# Approximation of point equilibria in MATSim

Gunnar Flötteröd

September 5, 2023

LINKÖPING UNIVERSITY vti

# Motivation

- Try to compete with static assignment.
  - Well understood computation process and solution properties.
  - Application (CBA) calls for reproducible, unique solution.
- [Speed. Parallel replanning, minimize number of mobsim runs.]
- [[Support facilities when using MATSim for decision support.]]
- For myself: Figure out if this is worth maintaining for external users.

# Code is here:

`https://github.com/vtisweden/matsim-projects`

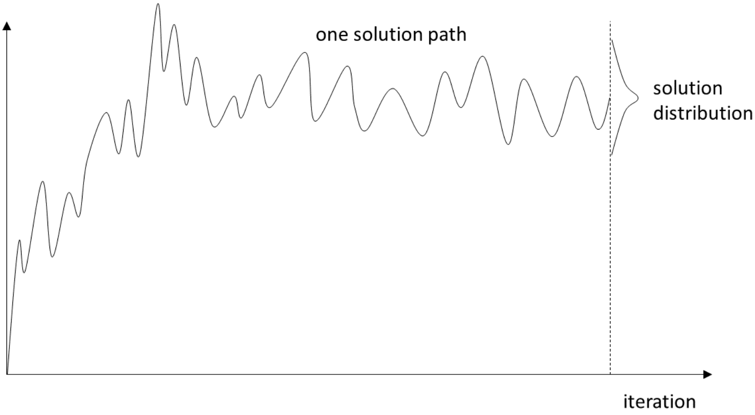
# org.matsim.contrib.emulation

- Move agents according to fixed travel times through the network.
- May be combined with replanning: agents replan less randomly.
- Much more lightweight than running a mobility simulation.
- Versions of this have been around in MATSim for decades.

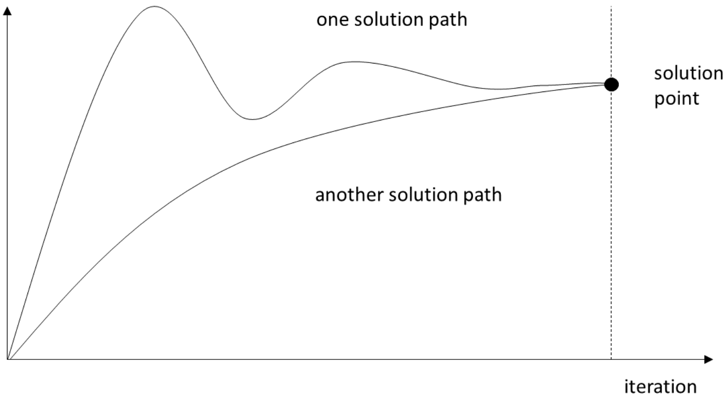
# Emulation is useful

- Play through what-if scenarios.
  - Replanning: discard obviously useless strategies.
  - Optimization (eg station placement): Linearize demand curves.
- Compute distances between populations.
  - Scoring: plan overlap.
  - Replanning: stabilization.
- Allows to compute gap functions that monitor deviation from equilibrium.

# Assignment: stochastic equilibrium



# Deterministic equilibrium



# Approximate deterministic equil.

- Find travel plans that cannot (or hardly) be unilaterally improved.
- Compatible with what MATSim's coevolutionary algorithm looks for.
- Look for point solutions (concrete plans, no distributions thereof).
  - Agents become "greedy" (strict utility maximizers).
  - Minimize "innovation noise" and solution variability.
  - Measurable solution quality (gap function).
- Attempts to get rid of within-assignment randomness.
  - Considers expected plan performance given stochastic mobsim.
  - Choice model error terms: if at all, simulated and freeze.

# org.matsim.contrib.greedo

- Turns MATSim into a point equilbrium approximizer. Balances
  - unilateral utility improvement (emulation-based replanning)
  - and step size (population distance between iterations).
- Details here:

| Session E5 | | | | |
|---|---|---|---|---|
| **Road Transport** | | | | **E9** |
| **Start** | **End** | **ID** | **Authors** | **Title** |
| thu 9:00 | thu 9:30 | 2329 | Felix Hofinger and Martin Fellendorf | Lane change behavior on motorways based on naturalistic trajectory data |
| thu 9:30 | thu 10:00 | 2643 | Gunnar Flötteröd | Improved precision in a heuristic for particle-based and stochastic dynamic traffic assignment |
| thu 10:00 | thu 10:30 | 2677 | Magdalena Schilling, Marvin V. Baumann, Jörg Sonnleitner, Markus Friedrich and Peter Vortisch | Design hourly volume estimation at freeway nodes using floating car data |

# Minimal configuration

```
<module name="emulation">
    <param name="iterationsPerCycle" value="10" />
    ...
</module>

<module name="greedo">
    <param name="replanningRateIterationExponent" value="-1.0" />
    <param name="populationDistance" value="Kernel" />
    <param name="replannerIdentifier" value="UPPERBOUND" />
   ...
</module>
```

# Code example (vanilla carsharing)

```
Greedo greedo = new Greedo();
greedo.setEmulator("twoway_vehicle", NetworkLegEmulator.class);
greedo.setEmulator("access_walk_tw", OnlyDepartureArrivalLegEmulator.class);
greedo.setEmulator("egress_walk_tw", OnlyDepartureArrivalLegEmulator.class);
greedo.setActivityEmulator
    (PlanCalcScoreConfigGroup.createStageActivityType("twoway"),
     OnlyStartEndActivityEmulator.class);
greedo.addHandler(RoadPricingEmulationHandler.class);
[...]
Config config = ...
greedo.meet(config);
[...]
Scenario scenario = ...
Controler controler = new Controler(scenario);
greedo.meet(controler);
[...]
controler.run();
```

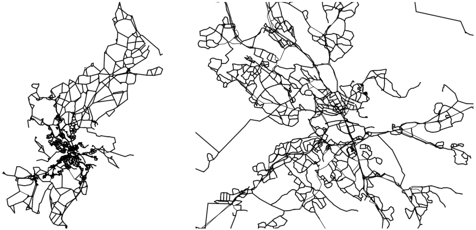
# Code example, continued

```
@Singleton
class GreedoReplanning implements PlansReplanning, ReplanningListener,
AfterMobsimListener

@Inject
GreedoReplanning(Provider<EmulationEngine> emulationEngineProvider, ...)

@Override
public void notifyReplanning(ReplanningEvent event) {
    [...]
    EmulationEngine emulationEngine = emulationEngineProvider.get();
    emulationEngine.setOverwriteTravelTimes(true);
    emulationEngine.emulate(iterationNumber, mode2travelTimes, eventHandler);
    [...]
```

# Stockholm scenario



- $>$20'000 links, $>$5000 agents (1% sample), car-only.
- All-day travel plans. Utility function:
  - penalization of travel times,
  - penalization of deviations from desired arrival time times.
- Better response plans computed by
  - shortest path calculations,
  - random departure time variations.

# Indicative results