MaaSSim.

agent-based two-sided mobility platform simulator

Rafał Kucharski r.m.kucharski@tudelft.nl
Oded Cats

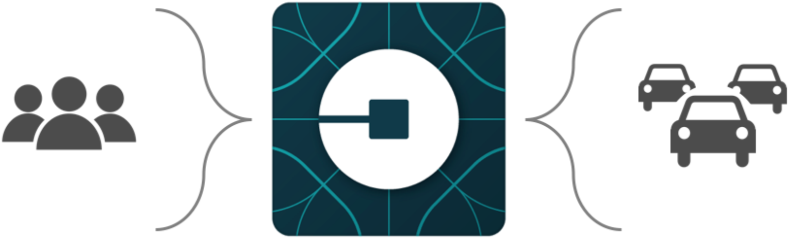
Two-sided platforms

Two-sided mobility platform:

two-sided supply (drivers, vehicles) and demand (travellers)

platform connects supply and demand

mobility offering travellers to supply their mobility needs (reach a destination)



MaaSim

Agent-based two-sided mobility platform simulator

MaaSSim

open source · python · lightweight · agent-based · simulator

The **why's**:

motivation emerging service, disruptive to urban mobility landscape

new to focus on phenomena central to two-sided platforms and not well-studied traffic flow, route choice, congestion, etc. Faster learning curve than well-established full-stack `MatSim, SUMO`, etc.

challenging independent decision makers: heterogenous, individual, adaptive, strategic

complex system dynamics driven by multiple agent classes

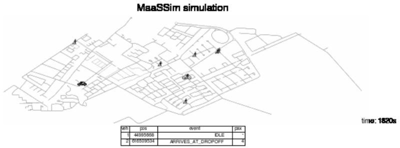
TUDelft

lab Smart Public Transport

MaaSim

https://github.com/RafalKucharskiPK/MaaSSim

an agent-based simulator, reproducing the dynamics of two-sided mobility platforms (like Uber and Lyft) in the context of urban transport networks.



veh	pos	event	pax
1	44995668	IDLE	-
2	616509534	ARRIVES_AT_DROPOFF	4

It models the behaviour and interactions of two kinds of agents:

(i) travellers, requesting to travel from their origin to a destination at a given time, and

(ii) drivers, supplying their travel needs by offering them rides.

The interactions between the two agent types are mediated by the:

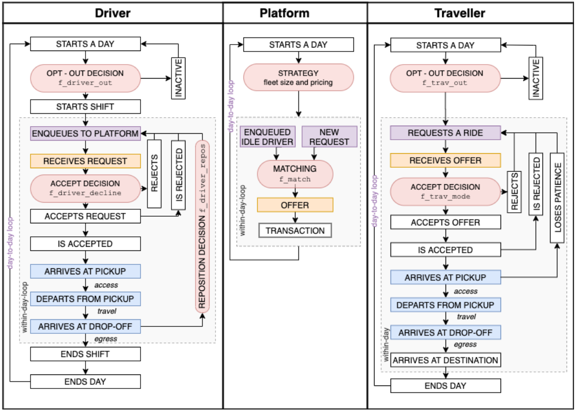
(iii) platform(s), matching demand and supply.

Both supply and demand are microscopic.

```
pip install maassim
```

Kucharski R. and Cats O. *MaaSSim – agent-based two-sided mobility platform simulator*(2020, arxiv.org/pdf/2011.12827)

MaaSSim Agent routines



travellers

- accepting offers,
- selecting platforms and modes,
- leaving the system

drivers

· leaving the system · accepting requests · re-positioning

platform

setting prices matching request

Smart Public Transport

Decisions

Interpretation

travellers

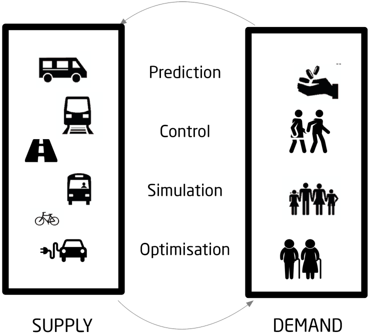
· human behaviour modelling (`discrete choice model`),
· evolution and adaptation (`reinforcement learning`),
· decision support.

drivers

· modelling actual human behaviour
· decision support
· optimal actions (`autonomous vehicles`)

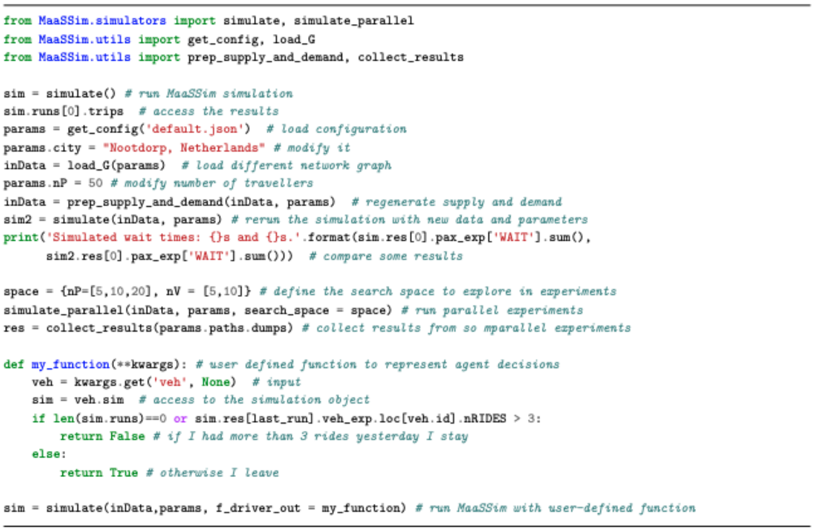
platform

· market actions (`game-theory`)
· distributed system (`control-theory`)



MaaSSim

Usage

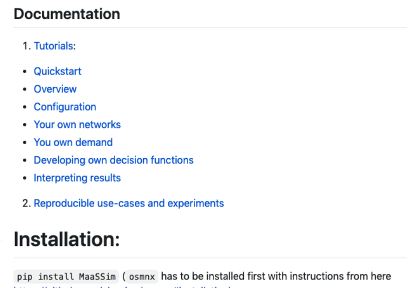


```
from MaaSSim.simulators import simulate, simulate_parallel
from MaaSSim.utils import get_config, load_G
from MaaSSim.utils import prep_supply_and_demand, collect_results

sim = simulate() # run MaaSSim simulation
sim.runs[0].trips  # access the results
params = get_config('default.json')  # load configuration
params.city = "Nootdorp, Netherlands" # modify it
inData = load_G(params)  # load different network graph
params.nP = 50 # modify number of travellers
inData = prep_supply_and_demand(inData, params)  # regenerate supply and demand
sim2 = simulate(inData, params) # rerun the simulation with new data and parameters
print('Simulated wait times: {}s and {}s.'.format(sim.res[0].pax_exp['WAIT'].sum(),
      sim2.res[0].pax_exp['WAIT'].sum()))  # compare some results

space = {nP=[5,10,20], nV = [5,10]} # define the search space to explore in experiments
simulate_parallel(inData, params, search_space = space) # run parallel experiments
res = collect_results(params.paths.dumps) # collect results from so mparallel experiments

def my_function(**kwargs): # user defined function to represent agent decisions
    veh = kwargs.get('veh', None)  # input
    sim = veh.sim  # access to the simulation object
    if len(sim.runs)==0 or sim.res[last_run].veh_exp.loc[veh.id].nRIDES > 3:
        return False # if I had more than 3 rides yesterday I stay
    else:
        return True # otherwise I leave

sim = simulate(inData,params, f_driver_out = my_function) # run MaaSSim with user-defined function
```

MaaSSim

public repository

1. public repository
2. open, short code
3. module, rather than a software
4. tutorial, examples, jupyter notebooks



Documentation

1. Tutorials:

- Quickstart
- Overview
- Configuration
- Your own networks
- You own demand
- Developing own decision functions
- Interpreting results

2. Reproducible use-cases and experiments

Installation:

`pip install MaaSSim` (`osmnx` has to be installed first with instructions from here

https://github.com/RafalKucharskiPK/MaaSSim

Questions

Discussion

Thank you!

Rafał Kucharski,
PostDoc @ Critical MaaS,
Tansportation and Planning TU Delft,
r.m.kucharski@tudelft.nl[1]

pre-print

MaaSSim – agent-based two-sided mobility platform simulator Rafał Kucharski, and Oded Cats arXiv preprint arXiv:2011.12827 (2020) [`http://arxiv.org/abs/2011.12827`]

[1] This research was supported by the CriticalMaaS project (grant number 804469), which is financed by the ERC and Amsterdam Institute of Advanced Metropolitan Solutions.