MATSim Traffic Signals Contrib – User Guide

Development Version
last updated November 25, 2015

Theresa Thunig Dominik Grether


MATSim
Multi-Agent Transport Simulation

Contents

Chapter 1

Tutorial

This tutorial is tested since MATSim 0.4.0. This version is written for use with MATSim 0.7.0, but is still under construction.

1.1 Introduction

The MATSim contrib for traffic lights is able to simulate traffic signals microscopically. For fixed-time control a default implementation is provided. No programming is required to simulate fixed-time traffic signal control. Traffic-responsive signal control needs custom implementation but can benefit from the provided infrastructure.

1.1.1 Terminology

To help you to understand xml formats, configuration, and code better, have a look at the terminology in Fig. 1.1.

MATSim term	Real world terms	Description
signal	traffic light, traffic signal	A physical box standing somewhere on the transport network indicating driving allowed/permitted
signal group	a group of traffic lights	Logical group of traffic lights, all lights display same color at the same time
signal control	traffic controller	Algorithm or control scheme that determines which colors are displayed by the different Signal Groups, e.g. fixed-time control
signal system	signalized crossing	Collection of Signal Groups that is controlled by the same Signal Control

Figure 1.1: Terminology

1.1.2 Overview

The contrib consists of several modules. None of them is required if you are willing to code yourself. The codebase is written under the assumption that the following features are optional:

- Lanes
- Intergreens

Both are required under certain circumstances, only. See the upcoming MATSim book for further hints.

Lanes

TBD.

Intergreens

MATSim's queue model implementation does not contain any collision detection. That is, a signal control can switch several conflicting approaches to green at the same time. The simulation does not report any error.

To prevent such faulty behavior, the intergreens component can be switched on. An xml file defines for each signal system the intergreen times of the signal groups, i.e. the minimal time period between the ending of the one and the beginning of the other signal groups green phase. This information is then used to validate the signal control while the simulation is executed and may result in a warning or an exception.

This is especially important if you implement a custom, traffic-responsive signal control strategy.

1.2 Usage & Configuration

1.2.1 Usage

- Include the .jar file of the contrib into your java classpath
- Write a java `main(..)` method that loads your config and contains the following lines of code:

```
Scenario scenario = ScenarioUtils.loadScenario( config ) ;
SignalSystemsConfigGroup signalsConfigGroup = ConfigUtils.addOrGetModule(config,
      SignalSystemsConfigGroup.GROUPNAME, SignalSystemsConfigGroup.class);
scenario.addScenarioElement(SignalsData.ELEMENT_NAME,
      new SignalsScenarioLoader(signalsConfigGroup).loadSignalsData());

Controler c = new Controler( scenario );
c.addOverridingModule(new SignalsModule());
```

- Execute `c.run(..)` somewhere in your `main(..)`

1.2.2 Config Options

If input files for signal systems are already available, simulation of traffic lights can be enabled via MATSim config options:

- Set at least three input file names in the config module signalsystems:
  - parameter name: `useSignalsystems` value: `true` or `false`
  - parameter name: `signalsystems` value: path to a file in the `signalSystems_v2.0.xsd` file format
  - parameter name: `signalgroups` value: path to a file in the `signalGroups_v2.0.xsd` file format

- parameter name: `signalcontrol` value: path to a file in the `signalControl_v2.0.xsd` file format
- parameter name: `useAmbertimes` (optional) value: `true` or `false`
- parameter name: `ambertimes` (optional) value: path to a file in the `amberTimes_v1.0.xsd` file format
- parameter name: `useIntergreenTimes` (optional) value: `true` or `false`
- parameter name: `intergreentimes` (optional) value: path to a file in the `intergreenTimes_v1.0.xsd` file format
- parameter name: `actionOnIntergreenViolation` (optional) value: `warn` or `exception`

1.3 Further Reading

A conceptual introduction to this MATSim contribution can be found in [2]. For case studies in German, see [1, 3, 4].

1.4 Generating Input Data – Example

There is a small example in order to help you getting started, the class is RunCreateTrafficSignalScenarioExample.

The example uses the network shown in Figure 1.2.

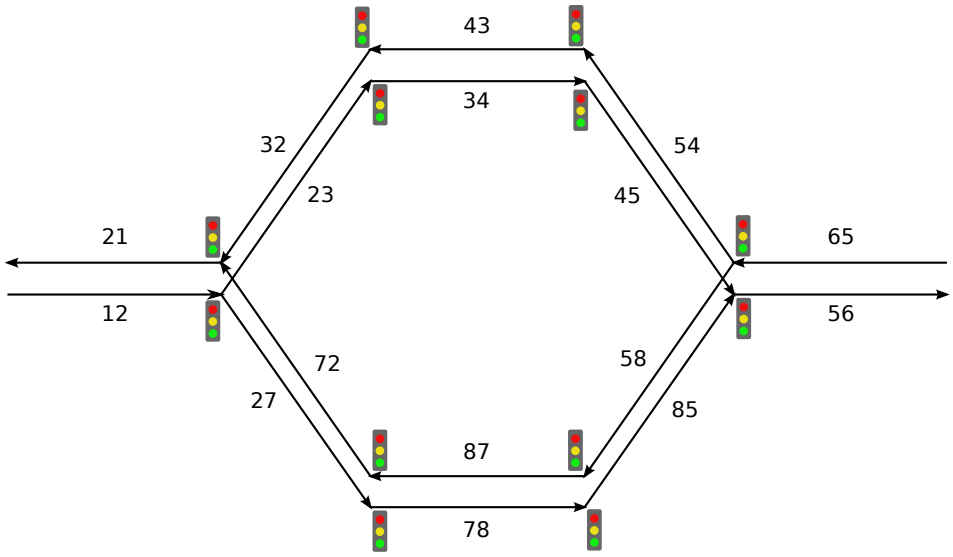

Figure 1.2: Network used by the scenario in RunCreateTrafficSignalScenarioExample. The numbers show the link ids. Traffic lights to be created are shown as small icons.

The following highlights important lines of code method by method.

Code

Input files for the examples can be found in the folder contribs/signals/examples/tutorial/example90TrafficLights of the current MATSim release.

Example code can be found in the folder src/main/java in the package tutorial.trafficsignals in the signals contrib.

RunCreateTrafficSignalScenarioExample.run()

This method sets up the config and writes the signals that are created in submethods to file.

As optional module, the signal systems module has to be switched on in the scenario first:

```
config.setUseSignalSystems(true);
```

Then, qsim as mobility simulation should be used:

```
config.controler().setMobsim("qsim");
```

After the scenario is loaded, the top-level container for all signal related data has to be added:

```
SignalSystemsConfigGroup signalsConfigGroup =
ConfigUtils.addOrGetModule(config,
SignalSystemsConfigGroup.GROUPNAME, SignalSystemsConfigGroup.class);
scenario.addScenarioElement(SignalsData.ELEMENT_NAME, new
SignalsScenarioLoader(signalsConfigGroup).loadSignalsData());
```

and is afterwards retrieved by:

```
SignalsData signalsData = (SignalsData)
scenario.getScenarioElement(SignalsData.ELEMENT_NAME);
```

RunCreateTrafficSignalScenarioExample.createSignalSystemsAndGroups(..)

This method creates the physics, i.e. locations of signals. Furthermore SignalGroups for the Signals are created.

First a SignalSystem has to be created and explicitly added to the container:

```
SignalSystemData sys =
systems.getFactory().createSignalSystemData(Id.create("3",
SignalSystem.class));
systems.addSignalSystemData(sys);
```

Then on link 23 a Signal with Id 1 can be created:

```
SignalData signal =
systems.getFactory().createSignalData(Id.create("1", Signal.class));
sys.addSignalData(signal);
signal.setLinkId(Id.create("23", Link.class));
```

The Signal must be added to the SignalSystem after creation.

This is continued until all Signals of the SignalSystem are created.

We then need some SignalGroup for the Signals. In this example each Signal has itsown group. That is done by calling:

```
SignalUtils.createAndAddSignalGroups4Signals(groups, sys);
```

RunCreateTrafficSignalScenarioExample.createSignalControl(..)

This method adds a fixed-time traffic signal control on top of the SignalGroups created by the last method. The SignalGroups control the hardware, i.e. the Signals. MATSim Events are created for SignalGroups.

Each SignalSystem is equiped with a fixed-time control. Each SignalSystem is setup the same way: A cycle time of 120 sec is used. Each direction gets green for second 0 to 55 within the cycle. Offsets for green waves are set to 0 seconds.

The Code can be read as follows:

Create a fixed-time control and add it to the container:

```
SignalSystemControllerData controller =
```

```
control.getFactory().createSignalSystemControllerData(id);
control.addSignalSystemControllerData(controller);
controller.setControllerIdentifier(
DefaultPlanbasedSignalSystemController.IDENTIFIER);
```

Then, create a fixed-time control plan (a plan can be disabled or changed at a certain time of day) and add it to the container:

```
SignalPlanData plan =
control.getFactory().createSignalPlanData(Id.create("1",
SignalPlan.class));
controller.addSignalPlanData(plan);
```

Each fixed-time control plan has specific attributes for cycle and synchronization offset:

```
plan.setCycleTime(cycle);
plan.setOffset(0);
```

Create specific green times (onset = switch to green, offset = switch to amber/red) for all SignalGroups and add them to the signal control plan via

```
SignalGroupSettingsData settings1 =
control.getFactory().createSignalGroupSettingsData(Id.create("1",
SignalGroup.class));
plan.addSignalGroupSettings(settings1);
settings1.setOnset(0);
settings1.setDropping(55);
```

Visualization

If you want to visualize the created scenario try to get and run the OTFVis contribution to MATSim (use MATSim head or nightly build instead of 0.4.1), see OTFVis(this contribution is unsupported, please submit patches instead of mails).

If OTFVis runs successfully, you may be able to run

`tutorial.trafficsignals.VisSimpleTrafficSignalScenario`

within an up to date (¿ 04.12.2012) checkout within eclipse.

Another option is to write the scenario to file via the example code and then start OTFVis with the created config file.

1.4.1 Links

- Technical Documentation (JavaDoc)
- OTFVis
- Dissertation of D. Grether

1.4.2 Intergreens

There is a small example how to create an intergreen xml file with given intergreen times for specific signal groups (see tutorial.trafficsignals.RunCreateIntergreensExample.java).

If there is no intergreen data available for your scenario you may use the intergreen times of a correct fixed time signal control, i. e. a signal control with realistic intergreen times where no collisions may occur (therefor you may look at playground.dgrether.signalsystems.sylvia.data.TtCalculateSimplifiedIntergreens.java).

1.5 Input Files – Examples

1.5.1 Intergreens

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<intergreenTimes xsi:schemaLocation="http://www.matsim.org/files/dtd http://www.
    matsim.org/files/dtd/intergreenTimes_v1.0.xsd" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xmlns="http://www.matsim.org/files/dtd">
      <signalSystem refId="23">
            <endingSignalGroup refId="1">
                  <beginningSignalGroup refId="2" timeSeconds="5" />
                  <beginningSignalGroup refId="3" timeSeconds="3" />
                  <beginningSignalGroup refId="4" timeSeconds="3" />
            </endingSignalGroup>
      </signalSystem>

      <signalSystem refId="42">
            <endingSignalGroup refId="1">
                  <beginningSignalGroup refId="2" timeSeconds="5" />
            </endingSignalGroup>
            <endingSignalGroup refId="2">
                  <beginningSignalGroup refId="1" timeSeconds="3" />
            </endingSignalGroup>
      </signalSystem>
</intergreenTimes>
```

Appendices

Bibliography

[1] J. Bischoff. Verkehrsabhängige Lichtsignalanlagensteuerung – Vergleich und simulationsbasierte Evaluation. Master's thesis, December 2010.

[2] D. S. Grether. *Extension of a Multi-Agent Transport Simulation for Traffic Signal Control and Air Transport Systems*. PhD thesis, Technische Universität Berlin, 2014.

[3] A. Neumann. Modellierung und Evaluation von Lichtsignalanlagen in Queue-Simulationen. Master's thesis, 2008.

[4] D. Röder. Modellierung und Simulation einer adaptiven Steuerung für Lichtsignalanlagen in Queue-Simulationen. Master's thesis, 2010.