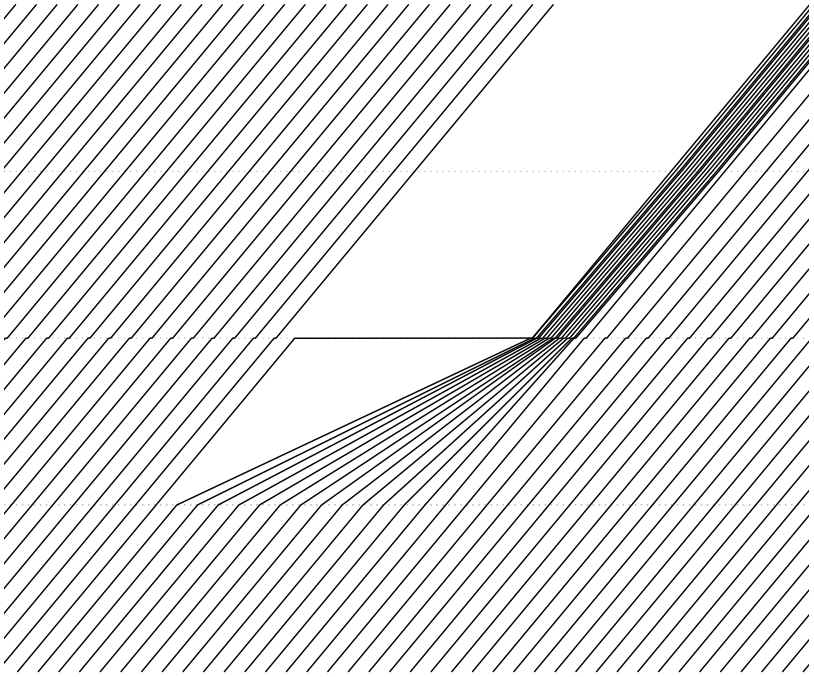


The influence of railroad crossings on networks in the MATSim environment

Flavio Poletti

Philipp A. Fuchs

Patrick M. Boesch

ETH Zürich

May 2016

STRC

16th Swiss Transport Research Conference

Monte Verità / Ascona, May 18 – 20, 2016

ETH Zürich

The influence of railroad crossings on networks in the MATSim environment

Flavio Poletti
ETH Zürich
polettif@student.ethz.ch

Philipp A. Fuchs
ETH Zürich
fuchsp@student.ethz.ch

Patrick M. Boesch
IVT
ETH Zurich
Stefano-Franscini-Platz 5
CH-8093 Zürich
phone: +41 44 633 39 52
patrick.boesch@ivt.baug.ethz.ch

May 2016

Abstract

Railroad crossings are obstacles in road traffic and can have a significant influence on car travel time, depending on location and closing times. Despite their impact on route choice, such crossings are not yet part of the MATSim environment. This paper presents a successful implementation of railroad crossings in MATSim. It allows to analyse the effect of railroad crossings on routing choices and traffic volumes. The proposed solution was tested on a simple network. The railroad crossings were implemented using the Time-Dependent Network package. Based on the events of an initial simulation run, *network change events* – which set the capacity of a crossing link to zero – were created and used subsequently. The expected effects on the network could be observed successfully (i.e. rerouting because of closed crossings). Therefore, it is recommended to include railroad crossings via *network change events* in simulations.

Keywords

Railroad crossing, Level crossing, Agent based modelling, Rerouting, Time-dependent network

1 Introduction

This paper presents the successful implementation of railroad crossings in MATSim. MATSim is an activity-based, extendable, multi-agent transport simulation framework implemented in Java. Over multiple iterations, every agent "repeatedly optimizes its daily activity schedule, while in competition for space-time slots with all other agents on the transportation infrastructure." (Horni *et al.,* 2016a, 4).

So far, railroad crossings were not implemented in MATSim (Horni *et al.,* 2016b) but their impact on route choice in real life can be significant, depending on their location and closing time. An implementation should be able to "close" a crossing link before a train passes and reopen it after the train has passed. The cars that want to cross should queue in front of the crossing during closing time. Section 2 covers the used methodology. Section 3 gives a detailed overview of the implementation. The new package was tested in a sample simulation to see whether there are new congested areas or if the overall behaviour of agents changes. The results of these test runs are summed up in section 4. In sections 5 and 6 a conclusion and recommendations about future work are given. This work does not cover obtaining crossing locations from existing networks or other sources.

2 Methodology

Railroad crossings can be implemented in MATSim using one of the following packages:

- *Signals* (Grether and Thunig, 2016): Introduce a traffic signal at exactly the position of the railroad crossing
- *Time-Dependent Network* (Nagel *et al.,* 2016a): Introduce an additional link on the road exactly where the railroad crossing would be. On this link one could:
  - Change the freespeed to zero during closing time
  - Change the freespeed to such a low speed, that it would take one car exactly the time needed for the crossing to close and open again.
  - Change the capacity to zero during closing time.

The pursued strategy is changing the capacity of a crossing link. The time-dependent network package allows to create *network change events* which depend (indirectly) on the public transport schedule. Changing the freespeed on a link to a speed below a certain limit is not possible by default. MATSim raises the speed to a minimum level in order to maintain a stable simulation. It is important to note that setting the capacity of a link to zero does not actually prevent vehicles from entering that link. However, only one vehicle is "allowed" to enter the crossing link. The vehicle is not able to leave the link until the capacity is reset to the initial level.

In MATSim, a *stuck time* is defined to prevent gridlocks. After this predefined *stuck time* vehicles are forced into the subsequent link (Horni and Nagel, 2016, 42). For railroad crossings, one should keep in mind to set this threshold higher than the closing time of the crossings. This is a crucial step because if missed, additional vehicles are allowed to enter the crossing link as soon as they waited longer than the *stuck time*.

The *network change events* could be generated from the public transport schedule file. To account for simulation dynamics, an approach using the events file of an initial simulation was chosen. An event handler detects when a train enters a link with a crossing and calculates the closing time of a corresponding road link. This approach allows more flexibility since it is independent of the transit schedule file and could use non scheduled events as well. However, if a train is delayed in later iterations, the crossing will not close at the right time. Therefore, one should keep in mind that the simulated day is a rather special case, since there are no delays in the railroad system. This is acceptable as railbound public transport is usually simulated on a separate, non-congested network anyway (Boesch and Ciari, 2015).

Other approaches such as changing the throughput of a node, for example via Qsim (Nagel *et al.,* 2016b), or changing the network while the simulation is running require changes in the MATSim core. However, such adaptions reduce usability as well as interoperability and would go beyond the scope of this work.

Implementing a signal at each crossing is too restrictive because the already existing *signals* package is cycle based. A separate signal group with a fixed cycle would have to be established for each crossing. This cycle stays the same for the whole day. Therefore, irregular time tables for public transport are not easy to implement. Creating multiple short cycles which are only viable during the time of a train crossing is too complex for good usability since multiple signal systems, groups and controllers would be required. Therefore, simulating railroad crossings using a time-dependent network has been chosen.

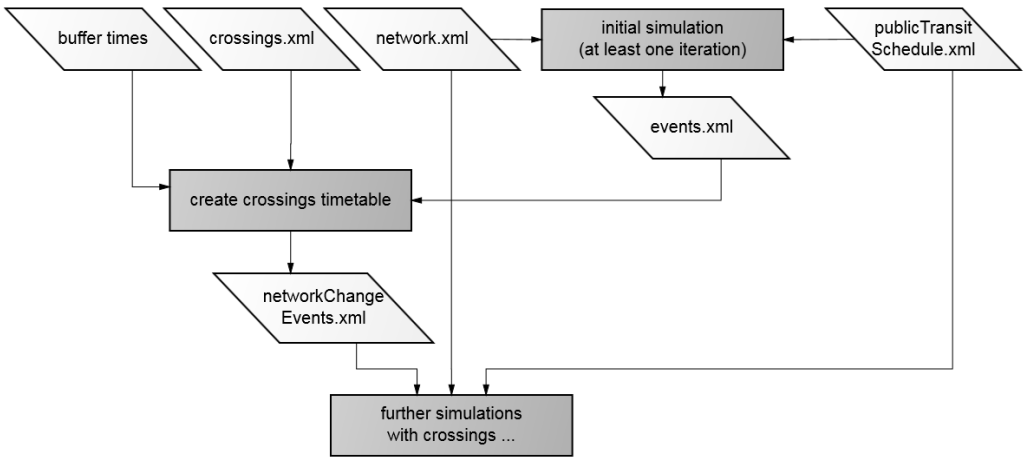
3 Implementation in MATSim

3.1 General Workflow

In MATSim, agents perform activities at different locations during a simulated day. Each time an agent enters or leaves a link, starts or ends an activity or changes transport modes, this is registered as an event. A simulation run consists of many iterations of the simulated day, during which agents optimise their plans in a co-evolutionary process. After a simulation (or after each iteration of it), the list of events can be analysed. This means that after at least one iteration of a simulation run one can check when a train entered a link with a crossing on it and calculate the crossing's closing time. This information is then stored in a *network change events* file which is used in subsequent simulations to determine the closing and reopening times of the crossing links. In order to create *network change events*, several inputs are needed (also shown in Figure 1):

- An adapted network file.
- A public transit schedule file.
- An events file obtained from an initial simulation run.
- A list of crossings which ties the public transport links to the corresponding crossing links.
- Buffer times (in seconds): The crossing link is closed during a specified time. Since the crossing should be closed well before the train arrives and stay closed until the last waggon passes, this time should be separated into one part before a train actually passes and another part afterwards.

Figure 1: This flow chart shows what inputs are needed to generate the *network change events* file. Processes are rectangular, in- and outputs are rhomboids.



3.2 In- and outputs

Network file Train links in a MATSim network do not follow the real railroads precisely. Usually, they are implemented as straight links from station to station. This means the positioning of the intersections needs to be done manually. The network is adapted with additional links at the intersections: It is assumed that a crossing-link has a length of ten meters. The length of the original link has been decreased in order to keep the total length constant. Figure 2 visualises this process. The length of the link does not have any significant impact on the simulation, shorter links are just harder to visualise.

Public Transit Schedule file A public transit schedule is required for the initial simulation to simulate trains, which in turn allows the detection of trains passing a crossing.

Events file Using the events file, the time a train enters a railroad link can be detected. By dividing the link length by the time difference between the train entering and leaving the link, the actual speed of the train can be calculated. The time needed for a train to reach the crossing is calculated by dividing the distance between the crossing and the link origin node by the speed of the train.

Crossing file The crossing links have to be tied to a railroad link in order for the simulation to know which link has to be closed if a train is arriving. Figure 3 shows an example of such a file. A *ptLink* is a railroad link whereas the *crossingLink* denotes the corresponding car link crossing the tracks.

Buffer times In Switzerland, a railroad crossing should not be closed longer than two minutes per passing (Bundesamt für Verkehr, 2014). In order to maintain straightforward conditions it was assumed that the crossing closes 80 seconds before the train passes the intersection and reopens 40 seconds after that moment. With these buffers the variations in the actual arrival time of a train (delays or early departures) can be covered to some extent.

Output: *network change events* The final product of the implementation is the *network change events* file that defines which link is changed at what time. It is used in further simulations that base on the same network and public transport schedule.

Figure 2: The solid link signifies a street link and the dashed one a rail link. The upper figure shows how the link has been before the change. The one below has an newly introduced crossing link at the intersection between the rail link and the street link.

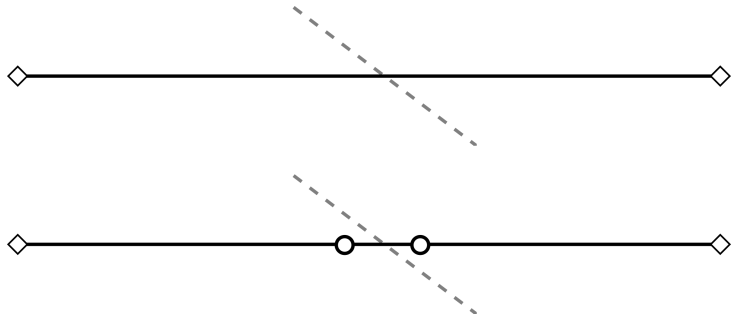


Figure 3: The crossings XML input file lists every link that crosses a public transport link (rail tracks in most cases). Usually the crossing links are the same for opposite public transport links. The order of the crossing links is irrelevant, the file is only used to connect the public transport links and their crossing links.

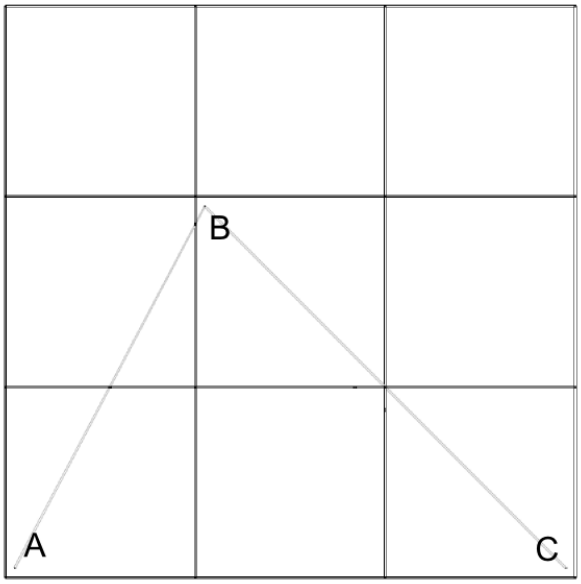
```
<crossings xmlns="http://www.matsim.org/files/dtd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.matsim.org/files/dtd
  ./dtd/crossings.xsd">

  <ptLink id="12">
    <crossingLink refId="1222-x" />
    <crossingLink refId="2212-x" />
    <crossingLink refId="2223-x" />
    <crossingLink refId="2322-x" />
  </ptLink>

  <ptLink id="21">
    <crossingLink refId="1222-x" />
    <crossingLink refId="2212-x" />
    <crossingLink refId="2223-x" />
    <crossingLink refId="2322-x" />
  </ptLink>

  ...

</crossings>
```

4 Testing

In a first step, a simple straight road with one crossing has been implemented to check if the proposed methodology works. In this scenario, the agents start every ten seconds during a certain period.

To analyse the changes in the behaviour of agents and the influence on a network level, the simple network *pt-tutorial* (shown in Figure 4) was used. It was assumed that a small scenario is sufficient to test the strategy using a time-dependent network since the developed code should work independently of the network size. The population consists of 2700 agents. According to the public transit schedule, a train leaves A for C (and vice versa) every 15 minutes between 6 AM and 11 PM.

Figure 4: The network used for the test scenario. The grey links are railroad links, the black ones car links. Trains drive from A to C via B (and vice versa).



The output of the MATSim simulation has been analysed in several ways. First the queueing behaviour has been observed using the visualisation software VIA (senozon AG, 2016). With it it was possible to check whether cars actually queue and to get an idea where the cars are going. In addition, the travel times of vehicles entering a crossing link and the traffic volumes on this link were analysed (in the simplistic one-link scenario). Furthermore, a time-space diagram has been established to show the queueing behaviour. Additionally, the traffic volumes over the whole day have been compared.

4.1 Travel time and queueing behaviour

As shown in the subsequent figures, using *network change events* to model railroad crossings has the desired effects on travel times and queueing behaviour.

Figure 5 shows that cars are prohibited from entering a closed crossing link (except the first one which is indicated by the solid red line). Since cars are not able to leave the link until the crossing reopens, their travel time (calculated as difference between entering the link and leaving the link) is "stretched" to the remaining closing time. During the crossing's closure, no car is on the link behind the crossing. Directly after the crossing is reopened, the traffic volume on this link is increasing to a very high level as shown in Figure 6. The volume is so high that the travel times increase slightly.

Figure 5: This graph shows the travel times for cars on the link ahead of the crossing, a crossing link and the link behind the crossing. The crossing link closes from 06:00 until 06:02. While the railroad crossing is closed (i.e. the crossing link's capacity is zero), travel times on the link ahead of the crossing increase accordingly. Vehicles entering right at the beginning of closing time take additional 120 seconds to pass through while vehicles entering towards the end have less travel time.

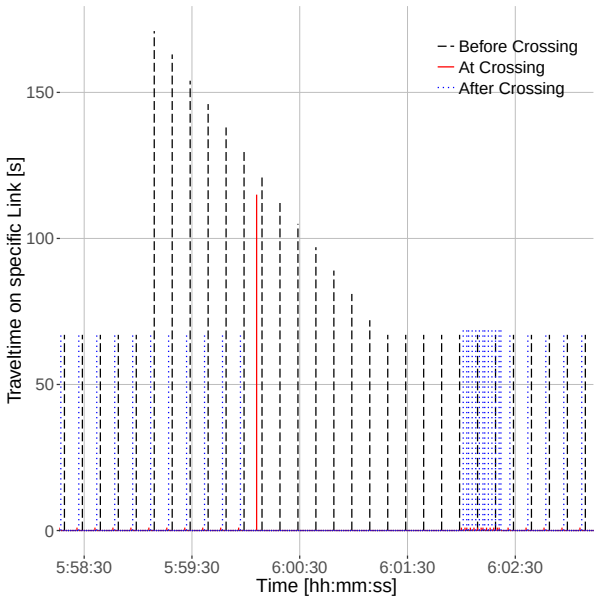


Figure 7 shows how the developing queue manifests itself during the closure of a crossing. The cars slow down on the link leading towards the crossing link. If a train arrives at the intersection, the travel times of cars entering the link in front of the crossing are adapted (as shown in Figure 5).

Figure 6: This graph shows the volumes of cars on the link ahead of the crossing, the crossing link and the link behind the crossing. The crossing link closes from 06:00 until 06:02. While the railroad crossing is closed, the volumes on the link ahead increase.

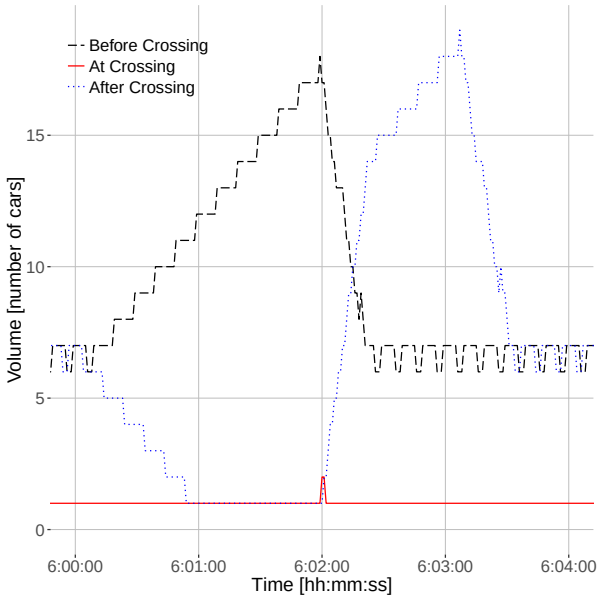
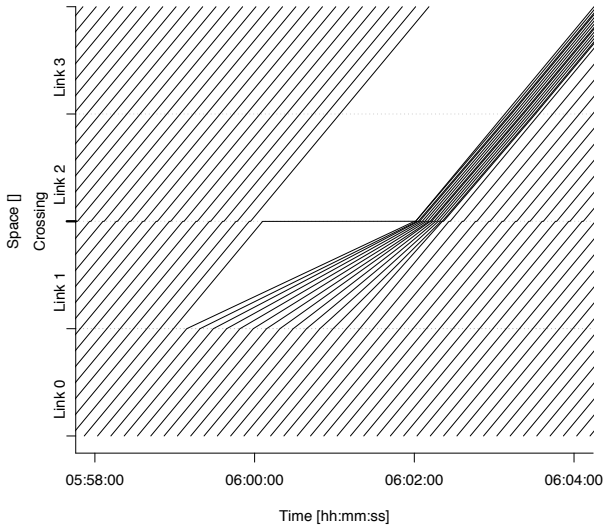


Figure 7: This Time-Space Diagram illustrates on the simplistic scenario how the cars are "queueing" in front of the closed crossing. Actually their travel-times are adapted in order to arrive at the crossing when it reopens. The plot only takes into account the *linkLeaveEvents*, since one cannot track an individual vehicle on a link.



4.2 Simulations on a test network

Behavioural changes of the agents have been tested on the aforementioned network (see Figure 4). To check whether the agents did change their behaviour or not, two simulations with 100 iterations each have been run: One with and one without closing crossing links. The average score of the executed plan did not change remarkably after 80 iterations.

The modal split change was less than 0.5 percentage points and was thus not relevant in this scenario. The main reason for the small change is the lack of public transport alternatives for most road routes affected by crossings in the scenario. Thus, agents rather change their route or departure time than the transport mode.

Figure 8: The width of the arrows indicates the absolute amount of traffic which has changed on one link over the whole day. The value next to it tells the relative change from the base state to the scenario with a crossing installed. *NaN* indicates that the link did not have traffic on it in the base scenario. The squares show the locations of railroad crossings.

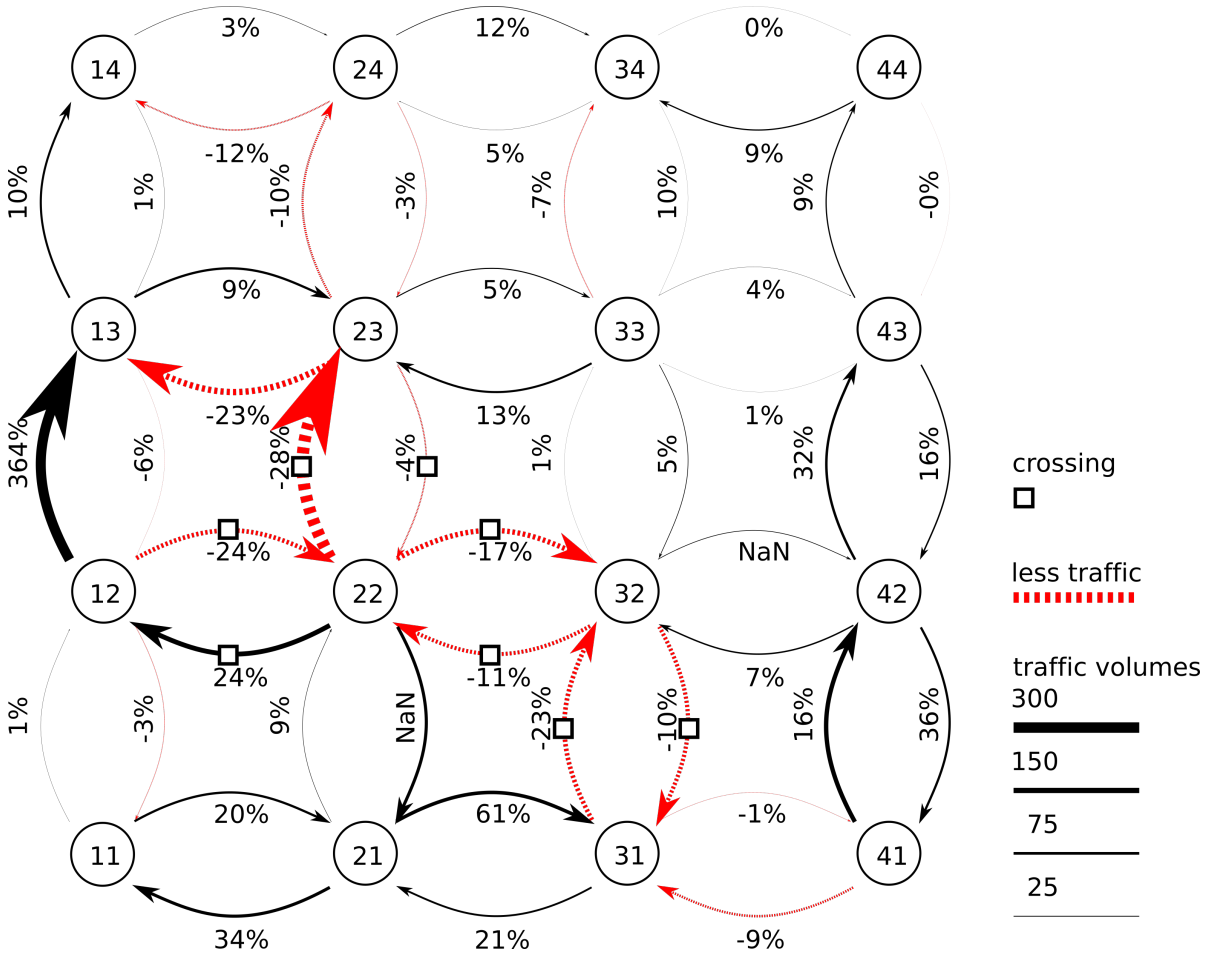


Figure 8 shows how the traffic in the scenario is distributed and in which direction the traffic reorganises. The names of the links are composed by the start and end node. For example the link from node *24* to node *34* is *2434* whereupon the opposing link is *3424*. As expected, nearly all links with a crossing have reduced traffic flows. This seems quite reasonable since a crossing is an obstacle in an agent's path and in order to avoid this obstacle he might change his route. Surprisingly, link *2212* has increased traffic volumes. The reason for this might be that activities are tied to certain links. To reach an activity, an agent has to drive along the referenced link. A lot of agents have activities on link *1121* which is accessed via the links *2212* and *1211*. Most of the links next to the crossing links experience a traffic increase. Some links without a crossing have decreasing traffic volumes (i.e. *2313* and *4131*). The reason for this might be that they are affected by the closed link in front or after them. It seems that the traffic detouring the crossing on link *3231* ends up coming to node *31* over link *2131*. Keeping that in mind it appears reasonable that the traffic on link *3141* does not really change at all. Some minor network effects are visible as well (e.g. the upper part of the system is not affected by the railways but experiences also changes in its traffic volume).

As expected, the impacts of railroad crossings on the agents' route choices are substantial. Agents tend to detour the links crossed by railways.

5 Conclusion

Using a *time-dependent network* with capacity changes of crossing links is a viable way to implement railroad crossings in MATSim. Agents react as expected and the impact of railroad crossings can be seen in route choices and different link volumes on the network. Including the presented package in a simulation is recommended, especially when a scenario either has a lot of railroad crossings or crossings with long closing times.

However, obtaining crossing locations, adapting the network and connecting crossings and railroad links still is a lot of manual work that has to be done before the package can be applied.

6 Future Work

The queueing behaviour could be simulated in a manner which conforms with the expectations. There are some open issues which should be mentioned.

- The time table for the crossings are generated in a first iteration and never changed again. Thus, if a train gets delayed in a later iteration, the crossing might close if the train is not arriving. In reality there would be a sensor which detects an arriving train and closes the crossing for cars. A next step would be an implementation of a sensor in the code to make the simulation more realistic.
- In this work only one train schedule has been tested. It would be worthwhile to check how different train schedules influence the behaviour of agents.
- The *pt-tutorial* scenario is rather a sandbox to test different possibilities of how to implement railroad crossing in the first place. In order to get more meaningful results, the simulation should be made using a bigger scenario with a more sophisticated train and road network.
- An additional script to obtain crossing locations and the corresponding railroad links automatically (from Open Street Map data for example) would complete the work flow and make the package ready to use without a lot of manual work.

Acknowledgement

We would like to thank Dr. Francesco Ciari and Milos Balac for their ideas as well as inputs on how to integrate railroad crossings in MATSim and especially for their support in Java related problems. As it has been important for us to be able to visualise the results, we would also like to thank senozon AG for letting us use their software VIA.

7 References

Boesch, P. M. and F. Ciari (2015) A Multi-Modal Network for MATSim, paper presented at the *15th Swiss Transport Research Conference*, Ascona, April 2015.

Bundesamt für Verkehr (2014) AB-EBV AB37c, `http://www.bav.admin.ch/grundlagen/03514/03533/03614/index.html?lang=de`. Access: January 2016.

Grether, D. and T. Thunig (2016) *The Multi-Agent Transportation Simulation MATSim*, chap. 12: Traffic Signals and Lanes, 86 – 92, Ubiquitiy, London.

Horni, A. and K. Nagel (2016) *The Multi-Agent Transportation Simulation MATSim*, chap. 4: More About Configuring MATSim, 39 – 49, Ubiquitiy, London.

Horni, A., K. Nagel and K. W. Axhausen (2016a) *The Multi-Agent Transportation Simulation MATSim*, chap. 1: Introducing MATSim, 3 – 8, Ubiquitiy, London.

Horni, A., K. Nagel and K. W. Axhausen (eds.) (2016b) *The Multi-Agent Transportation Simulation MATSim*, Ubiquity, London.

Nagel, K., M. Rieser and A. Horni (2016a) *The Multi-Agent Transportation Simulation MATSim*, chap. 6: MATSim Data Containers, 62 – 66, Ubiquitiy, London.

Nagel, K., M. Rieser and A. Horni (2016b) *The Multi-Agent Transportation Simulation MATSim*, chap. 11: QSim, 82 – 84, Ubiquitiy, London.

senozon AG (2016) VIA visualization and analysis tool, `http://via.senozon.com`. Access: January 2016.