMO) then send a message containing this vehicles attributes and the id of the node where it should continue to the client controlling the destination area. This means that some sort of mapping has to be maintained for which nodes correspond to each other in adjacesnt areas either explicitly (e.g a separate mapping file) or implicitly (e.g a node naming convention).

Although with a bit of work it would be possible to maintain routes across several areas, it would be simpler to instead predefine a number of routes for each border node that a vehicle arriving at that node can take and when a vehicle does arrive assign to it one of those routes.

To visualize a distributed simulation the easiest way seems to be to write the vehicle positions at each time step to output files and later plot them to a map. Additionally the SUMO documentation mentions the possibility of using TraCI to take a screenshot at each time step which can be combined into a video [16].

7 Summary

In this report I have given a brief introduction of the road traffic simulation package SUMO and how it can be used to create a road network for traffic simulation or import a network from OpenStreetMap. I have also described some ways how to generate traffic data for the simulation and how to output the simulation results. Finally I have described in general terms how SUMO could be used to run a distributed traffic simulation.

References

1. Ding Ding. Modeling and simulation of highway traffic using a cellular automaton approach. 2011. http://www.diva-portal.org/smash/get/diva2:483914/FULLTEXT01.pdf.
2. http://sumo-sim.org/.
3. Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 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. http://sumo-sim.org/pdf/sysmea_v5_n34_2012_4.pdf.
4. http://sumo-sim.org/userdoc/Networks/Building_Networks_from_own_XML-descriptions.html.
5. http://www.openstreetmap.org/copyright/en.
6. http://wiki.openstreetmap.org/wiki/Download.
7. http://wiki.openstreetmap.org/wiki/Overpass_API.
8. http://wiki.openstreetmap.org/wiki/OSM_XML.
9. http://wiki.openstreetmap.org/wiki/Data_Primitives.
10. http://sumo-sim.org/userdoc/Networks/Import/OpenStreetMap.html.
11. http://sumo-sim.org/userdoc/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html.
12. http://sumo-sim.org/userdoc/Demand/Activity-based_Demand_Generation.html.
13. http://sumo-sim.org/userdoc/Tools/Visualization.html.
14. A. Ventresque, Q. Bragard, E.S. Liu, D. Nowak, L. Murphy, G. Theodoropoulos, and Qi Liu. Spartsim: A space partitioning guided by road network for distributed traffic simulations. In *Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on*, pages 202–209, Oct 2012. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6365073.
15. http://sumo-sim.org/userdoc/TraCI.html.
16. http://sumo-sim.org/userdoc/FAQ.html#Building_videos_from_SUMO-GUI.