Creating a Population

MATSim Population

A minimal MATSim population in XML:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <plan>
      <activity type="home" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" />
      <activity type="work" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" />
      <activity type="shopping" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" />
      <activity type="home" x="890.0" y="685.0" />
    </plan>
  </person>
</population>
```

- A single person with 1 plan.
- Plan contains a list of activities (type, coordinate, end time) and legs (mode).
- Enough to start MATSim.

MATSim Population: PrepareForSim

Such a minimal population is missing some information to be executed:

- link per activity
- routes

MATSim will automatically compute these data when the simulation starts: `PrepareForSim`.

MATSim Population: Full Plan

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE population SYSTEM "http://www.matsim.org/files/dtd/population_v6.dtd">
<population>
  <person id="1">
    <plan score="140.00982053869416" selected="yes">
      <activity type="home" link="1112" x="890.0" y="685.0" end_time="07:43:56" />
      <leg mode="car" dep_time="07:18:16" trav_time="00:01:40">
        <route type="links" start_link="1112" end_link="2223" trav_time="00:31:40" distance="14403.0">1112 1222 2223</r
      </leg>
      <activity type="work" link="2223" x="1802.0" y="2782.0" end_time="17:09:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="2223" end_link="1312" trav_time="00:27:32" distance="12573.0">2223 2313 1312</r
      </leg>
      <activity type="shopping" link="1312" x="1802.0" y="2782.0" end_time="17:42:02" />
      <leg mode="car" dep_time="17:43:36" trav_time="00:05:00">
        <route type="links" start_link="1312" end_link="1112" trav_time="00:04:15" distance="3127.0">1312 1211 1112</ro
      </leg>
      <activity type="home" link="1112" x="890.0" y="685.0" />
    </plan>
  </person>
</population>
```

- plan.score will be set after execution.
- plan.selected marks which plan will be / was executed in the current iteration.

Population vs. Demand

Population: Collection of synthetic persons with attributes:

- home location
- age
- gender
- ...

Demand: Activities and trips performed by agents

- activity types and locations
- activity start- and end-times
- travel mode between activities
- travel route between activities

MATSim stores both population and demand in one file (population.xml, plans.xml) and uses "population" and "demand" without strict differentiation.

Creating a MATSim Population

Write XML by hand

- Only practical for very small test scenarios

Use MATSim API

- Only practical for test scenarios

Convert from existing data

- what existing data?

Creating a MATSim Population

There are no standardized data formats for population or demand data.

MATSim has no standardized algorithms to convert data to a population or to demand.

Population and demand generation is "do-it-yourself" (DIY).

This is often the biggest issue for people new to MATSim.

Creating a MATSim Population: Best Practice

Do-It-Yourself:

1. create a synthetic population (agents) ("population generation")
2. assign demand (plans) to the agents ("demand generation")

Alternative:

1. Use an existing population synthesizer
2. Convert the output to a MATSim population

Creating a MATSim Population: Population Synthesizers

There are several existing software packages for population synthesis.

They all come with their own specific advantages and disadvantages.

Examples:

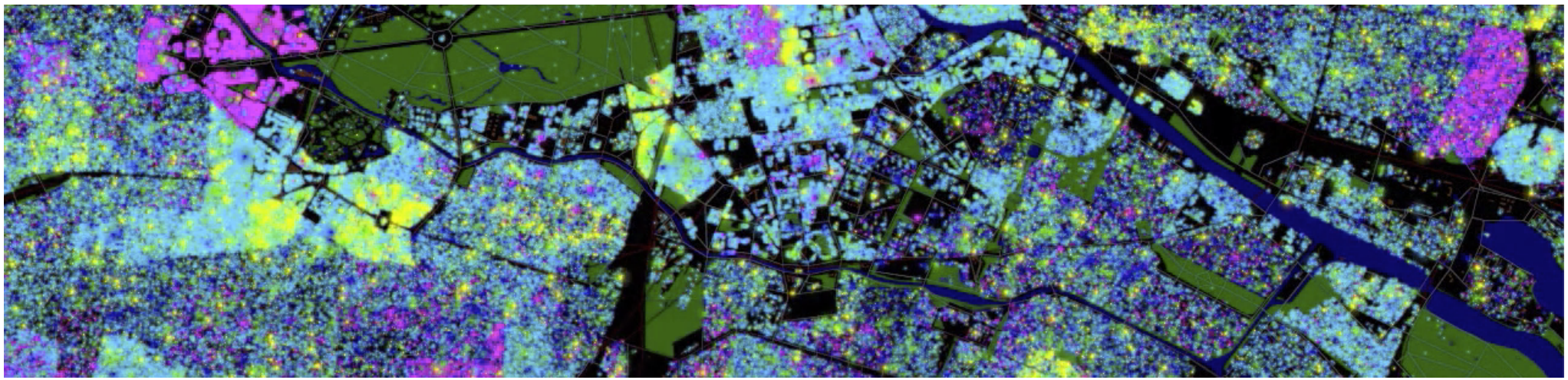
- Albatross
- Cemdap
- TASHA
- ... and many others

There have been some attempts to combine output of some of these with MATSim.

Creating a MATSim Population: What you need

- Population data
  often from census: # persons within a zone with some attributes (age, gender)
- Facility data about Homes, Workplaces, Schools, Shopping facilities, …
  # workplaces per zone, # school places per zone, ...
- Behaviour data: travel diary survey, commuter distances, travel time distributions

If some data set is missing, data could be cleverly made up,
or a random distribution within a zone can be assumed.



https://www.youtube.com/watch?v=rWTFg1UkZTc

Creating a MATSim Population: Input data

Travel Diary Survey

Reported days (activity chains) can be used as template for agents.

Activity chains might differ by region.

different distances, travel times, mode choice, ...

In Germany: MiD (Mobilität in Deutschland), SrV (System repräsentativer Verkehrsbefragungen), srv2018.de

Commuter Distances, Travel Time Distributions

Could be deducted from travel diary survey.

Helps with primary location choice.

Can also be used for model validation / calibration.

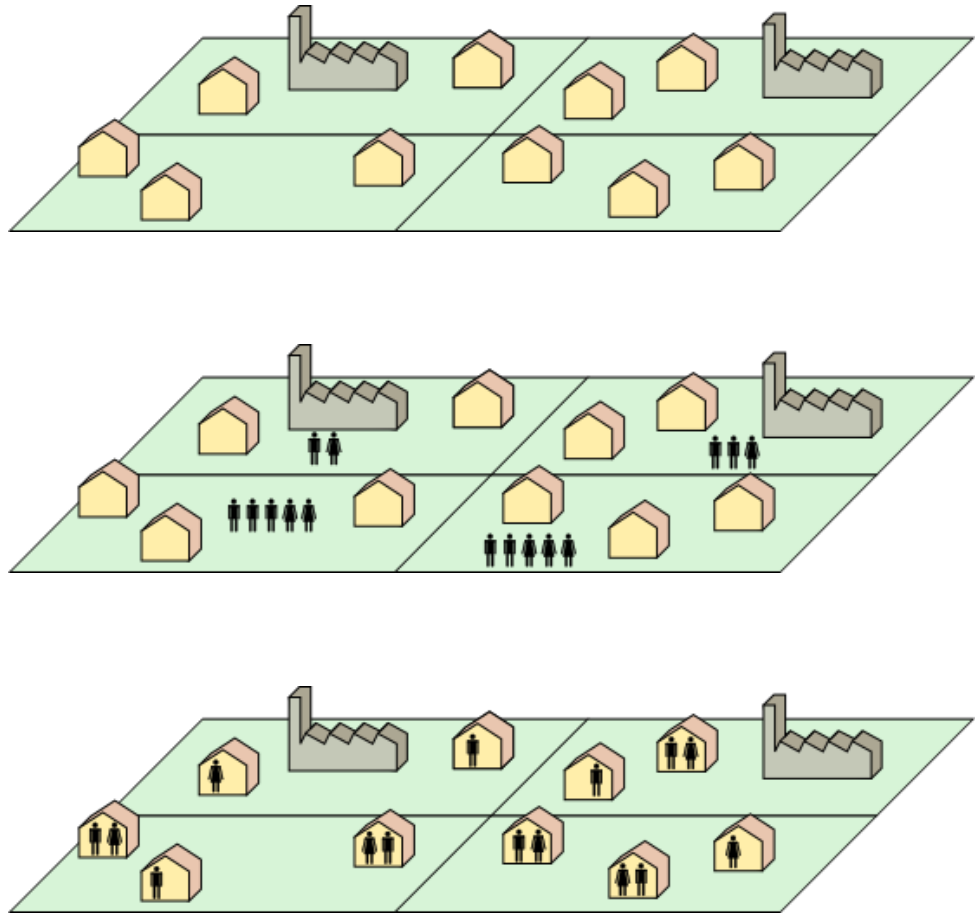
Creating a Population: DIY Best Practice

1. Create facilities per zone,
   either randomly distributed or based on
   detailed data

2. Create group of persons per zone
   with basic demographic attributes

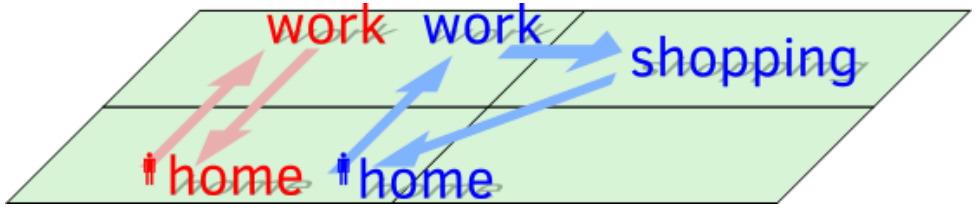
3. Assign persons to home facilities
   within zones

Optionally take households into account.



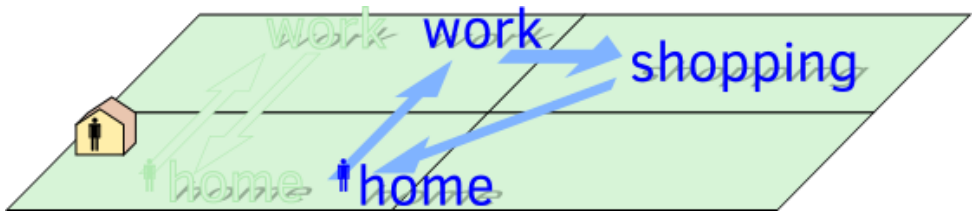
Creating a Demand: DIY Best Practice

1. Think of the travel diaries as agents' plans.



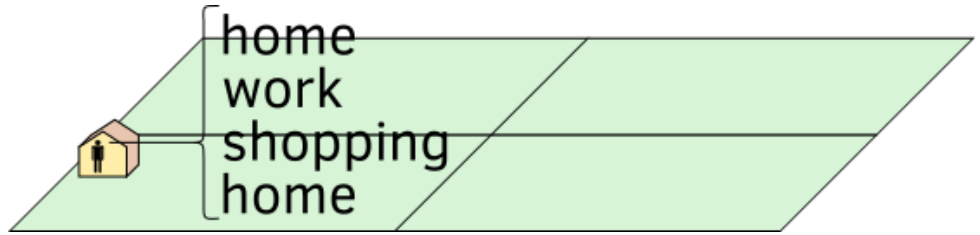
2. For each agent, select a travel diary from survey as template.

   Try to limit the set to make the random selection from, e.g. only select from travel diary respondents with same gender, similar age and similar home zone as agent.



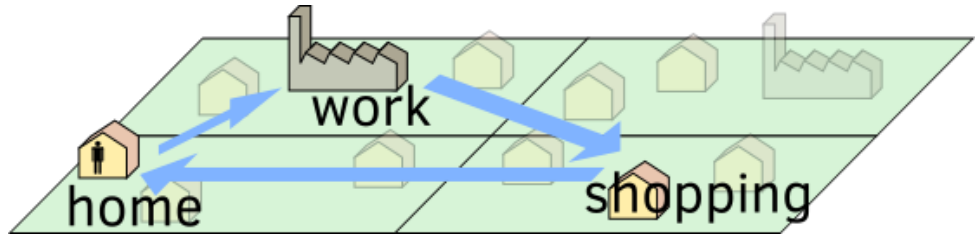
Creating a Demand: DIY Best Practice

3. Copy the activities from the chosen travel diary to the agent.



4. Assign locations (facilities) to the added activities.

   Try to take commuter matrix and/or distances from travel diary into account.



DIY Best Practice: Common Pitfalls

- Randomize / Smear activity end times when copying from template.

  Reported times are often rounded to 10 or 15 minutes.
  Survey usually covers <1% of population. Randomization is needed fur up-scaling.

- Make sure first and last activity are of the same type.

  MATSim has problems scoring plans if first and last activity are of different types.

DIY Best Practice: Getting started

As a general rule:

Start simple, and add more complex elements later.

Examples to start simple:

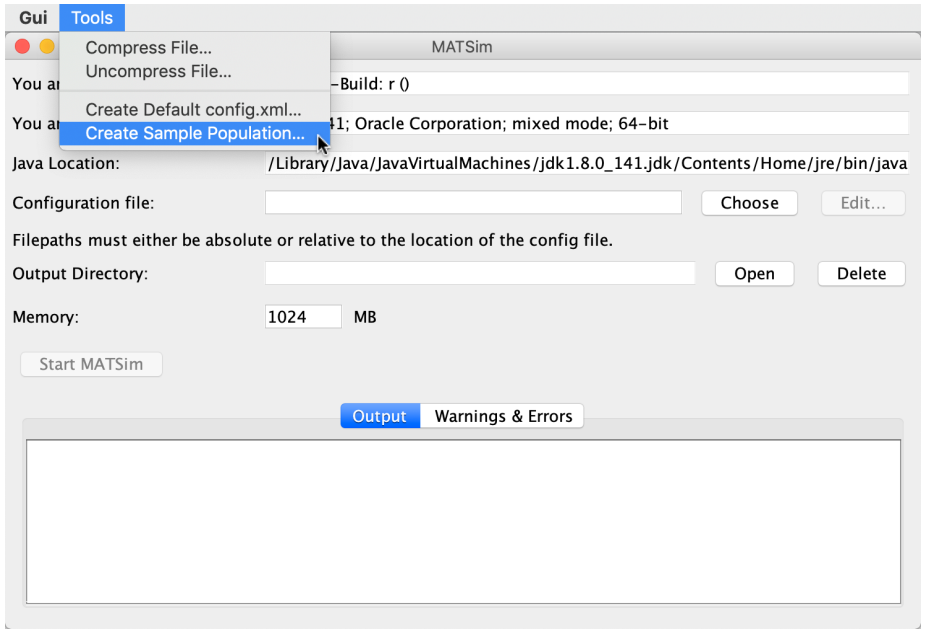
- Instead of polygon-zones, use just single coordinates for zones and use random coordinates around this "center".
- Alternative, use a random node as base for home or work location.
  The idea is that in places where a lot of people live, the network has more nodes.
- Having a population, first add work/education activities only. Add other activity types later.

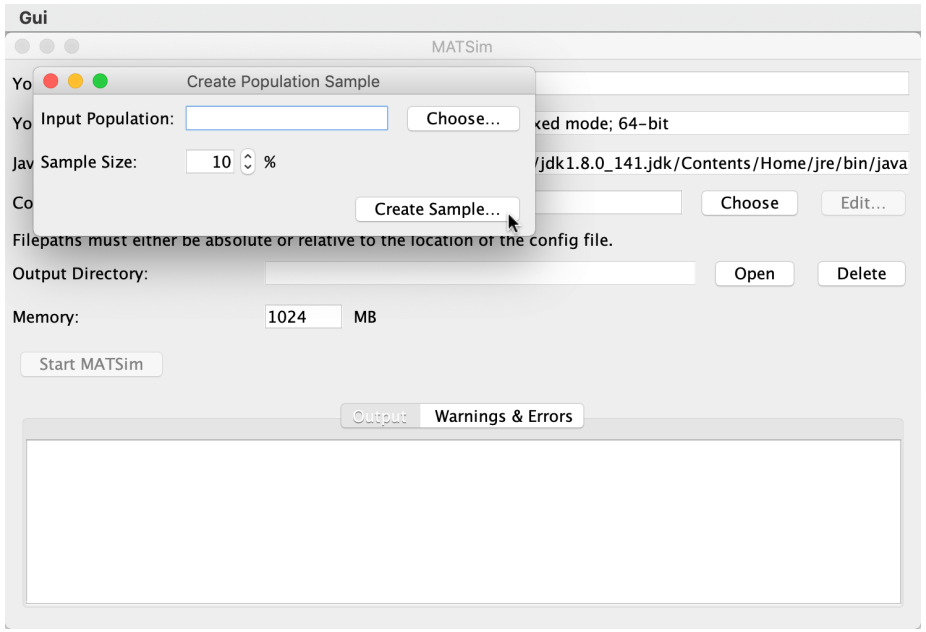
Creating a Population Sample

Simulating a full population requires many resources and much time.

Often, it is enough to simulate 10%, 25% or 50% of the population.
To start a new scenario, it's often best to start with a 1% sample only to get quick feedback.

MATSim Gui → Tools → Create Sample Population…





Simulate a Population Sample

When simulating a population sample, network capacities need to be adapted:

```
<module name="qsim" >
  <param name="flowCapacityFactor" value="0.1" />
  <param name="storageCapacityFactor" value="0.3" />
</module>

<!-- if counts are used -->
<module name="counts" >
  <param name="countsScaleFactor" value="10.0" />
</module>
```

Experience shows that `storageCapacityFactor` should be scaled less than `flowCapacityFactor`. Suggested values based on experience:

- 10%: flowCapacityFactor = 0.1, storageCapacityFactor = 0.3
- 25%: flowCapacityFactor = 0.25, storageCapacityFactor = 0.45
- 50%: flowCapacityFactor = 0.5, storageCapacityFactor = 0.6

(In theory, it would make more sense to scale the vehicles up, e.g. each simulated vehicle represents 10 vehicles in reality. MATSim currently chose the other way: the links are scaled down to let pass e.g. only every tenth vehicle.)

PrepareForSim

PrepareForSim does:

- assign activities to nearest link ("xy2links")
- calculate a route for each leg (network, public transport, teleportation)

xy2links is very basic and makes mistakes:

- assign activities to tunnels or bridges
- assign activities to highways

It's often better to run xy2links manually at the end of the demand modelling with a filtered network.

`org.matsim.run.XY2Links`

`org.matsim.core.population.algorithms.XY2Links`

MATSim Population: PrepareForSim

For large populations, `PrepareForSim` might take a long time (i.e. calculating all routes for the full population!)

MATSim outputs the population at the start of the first iteration in `ITERS/it.0/0.output_plans.xml.gz`.

This file can be re-used as input for other simulations with the same scenario but changed parameters.

Public Transport

Public Transport

Public Transport (PT, transit) in MATSim:

- simulation based on schedule
- transit vehicles run on the network
- transit vehicles have capacities
- transit vehicle serve stops where agents can board and alight
- alternative to teleportation

Public Transport is considered an extension, but is part of the default release.

Public Transport

Current limitations:

- Not frequency-based
  Event for future scenarios, a detailed schedule is required.
- not demand-responsive
- strictly line/route and stop based
  no para-transit
- complex train operations are not supported
  split / join trains, circular lines, board-/aligth-only stops (e.g. night trains)

Required Input

Public transport requires two input files:

transitSchedule.xml

- defines all transit stops
- defines transit services (lines, routes, departures)

transitVehicles.xml

- defines vehicle types and vehicle instances used to run the pt services

transitSchedule.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE transitSchedule SYSTEM "http://www.matsim.org/files/dtd/transitSchedule_v2.dtd">
<transitSchedule>
  <transitStops>
    ...
  </transitStops>

  <minimalTransferTimes>
    ...
  </minimalTransferTimes>

  <transitLine id="...">
    ...
  </transitLine>
  <transitLine id="...">
    ...
  </transitLine>
  ...
</transitSchedule>
```

Transit Stops

Transit Stops require:

- an Id
- coordinates x, y (z is optional)
- a link id on which a vehicle must be to serve this stop

```
<stopFacility id="192" x="05" y="0" linkRefId="1-2" />
```

Optionally, Transit Stops can have:

- a name
- a stop-area id
- flag if a vehicle blocks other traffic (e.g. bus-bay vs bus-stop on road)

```
<stopFacility id="192" x="05" y="0" linkRefId="1-2" name="Austraße" stopAreaId="83" isBlocking="true" />
```

Minimal Transfer Times

Larger stations are often modelled with multiple transit stops,
and require longer transfer times than at smaller stations.

Define the minimal time required to transfer between two stops:

```
<minimalTransferTimes>
    <relation fromStop="192" toStop="255" transferTime="420" />
</minimalTransferTimes>
```

This can also be used to define minimal transfer times at a single stop:

```
<minimalTransferTimes>
    <relation fromStop="192" toStop="192" transferTime="300" />
</minimalTransferTimes>
```

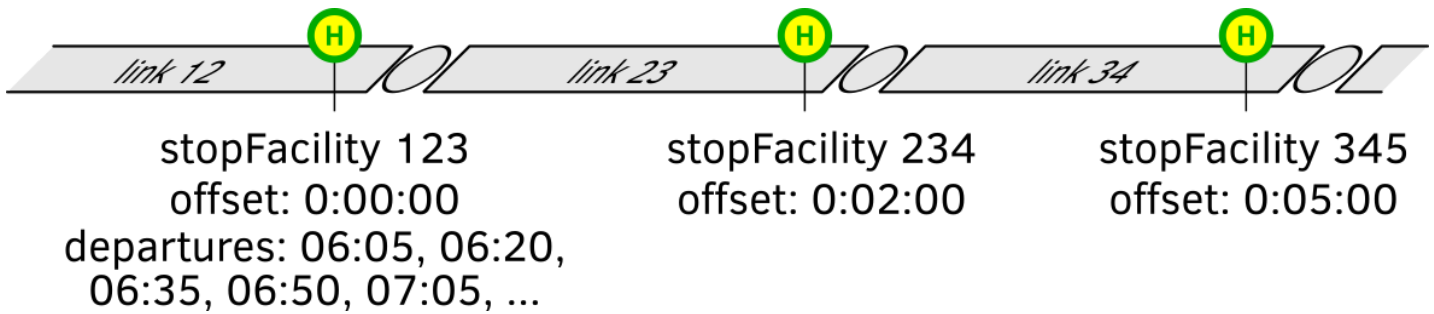
Transfer Times are directed.

The default MATSim pt router does not take stop-specific transfer times into account!

Transit Lines

Transit lines consist of one or more transit routes.
Each transit route

- specifies a mode (e.g. train, bus, tram, ...)
- lists the served stops in order
- lists the links travelled along in the network
- lists all departures at the first stop



Transit Lines

```
<transitLine id="yellow">
  <transitRoute id="yellow_1">
    <transportMode>train</transportMode>
    <routeProfile>
      <stop refId="192" departureOffset="00:00:00" arrivalOffset="00:00:00" awaitDeparture="true"/>
      <stop refId="293" departureOffset="00:00:30" arrivalOffset="00:00:30" awaitDeparture="true"/>
      <stop refId="495" departureOffset="00:30:00" arrivalOffset="00:30:00" awaitDeparture="true"/>
      <stop refId="596" departureOffset="00:30:30" arrivalOffset="00:30:30" awaitDeparture="true"/>
    </routeProfile>
    <route>
      <link refId="1-2"/>
      <link refId="2-3"/>
      <link refId="3-4"/>
      <link refId="4-5"/>
      <link refId="5-6"/>
    </route>
    <departures>
      <departure id="1000" departureTime="05:05:00" vehicleRefId="1000"/>
      <departure id="1001" departureTime="06:05:00" vehicleRefId="1001"/>
      <departure id="1002" departureTime="07:05:00" vehicleRefId="1002"/>
    </departures>
  </transitRoute>
</transitLine>
```

transitVehicles.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<vehicleDefinitions xmlns="http://www.matsim.org/files/dtd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.matsim.org/files/dtd http://www.matsim.org/files/dtd/vehicleDefinitions_v1.0.xsd">

  <vehicleType id="smTr">
    <description>Small Train</description>
    <capacity>
      <seats persons="50"/>
      <standingRoom persons="30"/>
    </capacity>
    <length meter="50.0"/>
    <maximumVelocity meterPerSecond="50.0" />
    <accessTime secondsPerPerson="0.5" />
    <egressTime secondsPerPerson="0.5" />
    <doorOperation mode="serial" />
    <passengerCarEquivalents pce="5.0" />
  </vehicleType>

  <vehicle id="tr_1" type="smTr"/>
  <vehicle id="tr_2" type="smTr"/>

</vehicleDefinitions>
```

Creating a Transit Schedule

Write XML by hand

- Only practical for very small test scenarios

Use MATSim API

- Only practical for test scenarios

Convert from existing data

- Recently, more and more schedules become open-data
- typically as GTFS (General Transit Feed Specification) or HAFAS raw data

  GTFS: Required by Google for Google Maps integration

Creating a Transit Schedule: Convert GTFS

Two alternatives

- pt2matsim
  by Flavio Poletti, ETH Zürich
  also does mapping of routes to network,
  also converts HAFAS data.
  good documenation.
  recommended

- GTFS2MATSim
  by VSP, TU Berlin

Both need to be programmatically called (no GUI).

Creating a Transit Schedule: Network

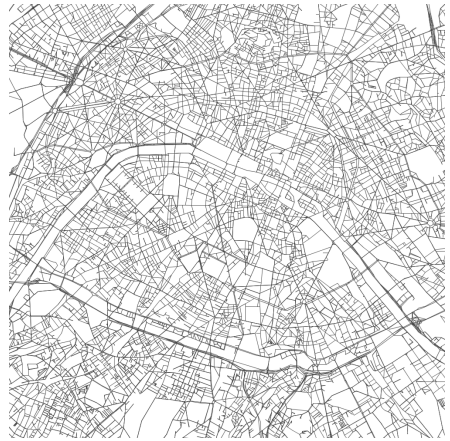
Most schedules only contain information about the location of stops, but not about the routes between stops.

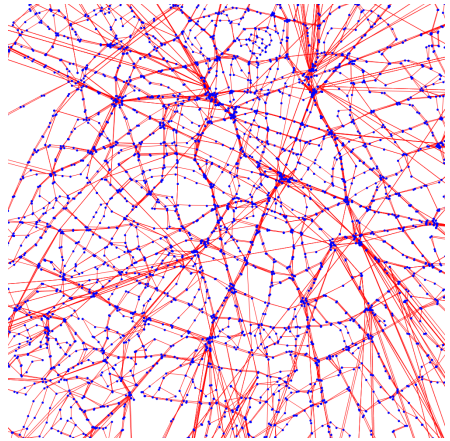
But the simulation needs routes between stops.

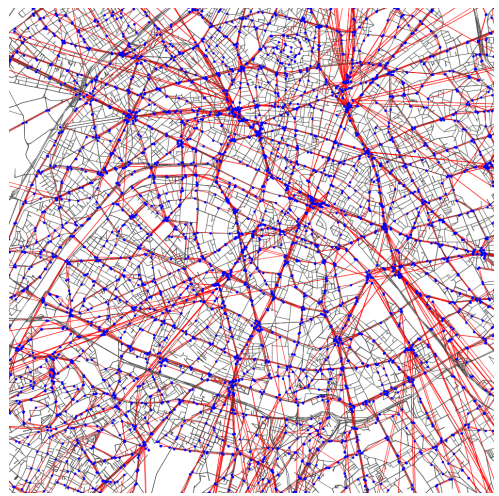
org.matsim.utils.pt.CreatePseudoNetwork

Creates a network with direct links between stops.

This network can then be merged with the road network.





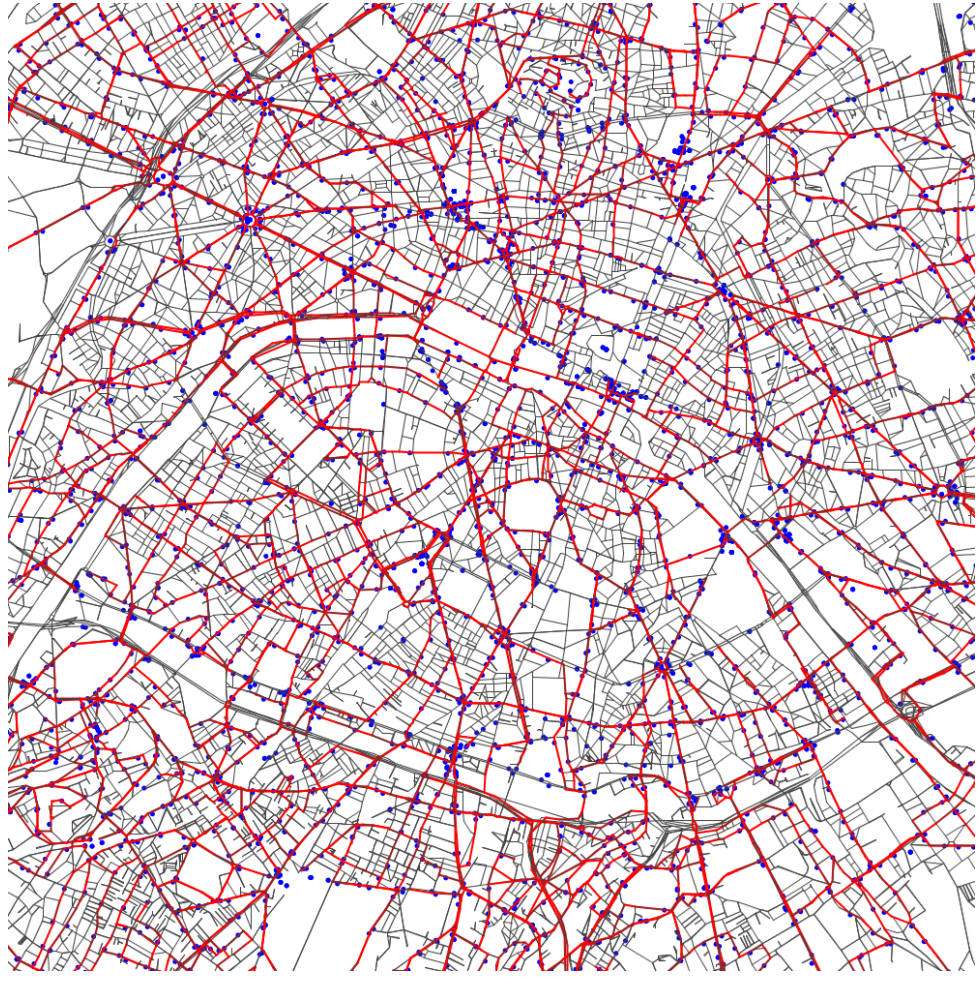


Creating a Transit Schedule: Network

PT2MATSim can try to map transit routes to OSM-networks.

Some artifacts may remain.

But for visualizations it's much improved over the PseudoNetwork.



Creating a Transit Schedule: pt2matsim

```
String fullNetworkFile = "01_ptnetwork.xml.gz";
String unmappedScheduleFile = "01_transitSchedule.xml.gz";
String transitVehiclesFile = "01_transitVehicles.xml.gz";
String mappedScheduleFile = "02_transitSchedule.xml.gz";
String mappedNetworkFile = "02_ptnetwork_mapped.xml.gz";

// create multimodal network
Osm2MultimodalNetwork.run(osmFile, fullNetworkFile, targetCrs);

// convert gtfs schedule
Gtfs2TransitSchedule.run(
    gtfsFile.getAbsolutePath(),
    "20190723", // 2019 July 23
    "EPSG:2154",
    unmappedScheduleFile,
    transitVehiclesFile);


// to be continued
```

Creating a Transit Schedule: pt2matsim (part 2)

```
// now comes the mapping of the schedule to the network
String mappingConfigFile = "01_ptmappingconfig.xml";

// prepare mapping config, mostly copied from org.matsim.pt2matsim.run.CreateDefaultPTMapperConfig
Config config = ConfigUtils.createConfig();
Set<String> toRemove = new HashSet<>(config.getModules().keySet());
toRemove.forEach(config::removeModule); // basically, remove all existing modules

PublicTransitMappingConfigGroup group = PublicTransitMappingConfigGroup.createDefaultConfig();
config.addModule(group); // now only add our special module

group.setInputNetworkFile(fullNetworkFile);
group.setInputScheduleFile(unmappedScheduleFile);
group.setOutputScheduleFile(mappedScheduleFile);
group.setOutputNetworkFile(mappedNetworkFile);

new ConfigWriter(config).write(mappingConfigFile);

// do the mapping
PublicTransitMapper.run(mappingConfigFile);
```

Enabling Public Transport in MATSim

```
<module name="transit" >

  <param name="useTransit" value="true" />

  <param name="transitScheduleFile" value="/path/to/transitSchedule.xml" />
  <param name="vehiclesFile" value="/path/to/transitVehicles.xml" />

  <param name="transitModes" value="pt" />

</module>
```

matsim-sbb-extensions

SBB (Swiss Federal Railways) provides some pt-related extensions to MATSim.
github.com/SchweizerischeBundesbahnen/matsim-sbb-extensions/

Deterministic Simulation of PT

An alternative public transport simulation where pt vehicles always run according to the schedule.

SwissRailRaptor

An alternative, fast pt router with highly reduced memory usage.

The use of SwissRailRaptor is highly recommended.
There are plans to make it the default pt router in MATSim in the future.

Skims

Code to calculate skim matrices (origin-destination performance indices).

Deterministic PT Simulation

The default "network-based" simulation of PT has some drawbacks, especially for railways.

1. The modelled level of detail differs between roads and rails.
    - Road-Network: links = road-segments, nodes = intersections
    - Rail-Networks: links ≠ rails, nodes ≠ switches

    links ≈ generalized connection between stops, one or more rails

2. Cars usually drive close to max. speed allowed, Trains don't.
   Trains have time slots and drive as fast as necessary to stay in slot.

3. Having parallel rails, trains can overtake each other. But not on MATSim links.

Deterministic PT Simulation

If trains are simulated on network links:

- trains will drive with max. speed
- max speed must be set to accommodate the fastest train connection
- most trains will arrive too early at next stop
- the fastest train can not drive faster to reduce a delay

Assuming trains run on schedule, this could not be modelled in MATSim due to too many simulation artifacts by simulating on network links.

Deterministic PT Simulation

The deterministic PT simulation teleports pt vehicles according the schedule from stop to stop.

It does not simulate the pt vehicles on links.
It can create link-events for pt vehicles for analysis and visualization purposes.

One can configure that only trains should be simulated deterministically, but others (e.g. buses) should still be simulated on links.

SwissRailRaptor

A much faster routing algorithm for transit schedules than the default implementation.

Speed-up depends on the scenario.
On large scenarios, routing speed-ups of up to 100x were observed while reducing memory consumption to 10%.

Can act as a replacement for the default pt router.

Provides additional features for advanced use cases.

Takes stop-specific minimal transfer times into account.

Using matsim-sbb-extensions

The extension provides detailed information how to use it:
github.com/SchweizerischeBundesbahnen/matsim-sbb-extensions/

Add the extensions as a dependency to `pom.xml`:

```
<dependency>
  <groupId>com.github.SchweizerischeBundesbahnen</groupId>
  <artifactId>matsim-sbb-extensions</artifactId>
  <version>11.2</version>
</dependency>
```

Using matsim-sbb-extensions

Add it to the controler:

```
Controler controler = new Controler(scenario);

// To use the fast pt router:
controler.addOverridingModule(new SwissRailRaptorModule());

// To use the deterministic pt simulation:
controler.addOverridingModule(new SBBTransitModule());
controler.configureQSimComponents(components -> {
    SBBTransitEngineQSimModule.configure(components);
    // if you have other extensions that provide QSim components, call their configure-method here.
});

controler.run();
```

Important note: The Controler and QSim get regularly modified to better integrate new features. In the next version of MATSim, the code above will no longer work.
Check out the up-to-date integration in the code repository: RunSBBExtension.java

Sample Simulations

Take care when running population samples for simulation:

10% private cars vs. 100% pt vehicles.

It does not make sense to run 10% of pt vehicles.
Instead, the pt vehicles should be scaled:

```
<vehicleType id="smTr">
  <description>Small Train</description>
  <capacity>
    <seats persons="5"/>
    <standingRoom persons="3"/>
  </capacity>
  <length meter="50.0"/>
  <maximumVelocity meterPerSecond="50.0" />
  <accessTime secondsPerPerson="5.0" />
  <egressTime secondsPerPerson="5.0" />
  <doorOperation mode="serial" />
  <passengerCarEquivalents pce="0.5" />
</vehicleType>
```

Replanning and Scoring

Replanning and Scoring

Replanning

Variation of the agents' plans set.

Scoring

Evaluation of the executed plan.

The scoring should react to differences introduced by the replanning, otherwise the agents cannot optimize certain aspects of their plans.

Replanning

1. Select one of the existing plans ("PlanSelector")
2. Make a copy of the plan
3. Modify the plan copy ("PlanStrategy")

In MATSim's config, replanning strategies are specified by names.

Replanning Strategies

Route Choice

- ReRoute *****

Departure Time Choice

- TimeAllocationMutator
- TimeAllocationMutator_ReRoute *****

Mode Choice

- ChangeSingleTripMode
- ChangeTripMode
- SubtourModeChoice *****

Plan Selection only

- SelectRandom
- BestScore
- KeepLastSelected
- ChangeExpBeta *****
- SelectExpBeta
- SelectPathSizeLogit

***** recommended strategies

Random mutation vs. Best-Response

Some strategies perform random mutation (e.g. TimeAllocationMutator, SubtourModeChoice), while other strategies apply a best-response mutation of the plan (e.g. ReRoute).

Best-Response:

- might find good plans faster
- might not create enough variation
- often suggests global perfect knowledge by agents

Random mutation:

- might get stuck in local optimum
- might force agents to try bad plan variations
- requires more iterations to find good plans

Ongoing research to find better strategies that combine advantages of both types.

Configuring Replanning

```
<module name="strategy" >
  <param name="maxAgentPlanMemorySize" value="3" />
  <param name="fractionOfIterationsToDisableInnovation" value="0.9" />

  <parameterset type="strategysettings">
    <param name="strategyName" value="ChangeExpBeta" />
    <param name="weight" value="0.7" />
  </parameterset>

  <parameterset type="strategysettings">
    <param name="strategyName" value="ReRoute" />
    <param name="weight" value="0.1" />
  </parameterset>

  <parameterset type="strategysettings">
    <param name="strategyName" value="TimeAllocationMutator_ReRoute" />
    <param name="weight" value="0.1" />
  </parameterset>

  <parameterset type="strategysettings">
    <param name="strategyName" value="SubtourModeChoice" />
    <param name="weight" value="0.1" />
    <param name="disableAfterIteration" value="70" />
  </parameterset>

</module>
```

Configuring Replanning: Global Settings

- **maxAgentPlanMemorySize**
  size of agents' choice set, typically between 3 and 5

- **fractionOfIterationsToDisableInnovation**
  Replanning can force agents to try out bad plans, which becomes a problem when the choice set is already heavily optimized. Disable innovation for the last few iterations to eliminate such artefacts.

- **planSelectorForRemoval**
  Defaults to "WorstPlanSelector" for evolutionary algorithm. Might result in agents' choice sets' plans being very similar to each other.

Route Choice

ReRoute

Calculates a route for each trip, depending on the mode and configuration.

- **Routing on network**
  Default for "car".
  Using time-dependent least-cost-path algorithms like Dijkstra's or A* with Landmarks.

- **Public Transport Schedule Routing**
  Default for "pt" if transit is enabled.
  Using a variant of Dijkstra's shortest-path algorithm by default (very slow).
  Alternative, faster implementation based on RAPTOR algorithm available.

- **Teleportation**
  Default for "walk", "bike".
  Travel time calculated either based on bee-line distance and average speed,
  or based on scaled freespeed travel time.

Configuring Route Choice

```
<module name="planscalcroute" >

  <!-- Network routing -->
  <param name="networkModes" value="car" />

  <!-- Teleportation -->
  <parameterset type="teleportedModeParameters" >
    <param name="mode" value="bike" />

    <param name="beelineDistanceFactor" value="1.3" />
    <!-- Travel time = (<beeline distance> * beelineDistanceFactor) / teleportedModeSpeed. -->
    <param name="teleportedModeSpeed" value="4.1666667" />
  </parameterset>

  <parameterset type="teleportedModeParameters" >
    <param name="mode" value="walk" />

    <param name="beelineDistanceFactor" value="1.3" />
    <param name="teleportedModeSpeed" value="0.8333333333333333" />
  </parameterset>

  <parameterset type="teleportedModeParameters" >
    <param name="mode" value="pt" />

    <!-- Using a free-speed factor. Travel time = teleportedModeFreespeedFactor * <freespeed car travel time>. -->
    <param name="teleportedModeFreespeedFactor" value="2.0" />
  </parameterset>
</module>
```

Network Routing

Find the fastest route through a network based on travel times from the last iteration.

Different routing algorithms for path-finding on the network are available:

- Dijkstra
- FastDijkstra
- AStarLandmarks
- FastAStarLandmarks

AStarLandmarks requires a little initialization time but is usually faster than Dijkstra.
The Fast*-variants are faster than the regular variants, but use more memory.

```
<module name="controler" >
  <!-- The type of routing (least cost path) algorithm used, may have the values:
  Dijkstra, FastDijkstra, AStarLandmarks or FastAStarLandmarks -->
  <param name="routingAlgorithmType" value="AStarLandmarks" />
</module>
```

Recommended: `AStarLandmarks`

Teleportation

Used to model alternative modes that do not need full details.
By default used for "walk" and "bike".

Agents with teleporting routes will "vanish" upon departure and "reappear" some time later at their destination.

Configuring Teleportation

Two kinds how teleportation travel time can be calculated:

- based on beeline distance
  e.g. "3 km/h with the estimated distance being 1.3 times the beeline distance."

```
<parameterset type="teleportedModeParameters" >
<param name="mode" value="walk" />

<param name="beelineDistanceFactor" value="1.3" />
<param name="teleportedModeSpeed" value="0.8333333333333333" />
</parameterset>
```

- based on freespeed travel times in network
  e.g. "it takes twice the time it would take in an empty network."

```
<parameterset type="teleportedModeParameters" >
<param name="mode" value="pt" />

<!-- Using a free-speed factor. Travel time = teleportedModeFreespeedFactor * <freespeed car travel time>. -->
<param name="teleportedModeFreespeedFactor" value="2.0" />
</parameterset>
```

Public Transport Routing

Public transport needs to be enabled:

```
<module name="transit" >
  <param name="useTransit" value="true" />
  <param name="transitModes" value="pt" />
</module>
```

Otherwise, pt legs will be teleported by default.

Departure Time Choice

TimeAllocationMutator

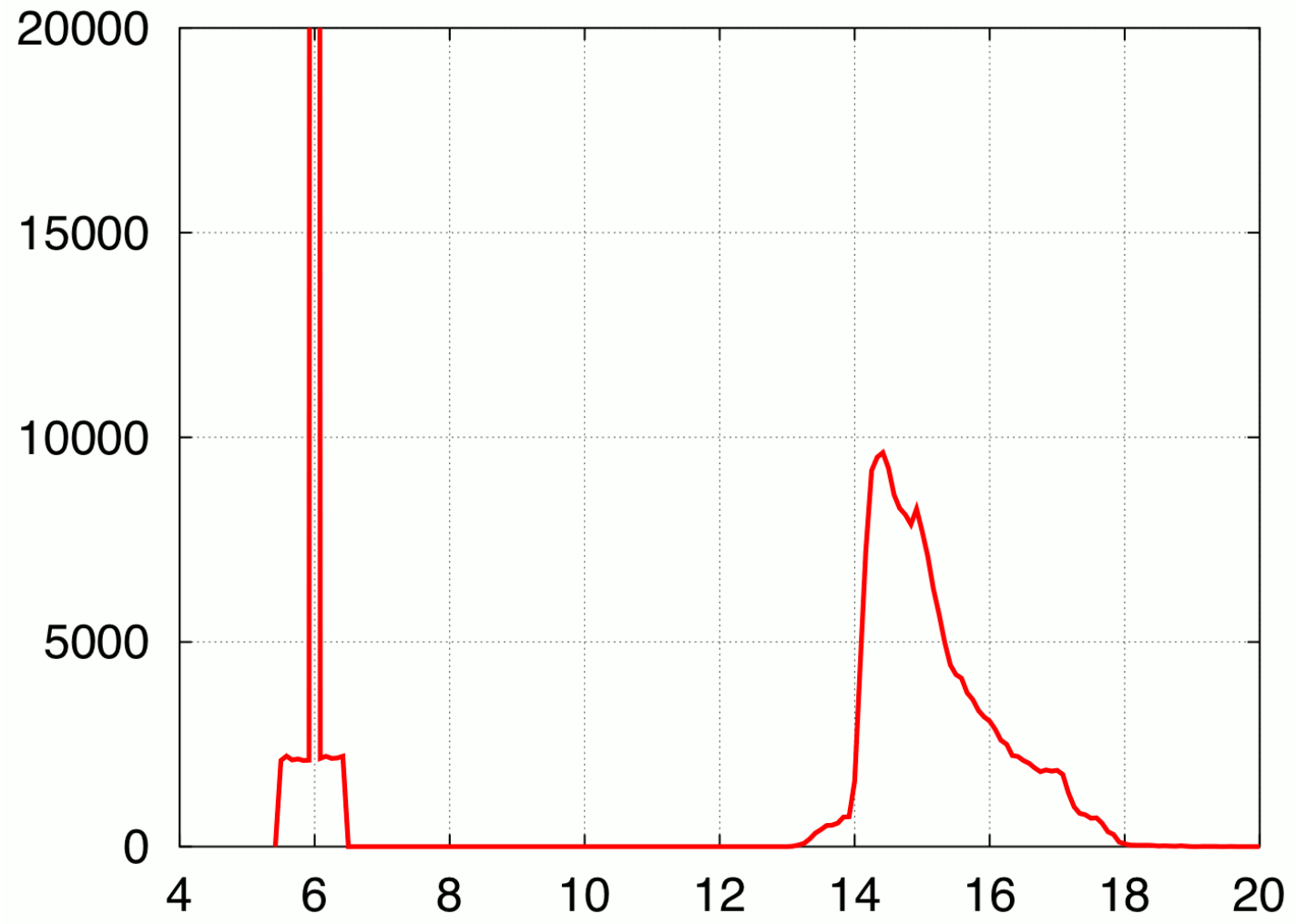
Randomly changes the activity end time (default ± 30min), can be configured in config.xml:

```
<module name="TimeAllocationMutator" >
  <param name="mutationRange" value="1800.0" />
</module>
```

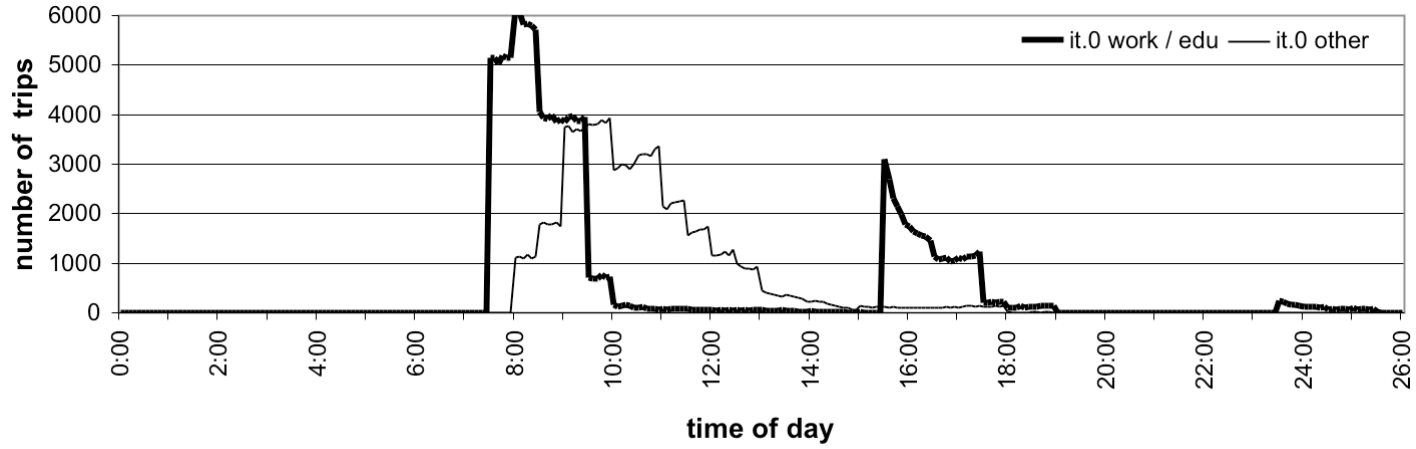
When using public transport (and in some other cases), it's advisable to ReRoute after changing the departure time. Use the strategy `TimeAllocationMutator_ReRoute` in that case:

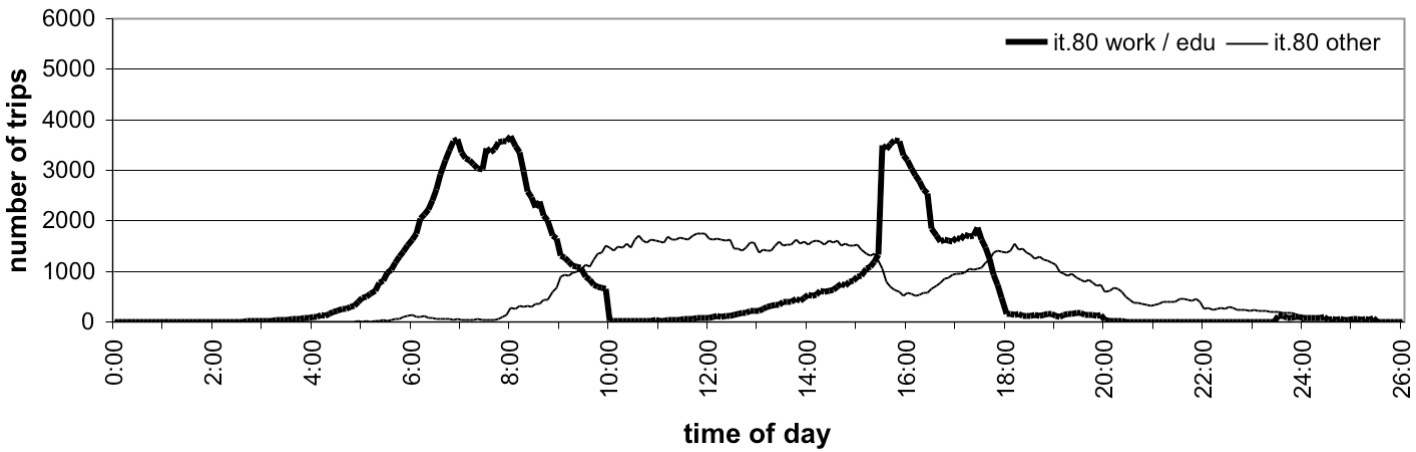
```
<module name="strategy" >
  <parameterset type="strategysettings">
    <param name="strategyName" value="TimeAllocationMutator_ReRoute" />
    ...
```

Departure Time Choice: Example



Departure Time Choice: Example





Rieser et al. 2007. *Truly agent-oriented coupling of an activity-based demand generation with a multi-agent traffic simulation*

Mode Choice

Different strategies available:

- **ChangeSingleTripMode**
  Changes the mode of a single trip in the plan.

- **ChangeTripMode**
  Changes the modes of all trips. Different modes for different trips are possible.
  No "mass conservation" (e.g. car, pt, car is a possible outcome).

- **SubtourModeChoice**
  Analyses the plan for sub-tours, assignes modes to all legs in each sub-tour.
  Respects mass conversation.
  Optionally takes car-availability into account.

Mode Choice and Car Availability

Default: Persons **not** allowed to use a car should have one of the two attributes:

- carAvail = "never",
- hasLicense = "no"

These attributes currently need to be stored "externally".

Option to respect car availability needs to be enabled in configuration.

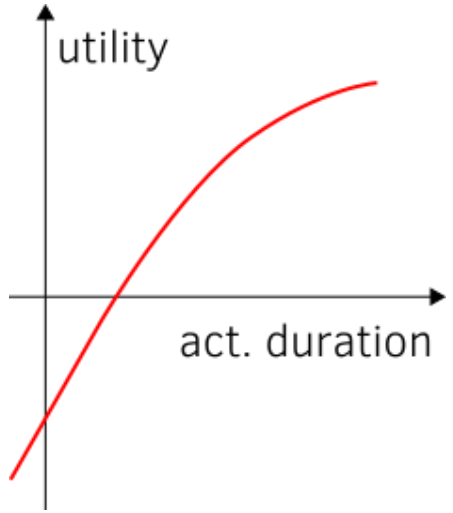
Scoring

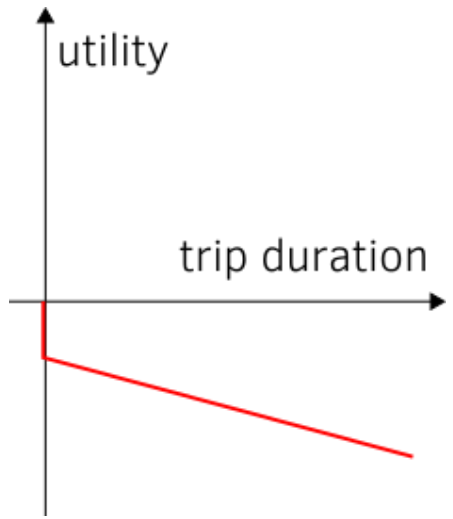
Scoring is responsible to evaluate the performance of the executed plans.

Terms:

- Positive utility for performing activities
- Negative utility for travelling
- Negative utility for monetary costs (tolls, fares, …)
- Negative utility for arriving late, leaving early, …
- Negative utility for monetary costs

Sum up all terms for each agent over day.





Scoring: config.xml

```
<module name="planCalcScore" >
  <parameterset type="scoringParameters">
    <param name="earlyDeparture" value="-0.0" />
    <param name="lateArrival" value="-18.0" />
    <param name="marginalUtilityOfMoney" value="1.0" />
    <param name="performing" value="6.0" />
    <param name="subpopulation" value="null" />
    <param name="utilityOfLineSwitch" value="-1.0" />
    <param name="waiting" value="-0.0" />
    <param name="waitingPt" value="-6.0" />

    <parameterset type="activityParams">
      ...
    </parameterset>
    <parameterset type="activityParams">
      ...
    </parameterset>

    <parameterset type="modeParams">
      ...
    </parameterset>
    <parameterset type="modeParams">
      ...
    </parameterset>

  </parameterset>
  <parameterset type="scoringParameters">
    ... parameters for a (different) subpopulation
  </parameterset>
</module>
```

Scoring: config.xml: activityParams

```
<parameterset type="activityParams">
  <param name="activityType" value="work" />

  <param name="openingTime" value="06:00:00" />
  <param name="closingTime" value="20:00:00" />

  <param name="latestStartTime" value="undefined" />
  <param name="earliestEndTime" value="undefined" />

  <param name="minimalDuration" value="03:00:00" />
  <param name="typicalDuration" value="08:00:00" />
</parameterset>
```

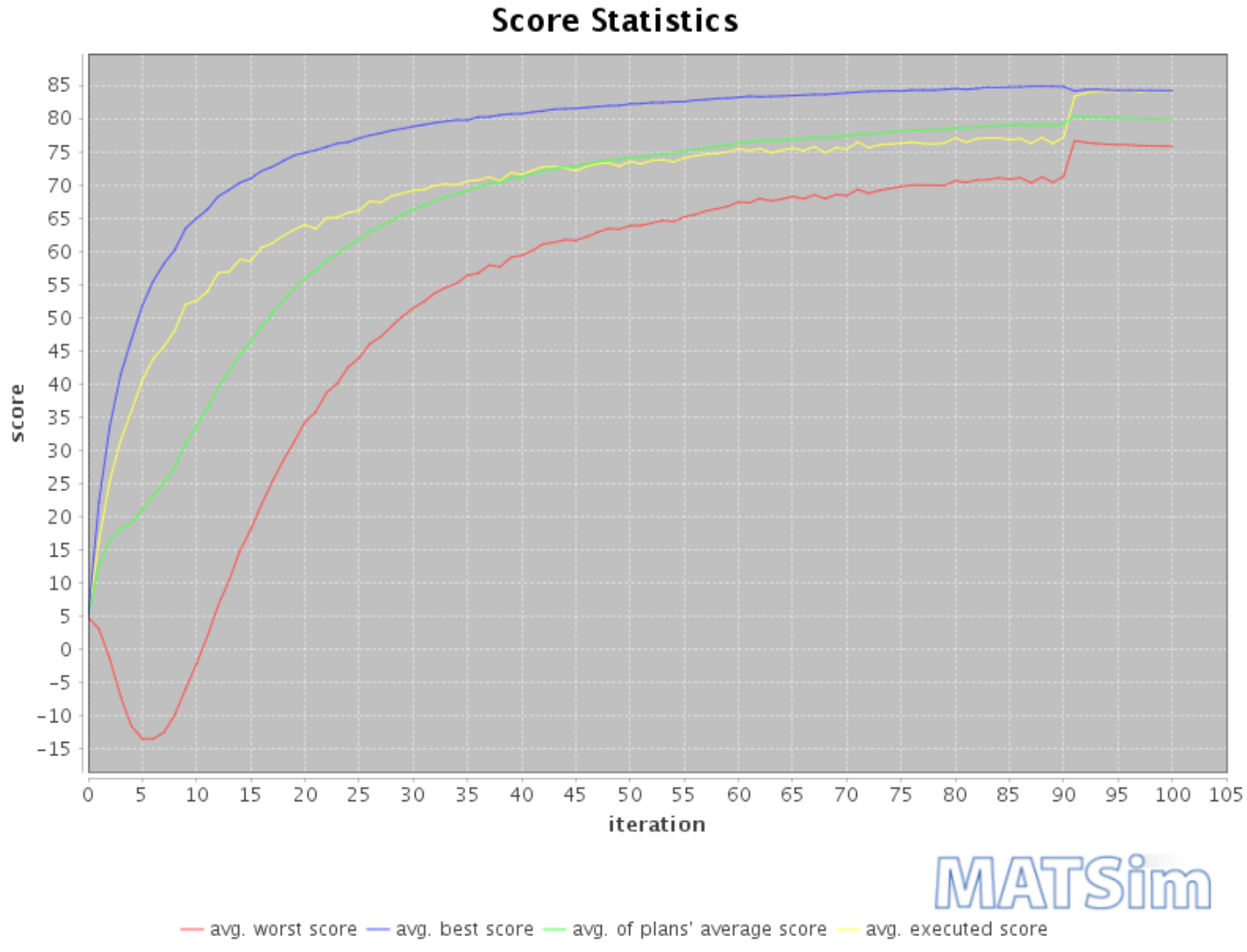
Scoring: config.xml: modeParams

```
<parameterset type="modeParams">
  <param name="mode" value="car" />

  <param name="constant" value="0.0" />

  <param name="marginalUtilityOfTraveling_util_hr" value="-6.0" />
  <param name="marginalUtilityOfDistance_util_m" value="0.0" />

  <param name="monetaryDistanceRate" value="0.0" />
</parameterset>
```

- **marginalUtilityOfTraveling**: normally negative, on top of opportunity cost of time.
- **marginalUtilityOfDistance**: normally negative, on top of the time (dis)utility
- **monetaryDistanceRate**: normally negative, monetary cost per distance

Monetary costs are converted into utility using the **marginalUtilityOfMoney** from the enclosing parameterset.

Scoring: scorestats.png



Sub-Populations

Persongroup-specific Replanning

In some cases, not every replanning strategy should be applied to every agent.

Example: Agents representing freight vehicle trips should not do mode choice.

In MATSim, this can be solved with "subpopulations".

Subpopulations

Every agent should have an attribute "subpopulation" (or similar name) with the corresponding group as value.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE objectAttributes SYSTEM "http://matsim.org/files/dtd/objectattributes_v1.dtd">
<objectAttributes>
    <object id="1">
        <attribute name="subpop" class="java.lang.String">regular</attribute>
    </object>
    <object id="2">
        <attribute name="subpop" class="java.lang.String">regular</attribute>
    </object>
    <object id="3">
        <attribute name="subpop" class="java.lang.String">freight</attribute>
    </object>
    <object id="4">
        <attribute name="subpop" class="java.lang.String">freight</attribute>
    </object>
    <object id="5">
        <attribute name="subpop" class="java.lang.String">regular</attribute>
    </object>
</objectAttributes>
```

At the moment, these agent attributes need to be specified in a separate file "personAttributes.xml".

Configuring Replanning for Subpopulations

```
<module name="plans" >
  <param name="inputPlansFile" value="/path/to/population.xml.gz" />
  <param name="inputPersonAttributesFile" value="/path/to/personAttributes.xml.gz" />
  <param name="subpopulationAttributeName" value="subpop" />
</module>

<module name="strategy" >
  <param name="maxAgentPlanMemorySize" value="3" />
  <param name="fractionOfIterationsToDisableInnovation" value="0.9" />

  <parameterset type="strategysettings">
    <param name="strategyName" value="ChangeExpBeta" />
    <param name="weight" value="0.7" />
    <param name="subpopulation" value="regular" />
  </parameterset>

  <parameterset type="strategysettings">
    <param name="strategyName" value="ChangeExpBeta" />
    <param name="weight" value="0.7" />
    <param name="subpopulation" value="freight" />
  </parameterset>

  <parameterset type="strategysettings">
    <param name="strategyName" value="SubtourModeChoice" />
    <param name="weight" value="0.1" />
    <param name="disableAfterIteration" value="70" />
    <param name="subpopulation" value="regular" />
  </parameterset>
</module>
```

Creating Subpopulation Attributes

```
Population population = scenario.getPopulation();
Person person1 = population.getPersons().get(Id.create(1, Person.class));
ObjectAttributes oa = population.getPersonAttributes();

oa.putAttribute(person1.getId().toString(), "subpop", "regular");

new ObjectAttributesXmlWriter(oa).writeFile("/path/to/personAttributes.xml.gz");
```

The following is the new way of storing person attributes, but it is not yet supported for the subpopulation feature:

```
person1.getAttributes().putAttribute("subpop", "regular");
```

Scoring Subpopulations

By default, every agent receives the same scoring parameters.

But in some cases, scoring parameters might be different, e.g. value of travel time in public transport might be lower if you have a season ticket or a card to travel at reduced fares.

MATSim supports providing different scoring parameters to different subpopulations.

Configuring Scoring for Subpopulations

```
<module name="planCalcScore" >
  <parameterset type="scoringParameters" >
    <param name="subpopulation" value="null" /> <!-- null acts as default / fall-back -->
    ...
    <parameterset type="activityParams" >
      <param name="activityType" value="home" />
      ...
    </parameterset>

    <parameterset type="modeParams" >
      <param name="mode" value="car" />
      ...
    </parameterset>
  </parameterset>

  <parameterset type="scoringParameters" >
    <param name="subpopulation" value="pt-season-ticket" />
    ...
    <parameterset type="activityParams" >
      <param name="activityType" value="home" />
      ...
    </parameterset>

    <parameterset type="modeParams" >
      <param name="mode" value="car" />
      ...
    </parameterset>
  </parameterset>
</parameterset>
```