Problem: SUMO does not actually support taxi dispatching in weakly connected network, because taxi should be able to go from any edge to any edge (e.g. with idle=”circling” parameter).

Goal: to create strongly connected network for any given SUMO network file and to create minimum working example of ride-pooling with random demand.

Steps:

1. Produce plain xml files from original \*.net.xml file:

netconvert -s area1\_connected.net.xml -p plain

It produces the following files:

plain.nod.xml – node definitions

plain.edg.xml – edges and lanes definition

plain.con.xml – connections definition. Connection in SUMO defines the possibility of moving from (edge1, lane1) to (edge2, lane2)

plain.tll.xml – contains traffic lights logic

plain.typ.xml – contains types of roads

1. Use the script plain\_scc.py which produces new plain xml files which contain only part of the network which is in the largest strongly connected component.

Script consists of the following steps:

* creation of the directed graph when nodes are tuples (edge, lane), and edge is the connection;
* finding the giant connected component (GCC) in this graph (that is, giant strongly connected component in the original network);
* creation of the new connections file which contains only connections which are in GCC;
* creation of the new traffic light file which contains only connections which are in GCC;
* creation of the new edge file which contains only edges and lanes which are in GCC;
* creation of the new node file with nodes only in GCC;
* modifying traffic lights file to remove lights from the removed nodes.

1. Creation of SUMO network file from the modified plain xml files:

netconvert -n area1\_gcc\_plain.nod.xml -e area1\_gcc\_plain.edg.xml -x area1\_gcc\_plain.con.xml -i area1\_gcc\_plain.tll.xml -t plain.typ.xml -o area1\_gcc\_plain.net.xml

netconvert -n area3\_gcc\_plain.nod.xml -e area3\_gcc\_plain.edg.xml -x area3\_gcc\_plain.con.xml -i area3\_gcc\_plain.tll.xml -t plain.typ.xml -o area3\_gcc\_plain.net.xml

1. Creation of the file with parking areas for the new network:

(base) C:\Program Files (x86)\Eclipse\Sumo\tools>python generateParkingAreas.py -n "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area1\_gcc\_plain.net.xml" -o "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area1\_gcc\_parkingareas\_plain.add.xml"

python generateParkingAreas.py -n "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area3\_gcc\_plain.net.xml" -o "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area3\_gcc\_parkingareas\_plain.add.xml"

1. Generation of random demand for the minimum working example:

python randomTrips.py -n "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area1\_gcc\_plain.net.xml" -o "C:\Users\bochenin\RL project\materials\ridesharing\Helsinki\toy example\demand generation\area1\_gcc\_plain.trips.xml"

1. Creation of the route file with taxi trips. As for the new network parking areas may not be created for all of the edges, script genPassengers.py was modified to add the taxi trips only for origins and destinations which have parking areas.

python genPassengers.py -t "demand generation\area1\_gcc\_plain.trips.xml" -pp 50 -pt 30 -pc 2 -pa "demand generation\area1\_gcc\_parkingareas\_plain.add.xml"

python genPassengers.py -t "demand generation\area3\_connected\_sampled.trips.xml" -pp 50 -pt 30 -pc 2 -pa "demand generation\area3\_gcc\_parkingareas\_plain.add.xml"

For the case when we want to generate demand for GCC from the existing route and trip file, step 5 is done with the scripts genGCCtrips.py. The input files for the script are:

* file with GCC (\*.net.xml)
* file with routes from disconnected network (the result of simulation);
* file with trips from disconnected network.

The parameters of the script are:

* last departure time (all trips starting after this time will be cut);
* minimum edges (only subroutes with a number of edges larger that this parameter are included);
* sampling ratio (% of the valid trips to be included in the resulting trip file).

This script generates new trip file containing valid trip for GCC. It searches for valid subroutes from the available trips and calculates new departure time for the subroute assuming that the index of starting edge of the subroute is proportional to its departure time. Approximation of trip time is within \*.rou file (depart and arrival fields) as this route file is the result of the simulation.

python genPassengers.py -t "demand generation\area1\_connected\_sampled.trips.xml" -pp 50 -pt 30 -pc 2 -pa "demand generation\area1\_gcc\_parkingareas\_plain.add.xml"

Hint: after that process some junctions may got stuck as we may reduce the number of lanes. The way to cope with that is the visual inspection and setting junction type to “unregulated” in problematic places.