

Welcome to SUMO 2020



From Traffic Flow to Mobility Modelling

Oct. 26th – 28th
Cyberspace



Knowledge for Tomorrow



SUMO Tutorial

Jakob Erdmann

SUMO2020, Online



Knowledge for Tomorrow



Outline

- Prerequisites
- 3-Click scenario generation with `osmWebWizard.py`
- Network editing
- Creating traffic from counting data
- Taxis



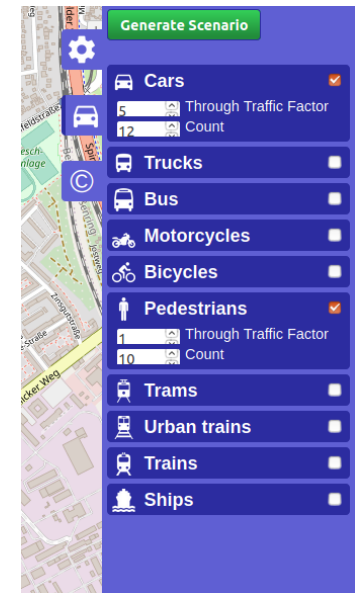
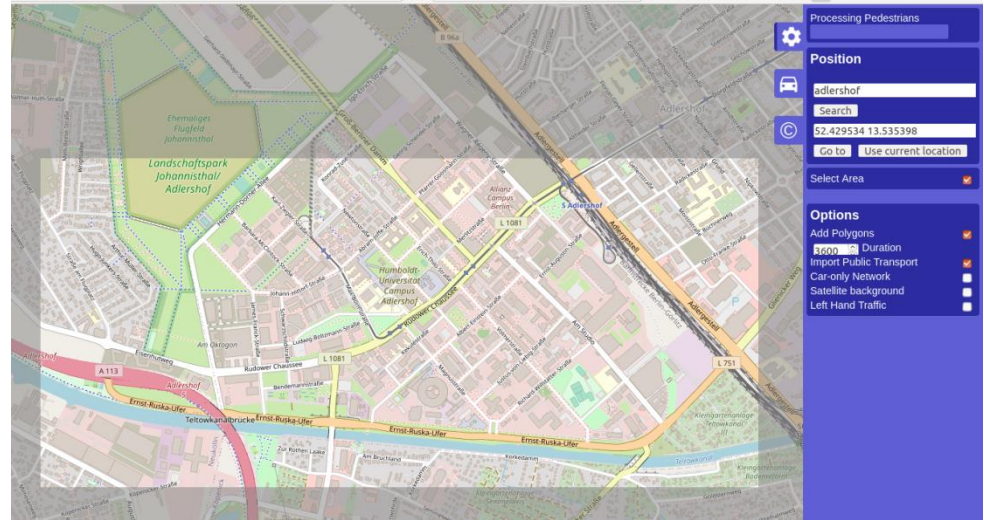
Prerequisites

- SUMO 1.7.0 or latest development version sumo.dlr.de/wiki/Downloads
- Python: www.python.org/download/
- Text Editor (i.e. notepad-plus-plus.org/)
- Data files: sumo.dlr.de/daily/sumo2020_tutorial.zip

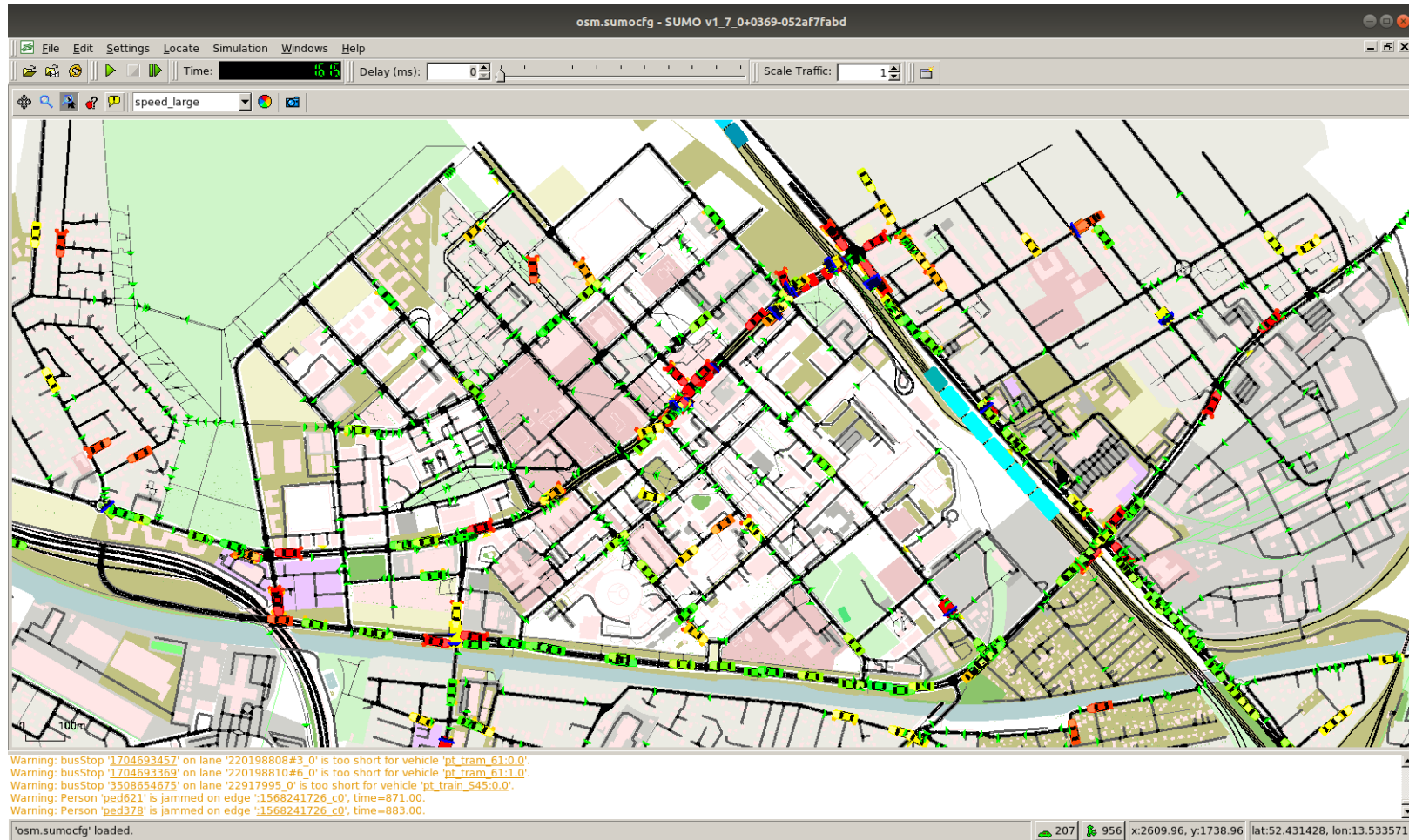


osmWebWizard

- [tools/osmWebWizard.py](https://tools.osmwebwizard.py)
- OpenStreetMap network data
- **Random traffic**
- Configure
 - Area
 - Traffic modes
 - Traffic volume
 - Fraction of through-traffic
 - Public Transport
 - Scenario duration
 - Building Shapes and Points-of-Interest (cosmetic)
 - *Satellite background*
- Generated files allow rebuilding and adapting the scenario
- Example data in 01_wizard



osmWebWizard - Simulation



osmWebWizard - Generated Files

- Scenario input

- `osm.sumocfg`: configuration file (load with **sumo**, **sumo-gui**)
- `osm.net.xml`: simulation network
- `osm.passenger.trips.xml`: passenger cars
- `osm.pedestrian.rou.xml`: persons
- `osm_pt.rou.xml`: busses, trams, ...
- `osm_stops.add.xml`: public transport stop locations
- `osm.poly.xml`: building shapes and POIs
- `osm.view.xml`: sumo-gui settings for delay, colors,...

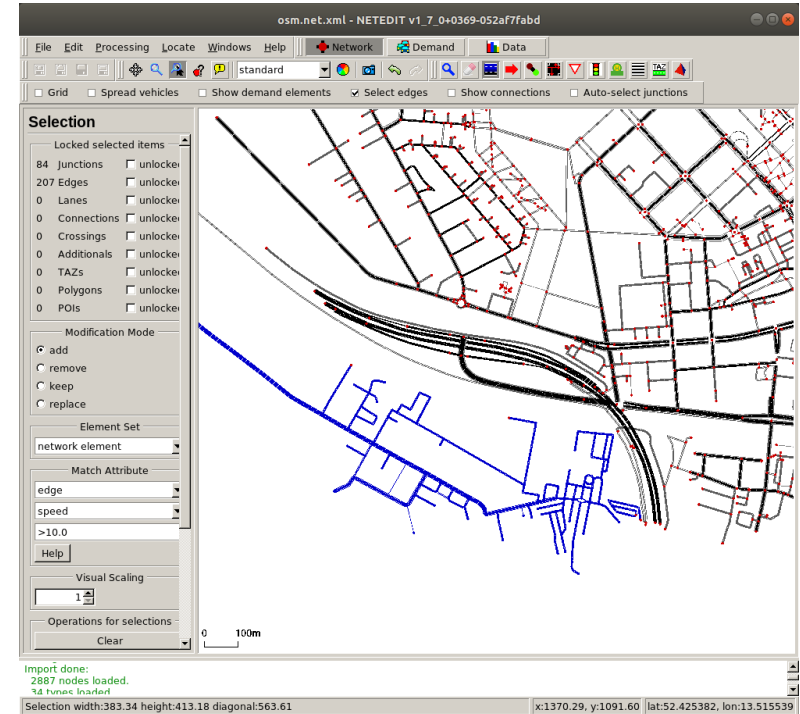
- Rebuilding:

- `osm_bbox.osm.xml`: raw OSM data
- `osm.netccfg`: rebuild network and stops (**netconvert**)
- `osm.polycfg`: rebuild shapes (**polyconvert**)
- `build.bat`: rebuilt traffic (cars, persons, public transport schedule,...)
- `osm_ptlines.xml`: intermediate public transport data



Network Editing - Delete Roads

- Load network osm.net.xml
- Load additional osm_stops.add.xml
- Delete things (i.e. disconnected roads)
 - Select mode (Hotkey: **S**)
 - Shift-click for rectangle selection
 - key to delete
 - F6 cleans up isolated junctions
- Save network, save stops



- Example data in 02_netedit
- run **build.bat** to adapt traffic to changes
 - **On Windows, sumo-gui must be closed!**



Network Editing - Add Turn Lane

- Load network osm.net.xml
- Inspect mode (I)
- (optional) Right-click on edge and split
- Change laneNumber of Edge
- Adjust geometry
- Restore pedestrian crossing
 - Crossing mode (R)
 - Select junction
 - Click on edge to be crossed
 - Enter
- Rebuilding demand not needed (unless edges were split)



Network Editing - Change traffic light type

- Load network `osm.net.xml`
 - Inspect mode (**I**)
 - Change 'tlType' of traffic light junction
 - Check Signal plan in traffic light mode (**T**)
 - Save network
-
- Example data in `03_nedit`



Traffic

- Example scenario traffic has three components
 - Cars: random origin, destination, "fastest" route
 - Public transport (routes, stops, interval from OSM, schedule random)
 - Persons: random origin, destination, "fastest" intermodal route
 - ~3% have missed the bus at the end of service period
- Regenerated car traffic based on local counts
- Replace schedule-based public transport with DRT

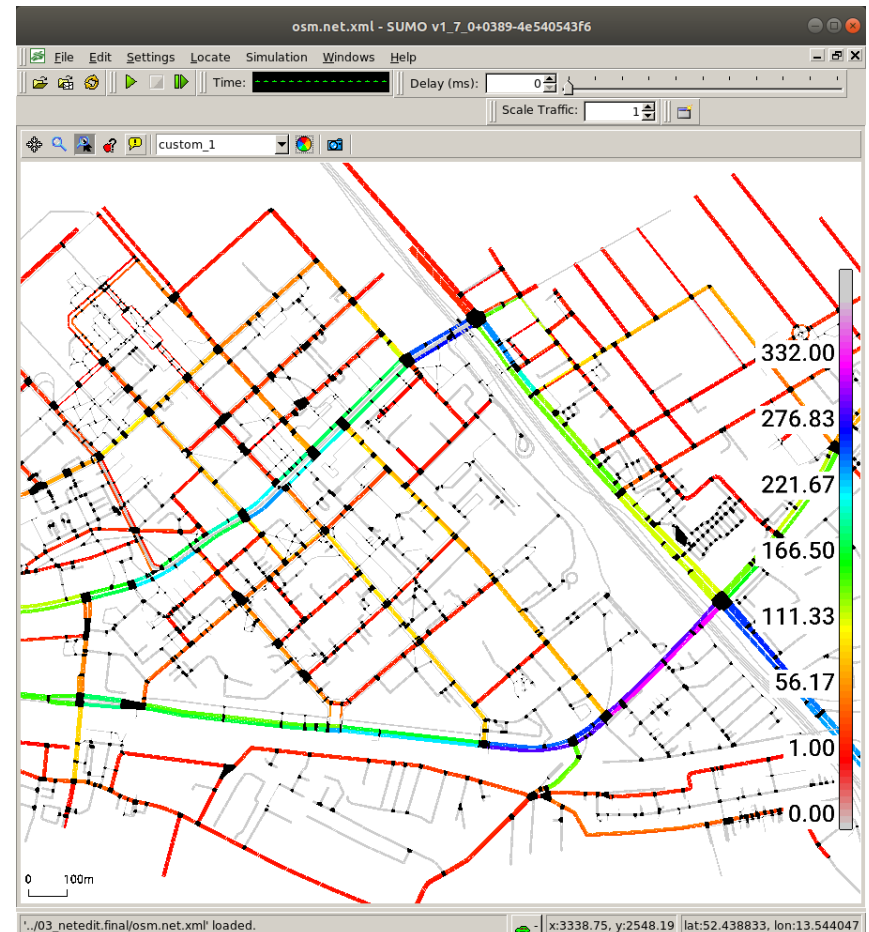
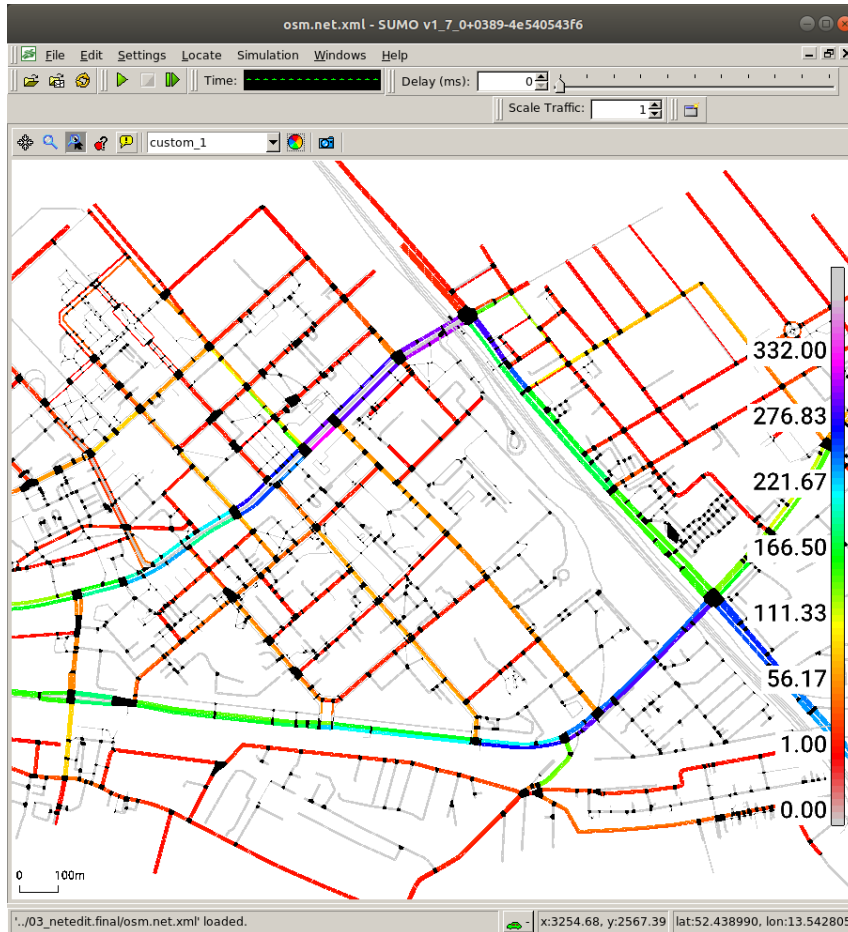


Traffic From Counts

- Input Data: vehicle counts on roads, turn-counts
- Tool: **routeSampler.py** - samples from candidate routes to match counts
 - Compare: **dfrouter** - no control over generated routes
 - Compare: **flowrouter.py** - blacklist undesirable routes
- Define candidate routes with simulation
 - `sumo -c osm.sumocfg --vehroute-output vehroutes.xml --vehroute-output.skip-ptlines`
 - "fastest" routes in a traffic-filled network
- Define counts manually with **netedit** - "Data mode"



Traffic - Fastest routes w/o traffic



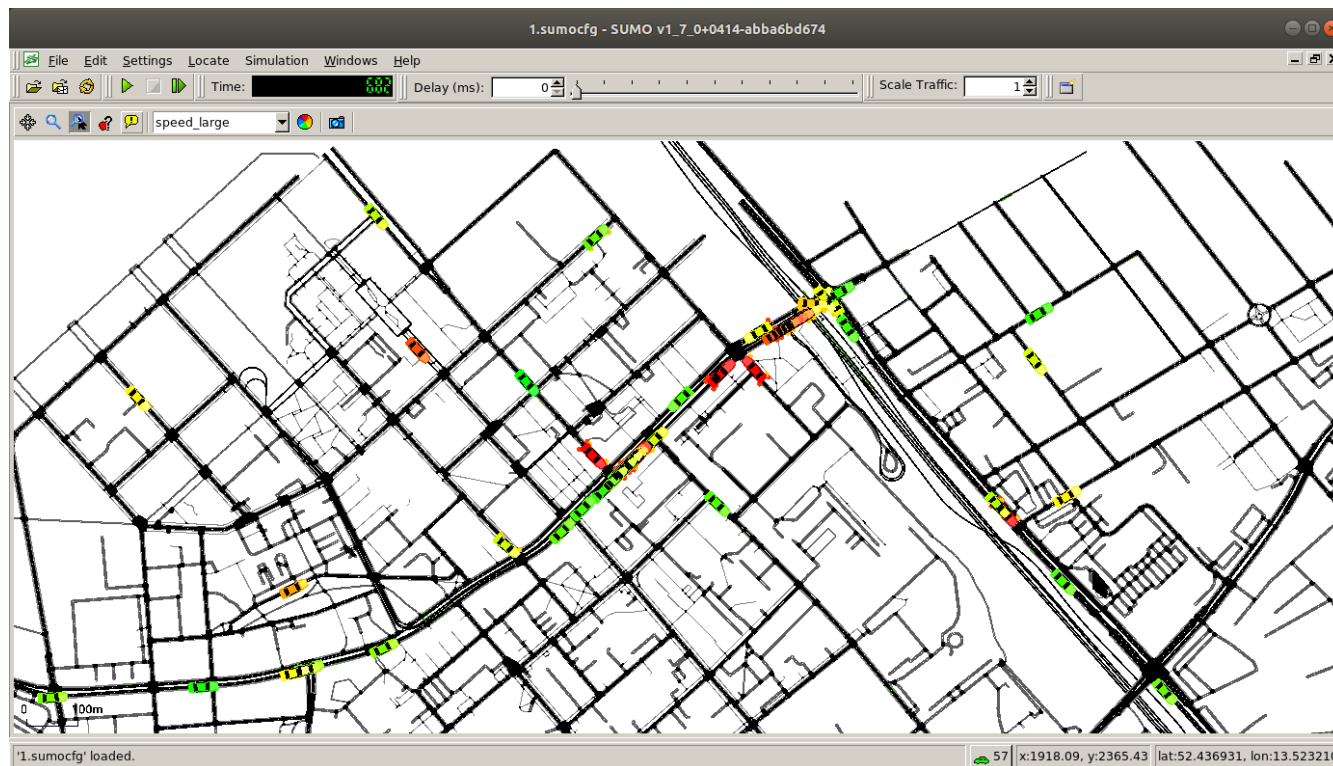
Netedit - Define Edge Counts

- Load `osm.net.xml`
- Enter Data supermode (**F4**)
- Enter edgeData mode (**E**)
- Create new dataSet
- Create new interval
- Define (default) data attribute `"entered=500"`
- Add new edges to dataSet via left-click
- Modify default attribute or edit specific edge attributes in inspect mode (**I**)
- Save Data Elements to file `edgecounts.xml`



routeSampler.py

- `tools/routeSampler.py -r vehroutes.xml --edgedata-files edgecounts.xml -o passenger.sampled.rou.xml`
- Test new traffic by loading `1.sumocfg` in **sumo-gui**



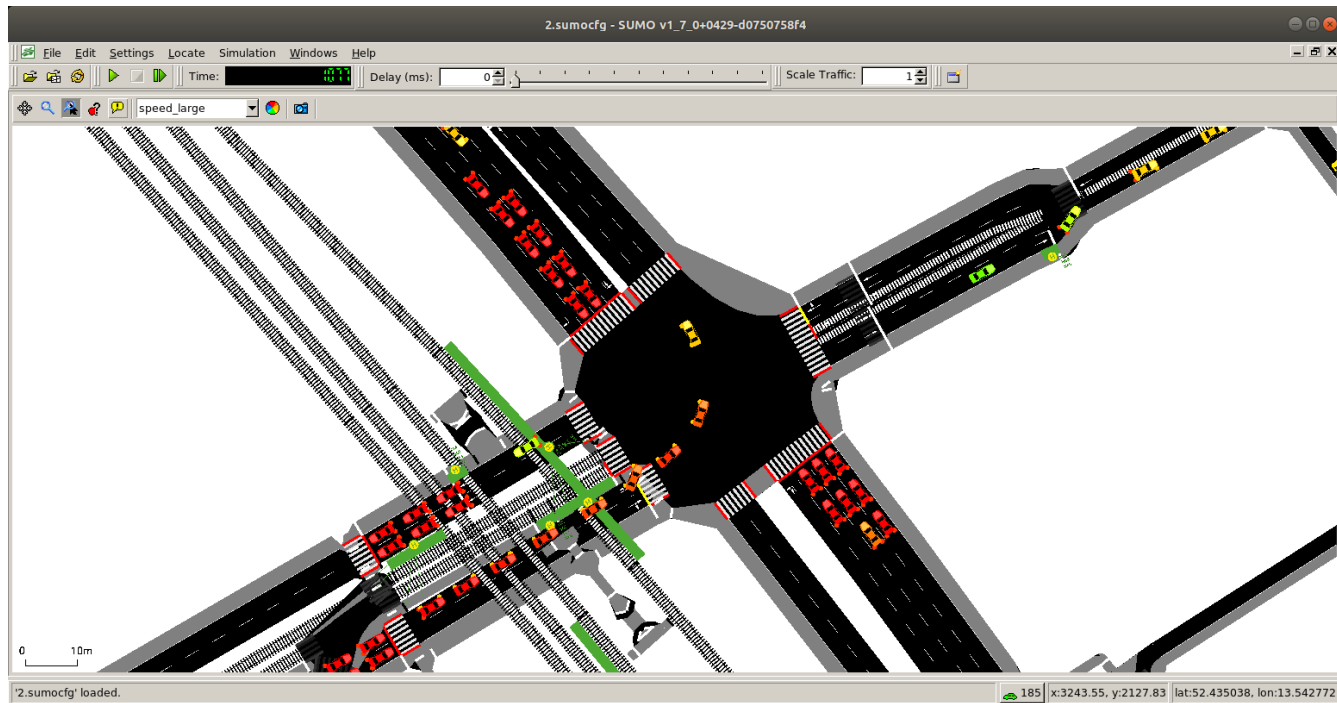
Netedit - Define Turn Counts

- Load `osm.net.xml`
- Enter Data supermode (**F4**)
- Enter edgeRelation mode (**R**)
- Create new dataSet
- Create new interval
- Define (default) data attribute `"count=500"`
- Add new edges to dataSet
 - Click on first edge
 - Click on second edge
 - Enter
- Modify default attribute or edit specific edge attributes in inspect mode (**I**)
- Save Data Elements to file `turncounts.xml`



routeSampler.py - more counting data

- `tools/routeSampler.py -r vehroutes.xml --edgedata-files edgecounts.xml --turn-files turncounts.xml -o passenger.sampled2.rou.xml`
- Test new traffic by loading `2.sumocfg` in **sumo-gui**



Taxis (Demand Responsive Transport)

- Define taxi fleet
 - Define persons that can use taxi service
 - Configure dispatch algorithm
 - Outlook
-
- Note: Taxi in SUMO encompasses all DRT topics (ride-sharing, virtual stations and real stations, ...)
 - Caveat:
 - Taxi features are new and under development
 - Issues were found (and many already fixed) during preparation of this tutorial
 - Tutorial files will work with v1.7.0 by carefully stepping around issues. Upgrading to the development version is recommended when building your own Taxi scenarios



Taxi Fleet

```
<vType id="taxi" vClass="taxi">
  <param key="has.taxi.device" value="true"/>
</vType>
<trip id="t0" type="taxi" depart="0.00" from="e1" to="e1">
  <stop lane="e1_0" triggered="person"/>
</trip>
```

- Define vehicle with taxi device, make sure it stays in the simulation until receiving the first dispatch request
- Can also define many taxis at once

```
<flow id="taxi" type="taxi" begin="0" number="50" period="2">
  <stop lane="e1_0" triggered="person"/>
</flow>
```



Taxis Users

- Intermodal routing should decide between walking and taxi
 - Currently, only criterion is travel time (not price, convenience, ...)
- Random trips that are permitted to use taxi and walking:

```
%sumo_home%\tools\randomTrips.py -n net.net.xml -e 1800  
--persontrips --trip-attributes "modes=\"taxi\""  
-o persontrips.xml
```

- generates 1800 persons with random trips on the given network
- example call in 05_taxi.final\build_taxi_users.bat



Intermodal Routing

- Modelling decision: where are taxis allowed to pick up / drop off customers?
- Possible choice: using existing busStop infrastructure

```
duarouter -n net.net.xml -r persontrips.xml -a stops.add.xml  
--persontrip.transfer.walk-taxi ptStops  
--persontrip.transfer.taxi-walk ptStops  
--ignore-errors -o persons_taxi_ptStops.rou.xml
```

- input trips might be disconnected (network issues) - ignoring filters
- taxi fleet not needed at this point (availability is assumed)
- example config in 05_taxi.final\taxi_ptStops.duarcfg



Running the simulation

- example config in `05_taxi.final\taxi_ptStops.sumocfg`
 - takes almost 4 simulated hours for completion
 - Ride Statistics (1534 rides) WaitingTime: 5328
- more Taxis! (100 instead of 50)
 - `05_taxi.final\taxi_ptStops2.sumocfg`
 - 2 hours simulation until complete, WaitingTime: 2212
- ride sharing (at most 2 persons)
 - `05_taxi.final\taxi_ptStops3.sumocfg`
 - 2 hours simulation until complete, WaitingTime: 2048
- more ride sharing (dev version only)
 - `05_taxi.final\taxi_ptStops4.sumocfg`
 - 2 hours simulation until complete, WaitingTime: 1790
- conclusion



Taxi - Outlook

- Establish SUMO as a platform for Taxi/DRT algorithm tests
- Extend TraCI API to facilitate external algorithm tests (basic support is there)
- Add more example algorithms (re-dispatch while en-route)
- More options for behavior of idle taxis (option `--device.taxi.idle-algorithm` currently supports 'stop', 'randomCircling')
- Distinguish reservation time, scheduled pickup time, pickup time



Taxi - Bloopers / Workarounds

- had to use a network without train- and tramStops (it's complicated)
- removed 4 busStops from the list of available stops (they start or end in a cul-de-sac due to network boundary)
- dispatch algorithms 'greedyClosest' and 'routeExtension' exhibited exciting bugs with the tutorial input
- taxi routing in 1.7 cannot take into account fixed cost of waiting for the taxi to come. Taxi is called even for stepping across the road.
- shared rides in 1.7 fail if the group of people that can share a ride is bigger than the biggest taxi (the group is never picked up).



Bonus - Door to Door Service

- Modelling decision: where are taxis allowed to pick up / drop off customers?
- Possible choice: all roads
- Caveat
 - all roads must be strongly connected to the whole network (hence `connected.net.xml` with some extra edges)
 - needs dev-versio
- example config in `05_taxi.final\taxi_anywhere.duarcfg`
- 100 taxis, no ride sharing
 - `05_taxi.final\taxi_anywhere.sumocfg`
 - over 2 hours simulation until complete, `WaitingTime: 2821`



Conclusion

- Use [tools/osmWebWizard.py](#) to get a quick start
 - Read the documentation / FAQ at <http://sumo.dlr.de/docs>
 - Report any bugs you find to sumo-user@eclipse.org
 - Share your scenarios and results
-
- Talks to us. We are always looking for project partners! sumo@dlr.de

