Technical University of Munich

Interdisciplinary Project

Department of Informatics

Department of Civil, Geo and Environmental Engineering

Development of an efficient graphical network editor for an open-source agent-based transport simulation

Author:
Aikaterini Intzevidou

Supervisors:
Nico Kuehnel
Rolf Moeckel

August 1, 2021

Contents

List of Figures

1 Introduction

The MATSim network editor is a Java-based graphical user interface application used to make the creation and editing of transportation network files for MATSim easy. The application was built using modern tools and provides a map view based on which the user can create networks from scratch, edit existing networks with additions, deletions and modifications, or even just view a network.

```
<network name="example network">
   <nodes>
      <node id="1" x="0.0" y="0.0"/>
      <node id="2" x="1000.0" y="0.0"/>
      <node id="3" x="1000.0" y="1000.0"/>
   </nodes>
   <links>
      <link id="1" from="1" to="2" length="3000.00" capacity="3600"
            freespeed="27.78" permlanes="2" modes="car" />
      <link id="2" from="2" to="3" length="4000.00" capacity="1800"
            freespeed="27.78" permlanes="1" modes="car" />
      <link id="3" from="3" to="2" length="4000.00" capacity="1800"
            freespeed="27.78" permlanes="1" modes="car" />
      <link id="4" from="3" to="1" length="6000.00" capacity="3600"
            freespeed="27.78" permlanes="2" modes="car" />
   </links>
</network>
```

Figure 1: Network file example [1].

These networks are stored in .xml files, following the form compatible with MATSim, as seen in Figure 1. In the following chapters, we will describe the coding aspect of the project in greater detail, and will also provide a few use case examples of the application.

2 Setup and Prerequisites

2.1 Setup and Prerequisites

Technical specifications of the project:

- Java - openjdk 15
- Maven - maven 4.0.0
- JavaFX - javafx-sdk-11.0.2
- Map library - mapjfx 2.14.1
- Intellij Idea IDE 2021.1 Ultimate Edition

For setting up the project, we created a Maven project by following: File >New >Project and choosing Maven project on the panel in the right. Then, we added the JavaFX platform to the project, adding the openjfx Maven dependency to the pom.xml file.

The libraries needed for the project are the javafx-base, javafx-graphics, javafx-controls and javafx-fxml ones, which we include in the VM parameters:

```
-module-path /path/to/JavaFx/lib
–add-modules=javafx.swing,javafx.graphics,javafx.fxml,
javafx.media,javafx.web
–add-reads
javafx.graphics=ALL-UNNAMED
–add-opens
javafx.controls/com.sun.javafx.charts=ALL-UNNAMED
–add-opens
javafx.graphics/com.sun.javafx.iio=ALL-UNNAMED
–add-opens
javafx.graphics/com.sun.javafx.iio.common=ALL-UNNAMED
–add-opens
javafx.graphics/com.sun.javafx.css=ALL-UNNAMED
–add-opens
javafx.base/com.sun.javafx.runtime=ALL-UNNAMED
–add-exports
javafx.graphics/com.sun.javafx.tk=ALL-UNNAMED
–add-exports
javafx.graphics/com.sun.javafx.util=ALL-UNNAMED
```

For the map and the markers on the map, we used the mapjfx library.

2.2 Running the application

To run and use the application, one can simply download the project of pom.xml file from GitHub, open in an IDE - Intellij is recommended - wait for the project to load along with the required dependencies, then run the project.

Alternatively, one can download the .jar file and run the application directly.

3 User Interface

The graphical user interface of the editor contains a control panel on the top, a side panel on the right, a map view panel on the center-left and a detail panel on the bottom. Figure 2

Overall, the user can create new networks, import from .xml, .gz or .osm files, add, edit and delete nodes and links, run validation on the network to detect possible issues and run cleaning on the network, to remove the nodes or links that would be problematic in a simulation run. On top of that, the user can save their changes, change default settings of the map viewer, like the zoom and center coordinates, and view information about the network and its elements.

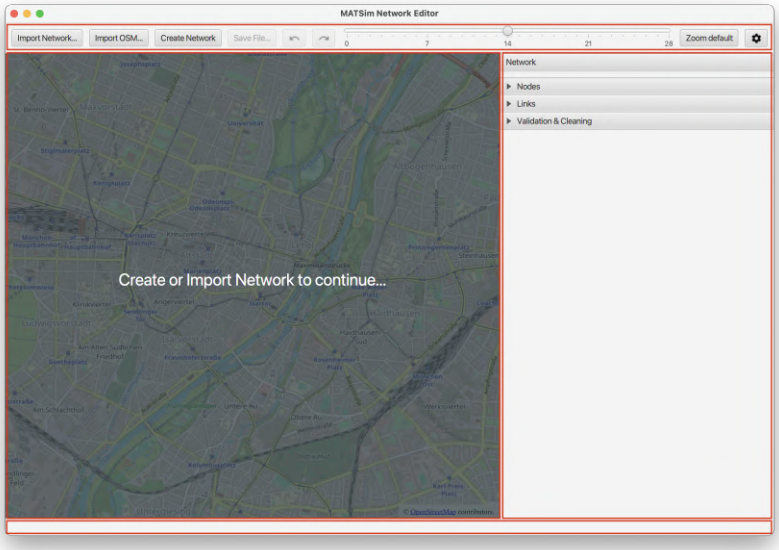

Figure 2: Start screen of the editor application.

3.1 Top Panel

The top panel contains an "Import Network" button, an "Import OSM" button, a "Create Network" button, a "Save" button, undo and redo buttons, a zoom bar and a "Zoom default" button, and finally, a settings button.

- Import Network: This feature opens up the file chooser of the operating system, for the user to select a .xml or .gz file containing a network to be loaded on the map. This network should adhere to the MATSim format 1. The user also needs to select the coordinate system in which the file is saved. This system does not have to necessary match the underlying system of the map which is WGS84, since the necessary transformations are performed automatically in loading and writing out the network, in order to match the desired coordinate system.

- Import OSM: Similarly to importing a network from a .xml file, this feature does the same but letting a user choose a .osm file to import a network from. The network is transformed into a MATSim-understood format and loaded on the map.

- Create Network: This feature creates an empty network object for the user to add their nodes and links to. A new dialog opens up to select the name of the network and choose the desired coordinate system for the newly created network.

- Zoom bar: This bar handles zooming in and out in the map panel. This action can also be done by zooming in using a trackpad or a mouse.
- Zoom default: Returns the map zoom and the zoom slider to the default value. The user can change this default value in the settings.
- Settings: By clicking on settings, a dialog appears where the user can change the default zoom value and the default map center, which is where the map zooms when opening the application. Initially, the zoom value is 14 and the map center is Munich. When working on a specific place on the map, setting this as the zoom center makes it faster to start editing after opening the application again in the future.

3.2 Side Panel

On the right of the editor exists the side panel with information regarding the currently loaded network. Inside it, there is a panel containing the general network information and an accordion panel containing information and actions regarding nodes, links and network validation/cleaning. In greater detail:

- Nodes: This panel contains a table with information about the nodes existing in the network. This information is the name/id of the node, the latitude and longitude of the node, as well as its in- and out-links if they exist. A user can click on a node on this table to select it. Afterwards, the edit and delete actions are also available for the selected node.
  - Edit: Opens up a dialog that allows the user to edit the node id, x and/or y coordinate of the selected node. The changes can be reflected both in the table and the map
  - Delete: Deletes the selected node from the network, the side panel and the map.
- Links: This panel contains a table with information about the network's links. That includes the link's name/id, from and to nodes, length, capacity, free speed, number of lanes, allowed modes and flow capacity of the link. This information is in accordance with MATSim. The table also contains the edit and delete functionalities for a selected link.
  - Edit: Opens up a dialog that allows the user to edit the link id, length, capacity, free speed, number of lanes, allowed modes and flow capacity of the selected link. The from- and to-node of the link cannot be changed.
  - Delete: Deletes the selected link from the network, side panel and the map.
- Validation & Cleaning: In this panel, there is the option to run validation and cleaning on the network.

– The validation checks through the elements of the network and detects possibly problematic ones that could create issues when running MATSim. These could be nodes not connected to other nodes (also known as "dangling" nodes), link attributes that are out of boundaries or bidirectional links with different attributes. After the validation is run, the nodes and/or links concerned are shown in the table, and the user can then edit or delete these links and nodes if needed.
– The network cleaning feature automatically removes the problematic nodes and/or links from the network. This functionality is part of the MATSim library, and mostly solves the problem of disconnected subnetworks. After running the cleaning function, the currently loaded network is saved in a file, the clean network is written in a new file and then loaded on our map along with all its information.

3.3 Map view

The main panel of the editor contains the map view of the application. The user can zoom in and out and click-and-drag to move to places within the map. Additionally, by left-clicking on the map, a new node is created and added to the network. Double-clicking on that node deletes it from the network. To create a link between two nodes the user needs to right-click on the first node, then right-click on the second node. The link creation dialog (Figure 3) appears for the user to set the attributes of the link. In a later section, we will present concrete examples on the step-by-step use of the editor.

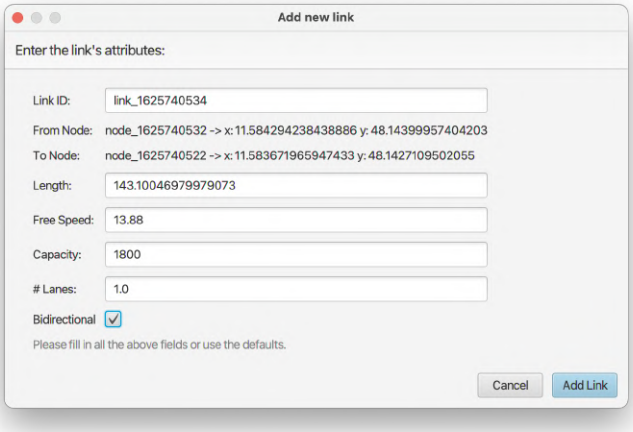

Figure 3: New link creation dialog.

3.4 Bottom Panel

The bottom panel contains information about the coordinates of the cursor on the map and the events happening on it, e.g. the creation and removal of nodes, and other clicks on the map and the node markers.

4 Classes

In this section, we will provide an overview of the classes of the application and their connections with each other and the features we have already discussed in the previous sections.

There are six classes in the application, namely the App, AppLaunch, MainController, NetworkInformation, ValidationTableEntry and ExtendedNetwork classes. *App* contains the "main" function and code to launch the main window of the javafx application. The fxml file and the connection to its controller are also in the *App* class. This fxml file contains the visual elements of the interface, and the *MainController* class has the code controlling them. The *Applauncher* class is basically a helper class that calls on the "main" function of the *App* class.

As mentioned above, the *MainController* class contains the code side of all the visual elements and is an integral part of the application. Each button, box and panel of the fxml file has its respective object here, and all the functionalities needed for each of them is implemented in this class. These functionalities include initializing the map view and all other visual elements, creating listeners for the clicks on the map view and the tables of the side panel, editing and deleting of nodes and links, running the validation for the network, opening an existing network file or creating a new network, cleaning of the network etc.

The *NetworkInfo* class also contains code for some visual elements, specifically for the labels displaying the information about the network on the side panel. Labels are used since the network information is not set to be directly modifiable. The name of the network, the number of nodes and links, the capacity, the lane width and the cell size of the network are the information included, according to the MATSim depiction of a network. Therefore, the class contains labels with the names of the attribute and labels with the respective values, as well as a list of the pairings between them, and functions to clear the labels and to update them with the new network information. This information is provided either from the loaded network file or set to default values for newly created networks.

The *ValidationTableEntry* class was created to show both nodes and links in a unified way in the validation and cleaning table of the side panel. Each object of the class contains an object - it can either be a *Node* or *Link* - the id of said object, and a respective message to show for the node/link. In the *MainController* class, when the validation function is run, for each possible issue, a new *ValidationTableEntry* object is added in a common list of issues, along with its respective descriptive message, and these objects are then visualized in the

validation and cleaning table.

The *ExtendedNetwork* class is a wrapper for the MATSim type *Network* with additional elements to connect with the visuals of the application. It contains a *Network* object, the path to the network file, the javafx boxes for the contents of the side panel and the respective tables, a *NetworkInfo* object, the coordinate system that the network should be stored in, a list with the validation issues and two Maps linking *Node* objects to visual markers on the map and *Link* objects to lines on the map respectively.

A lot of important functions are also implemented in this class. The paintToMap() function updates the node and link tables and creates the visual elements for them on the map. The table views are also initialized here, meaning the columns are set, along with their names and the type of information each column is to be filled with. The functions responsible for adding, editing and deleting nodes and links from the network are also implemented in *ExtendedNetwork*, including adding and removing the markers from the map and updating the pairing of objects to markers and lines. There are also helper functions necessary for these actions, like the creation of ids, finding a node by its coordinates, finding if a link exists in a network using its starting and ending nodes, clearing the information tables etc. To create the node and link ids in specific, the system time is used, in an effort to make the newly created ids unique.

5 Use cases

In this section, we present a few use cases to give a clearer idea of the editor's usage.

5.1 Scenario 1: Create a new network for the center of Munich using the DHDN_GK4 coordinate system

We first click on the "Create Network" button.

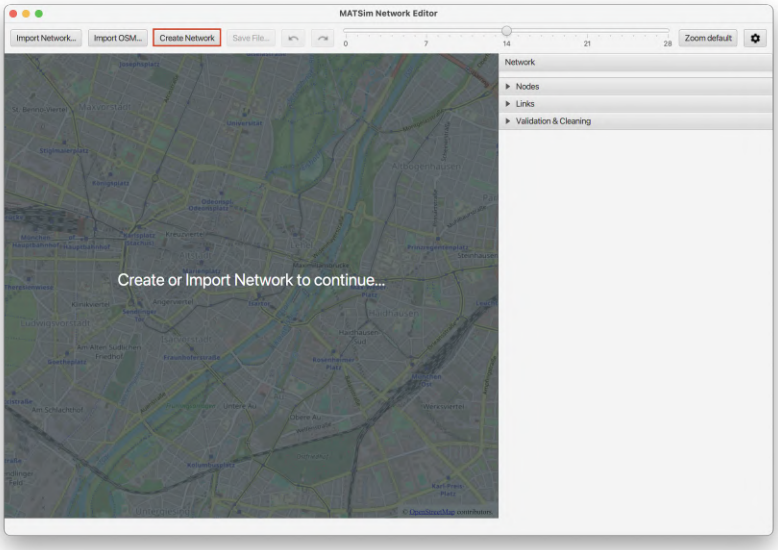

Figure 4: Selecting create network at the start screen.

A new dialog appears and prompts us to pick a coordinate system.

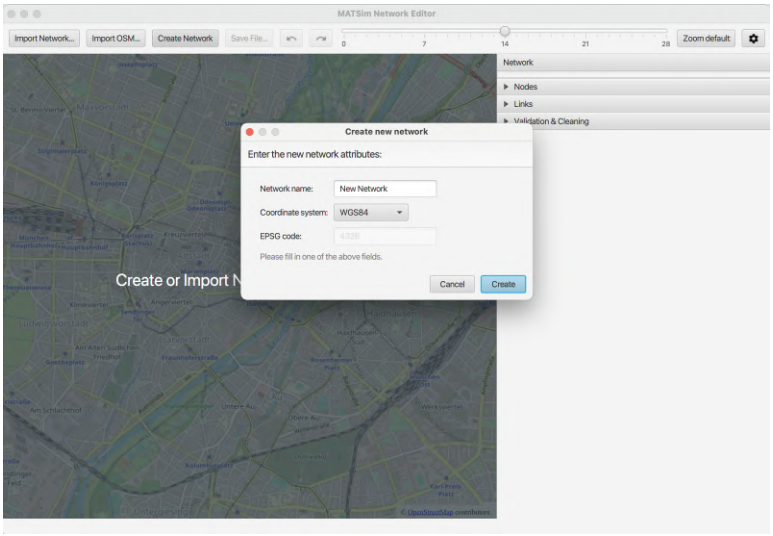

Figure 5: Coordinate system option dialog.

Then, we give our network an appropriate name and select our desired coordinate system from the drop-down. Lastly, we click on the "Create" button.

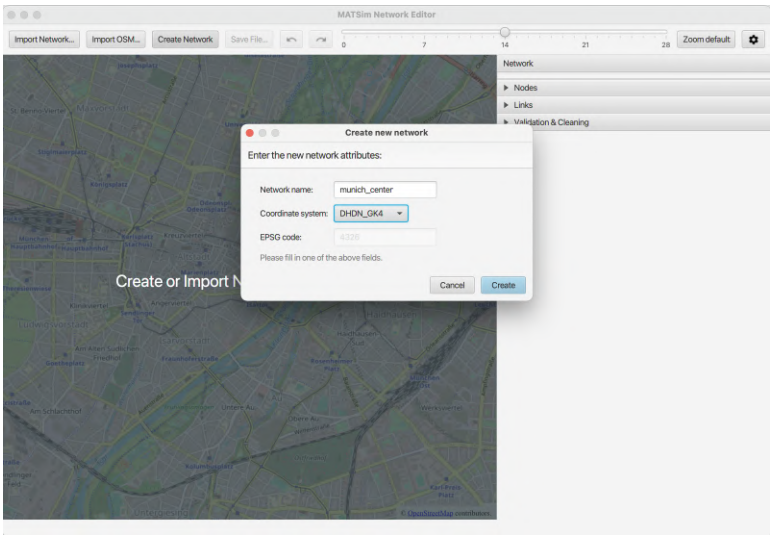

Figure 6: Options dialog after renaming and choosing coordinates system.

We can now see the details of the new network that we have created on the network information panel on the right of the editor.

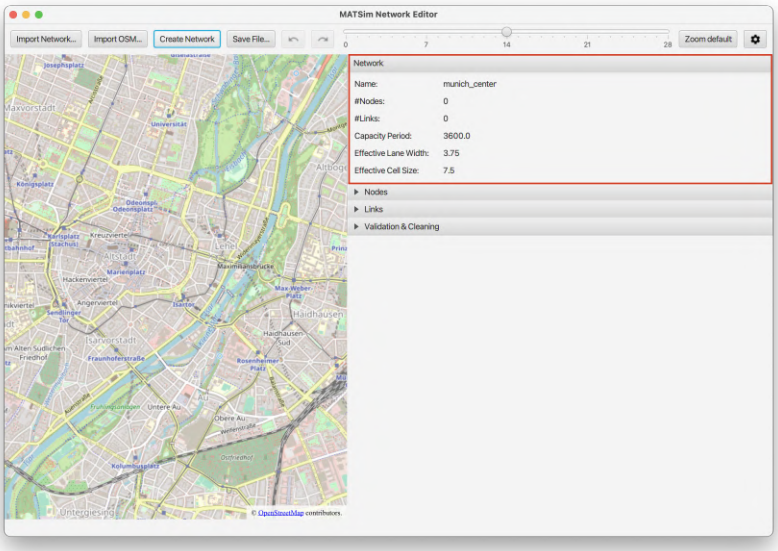

Figure 7: The editor after the new network is created.

5.2 Scenario 2: Import network from a standard MATSim format file

To import the network, we click on the "Import Network" file.

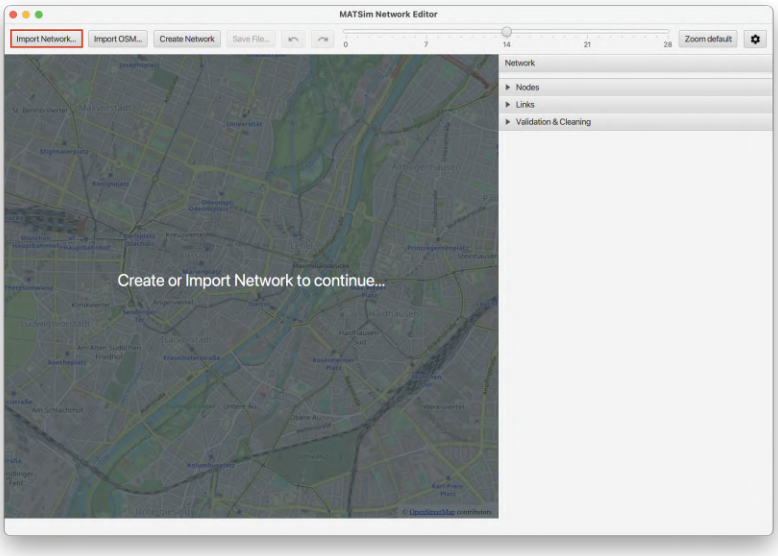

Figure 8: Selecting import network at the start screen.

A dialog appears for us to choose the coordinate system of the file to be opened. This is information that we should already have for the file we are trying to open so that the editor can correctly transform the coordinates into the coordinate system that it internally uses.

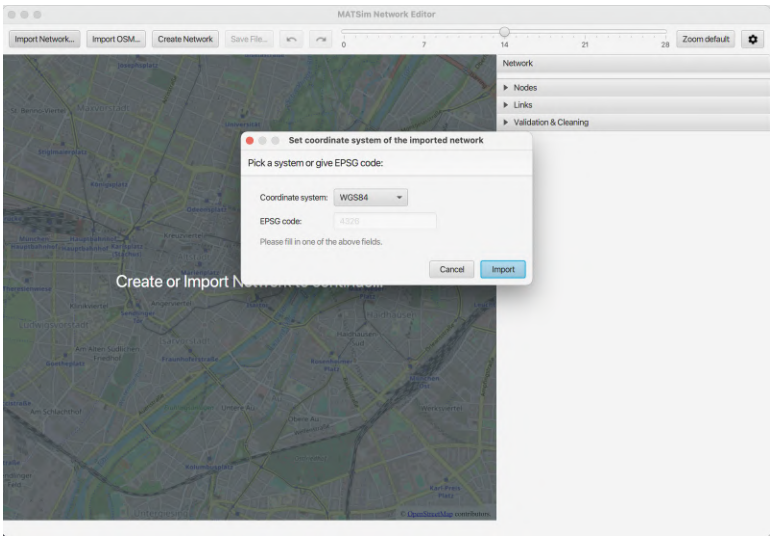

Figure 9: Dialog to choose the coordinate system of the file to be opened.

The file selector of our computer opens, and we select the file we desire to import the network from. This can be either a .xml or a .gz file, following the MATSim-compatible format as shown in Figure 1.

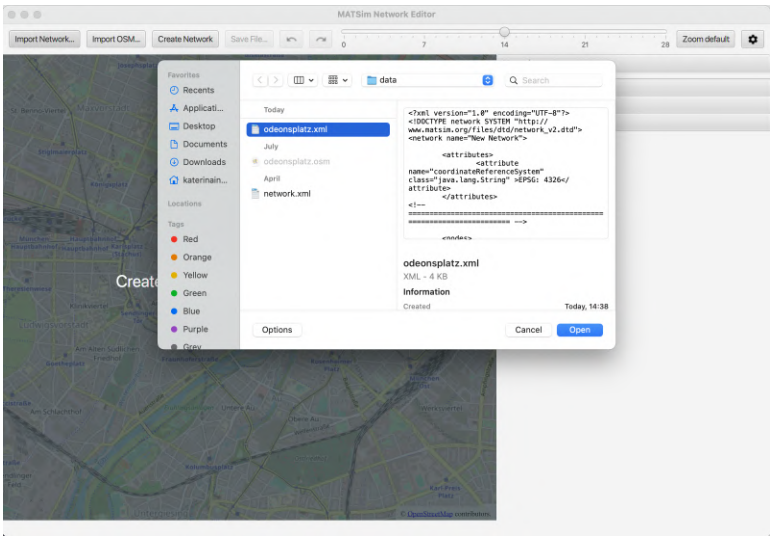

Figure 10: Prompt to choose a .xml or .gz file.

After we select the file and click on "Open", the network information is loaded, the nodes and links appear on the map and the side panel and we can

now edit, add and delete them as needed.

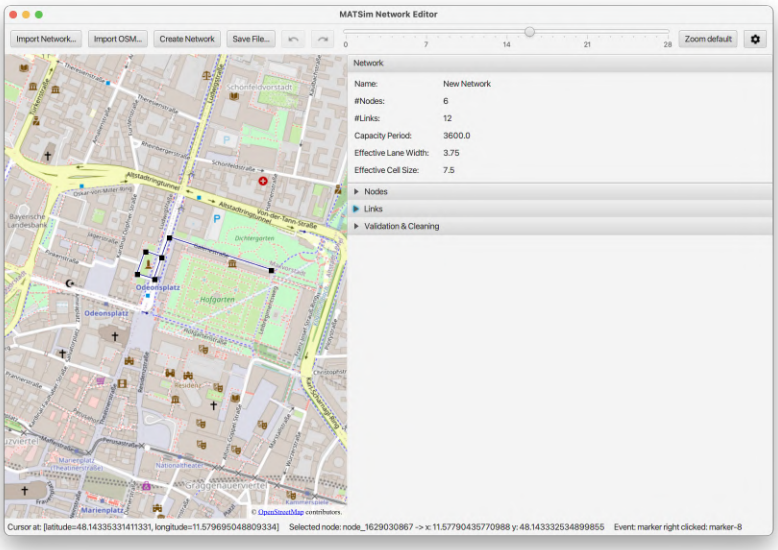

Figure 11: The network editor after a file is selected for importing.

5.3 Scenario 3: Import network from a .osm file

To import a network from a .osm file, we click on the "Import OSM" file.

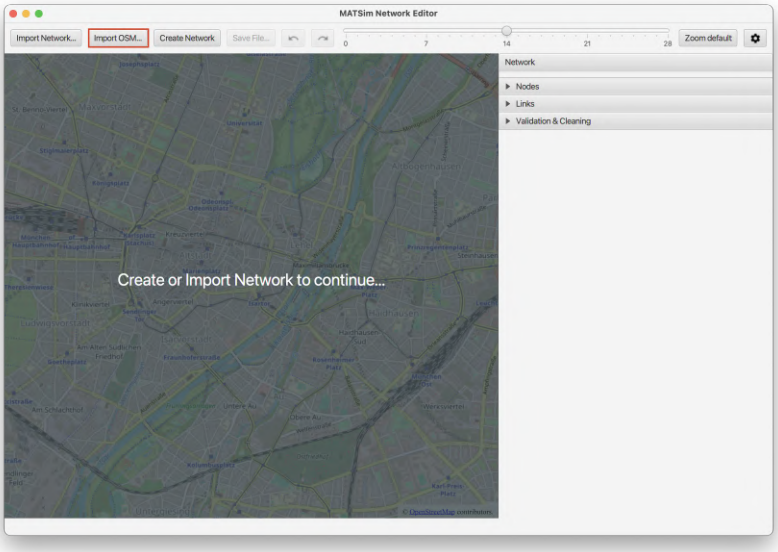

Figure 12: Selecting import OSM at the start screen.

Again the coordinate system dialog appears for us to choose, similarly to importing from a .xml file.

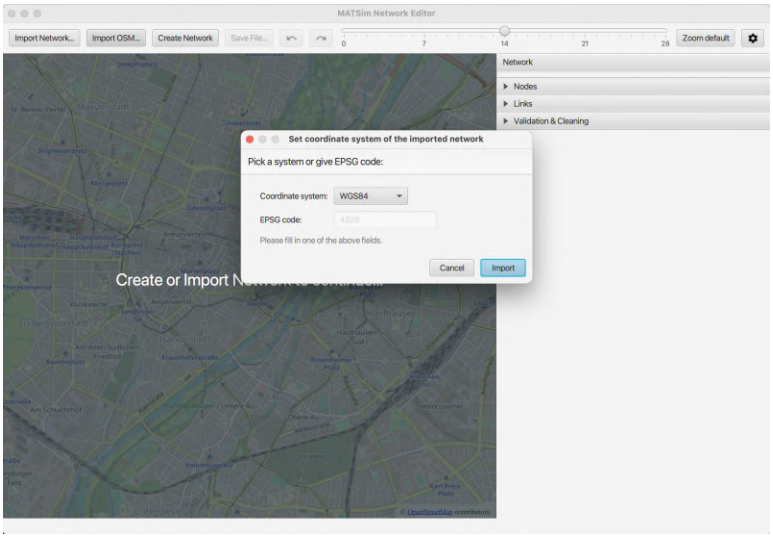

Figure 13: Dialog to choose the coordinate system of the file to be opened.

The file selector of our computer opens, and we select the file we desire to

import the network from. In this selection, .osm files are allowed.

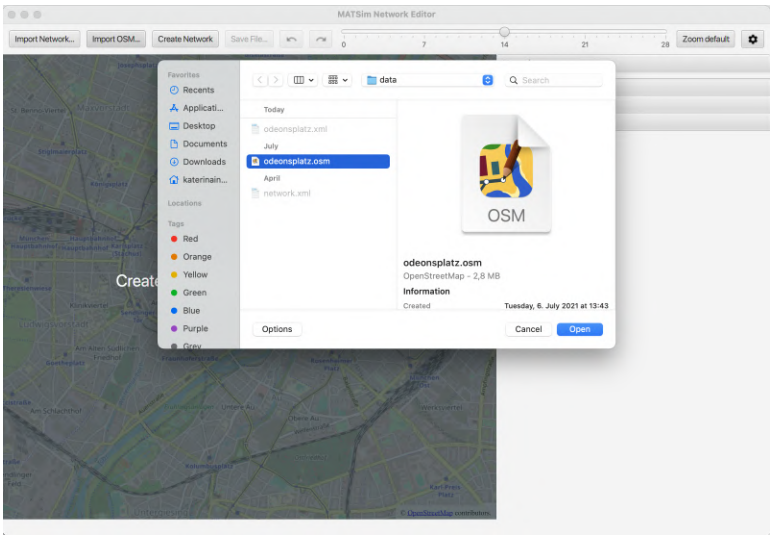

Figure 14: Prompt to choose a .osm file.

The system transforms the information into the MATSim format for networks and loads the information, drawing the elements on the map and displaying the information in the side panel.

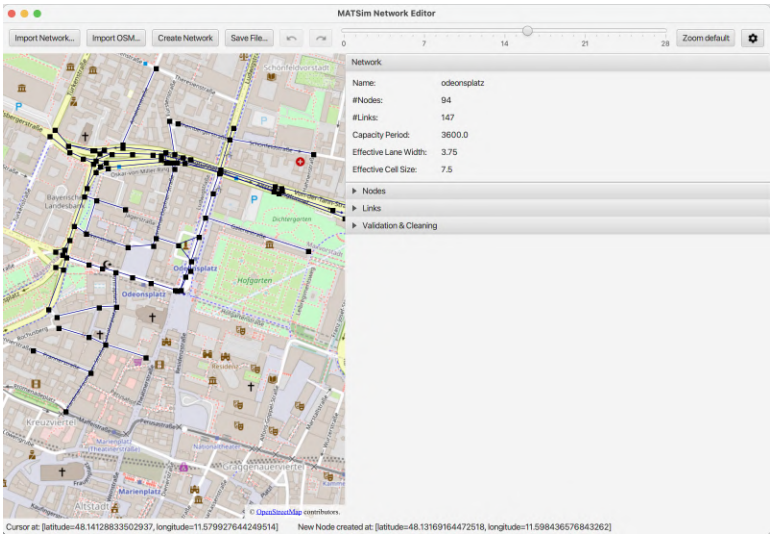

Figure 15: The network editor after the .osm file is selected for importing.

5.4 Scenario 4: Adding two nodes and connecting them with a link

Assuming a newly created network, we left-click on the map, at the point where we want to add the node.

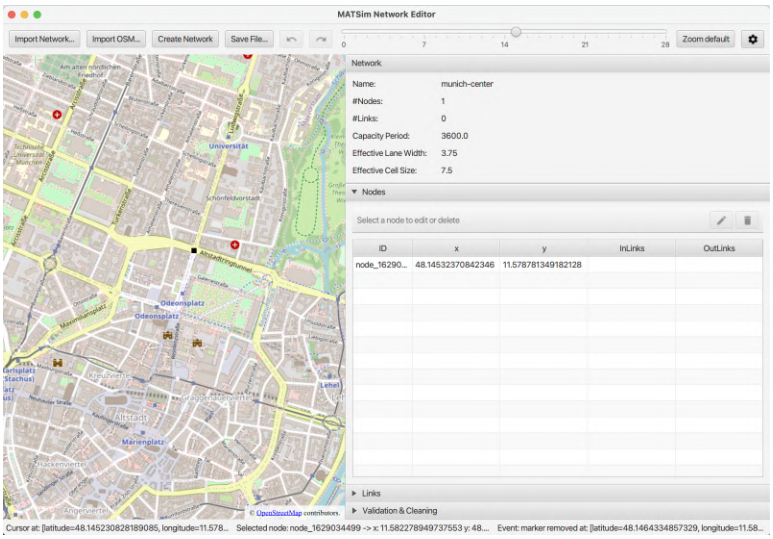

Figure 16: Creating first node on the map.

We do the same for the second node.

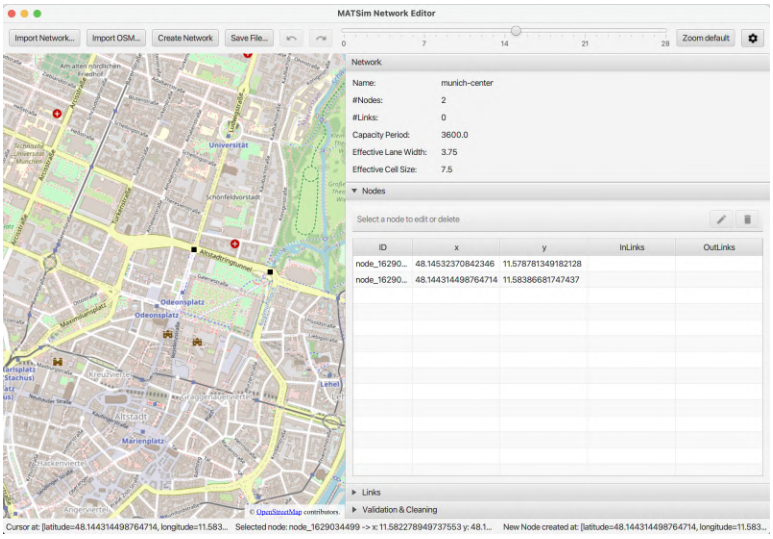

Figure 17: Creating second node on the map.

We right-click on the first node, then right-click on the second node to create the link.

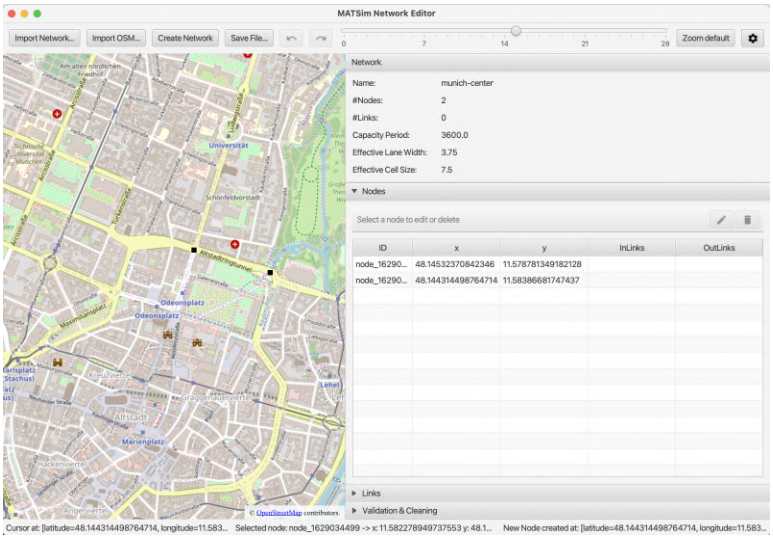

Figure 18: Creating second node on the map.

The link creation dialog appears and we decide to make the link non-bidirectional, then click on "Create".

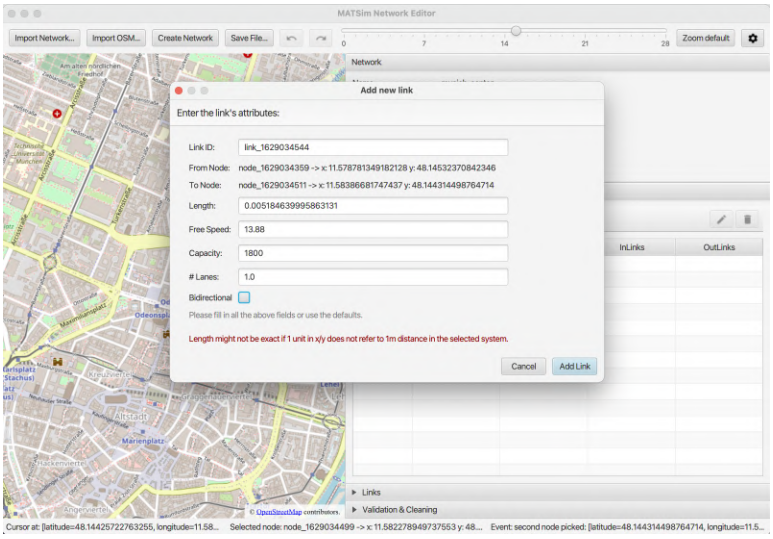

Figure 19: Creating second node on the map.

The link now appears on the map.

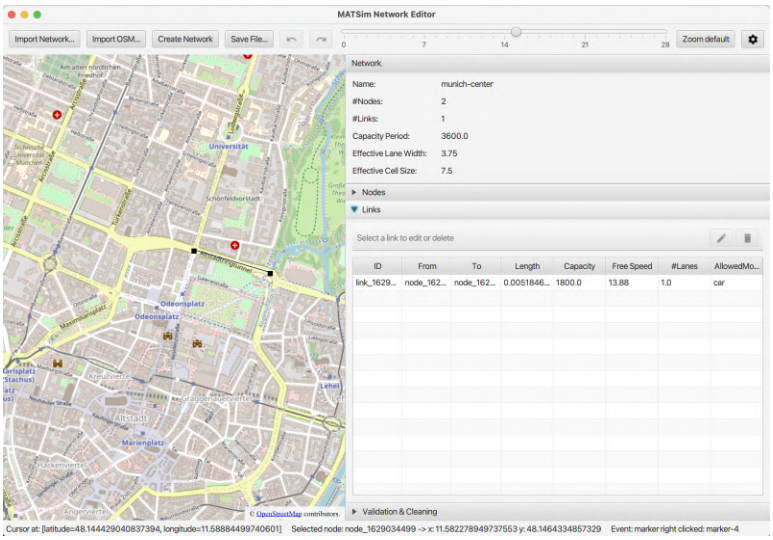

Figure 20: The editor after the creation of the link.

5.5 Scenario 5: Editing, then deleting a node

To edit the node, we first click on the node on the "Nodes" side panel to select it.

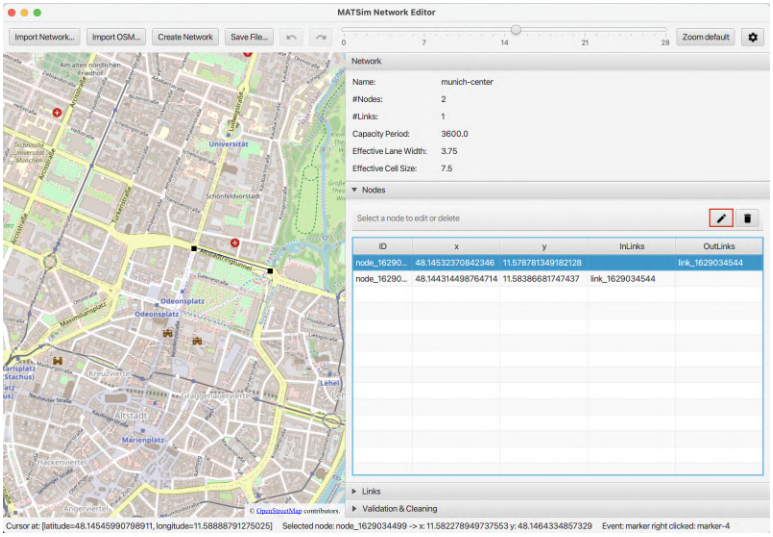

Figure 21: Node selected to edit.

We click on the pencil icon on the top right to edit the node and the node edit dialog appears.

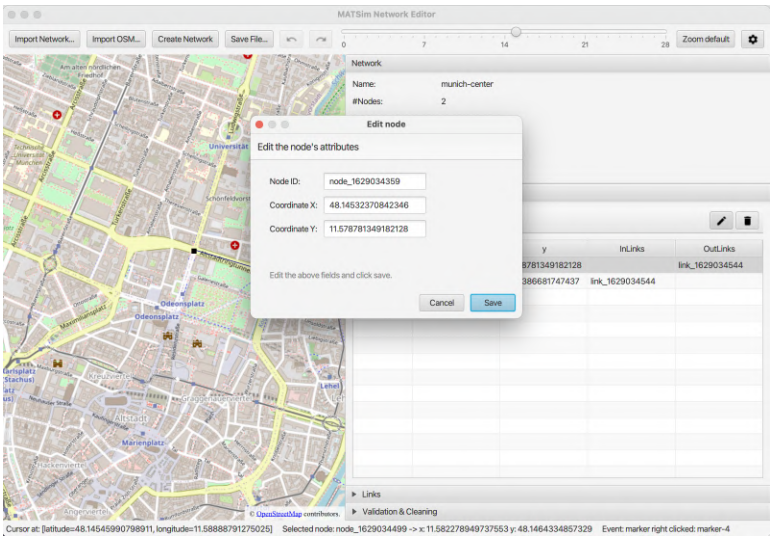

Figure 22: The editor with the node editing dialog open.

Let's say we want to change the name of the node. We type the new name and we then click on "Save".

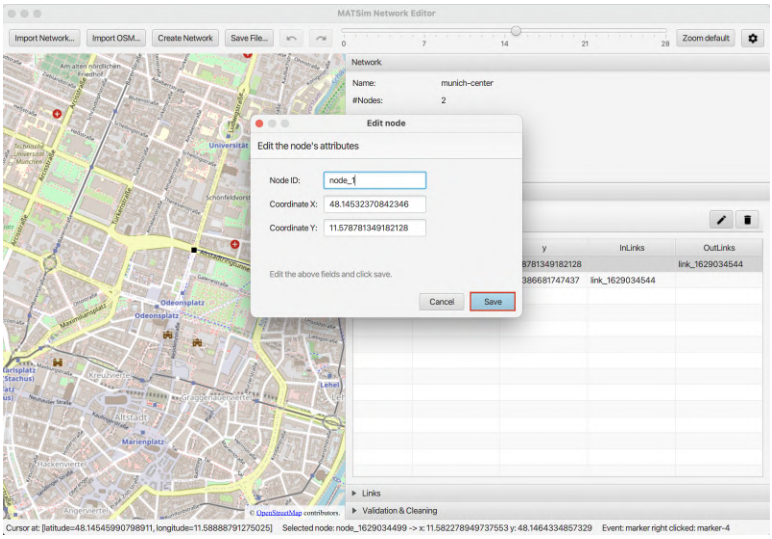

Figure 23: Node editing dialog after the name change.

We can now see the name of the node is changed on the side panel.

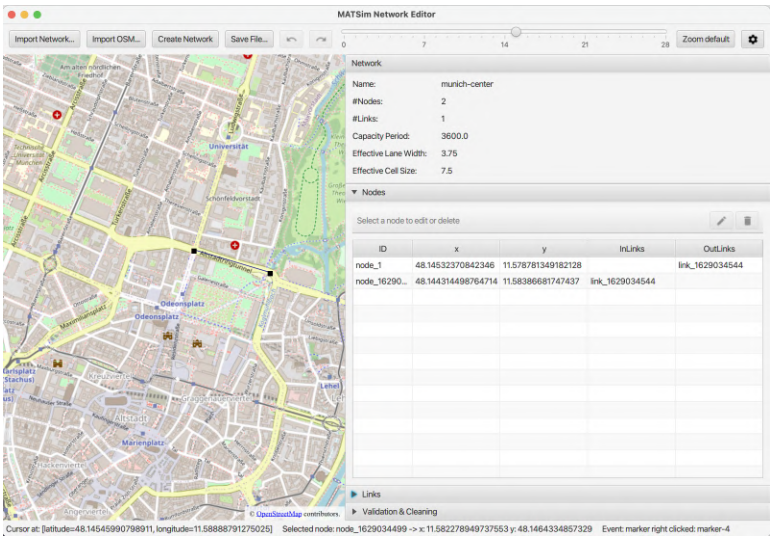

Figure 24: The editor after the editing of the node name.

We then decide we do not need this node and try to delete it. To do that, we click on the node on the side panel to select it, as we did before, then click on the bin node on the top right.

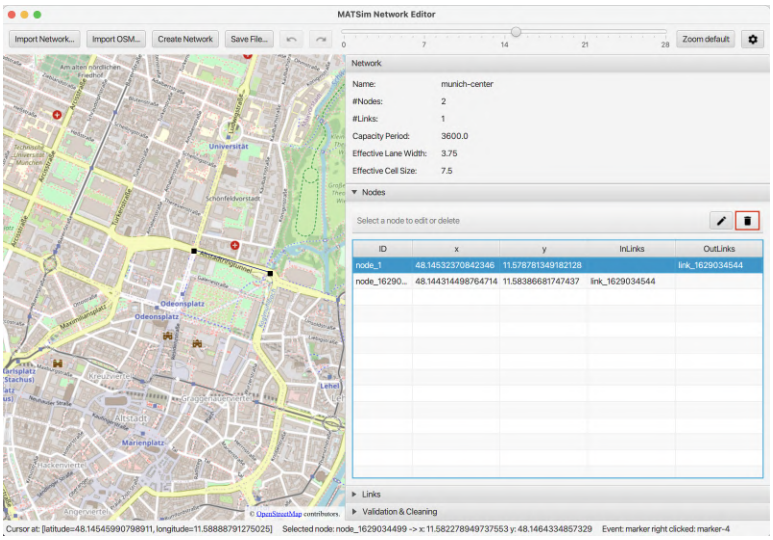

Figure 25: Select the node to delete from the network.

The node is now removed from the map and the side panel.

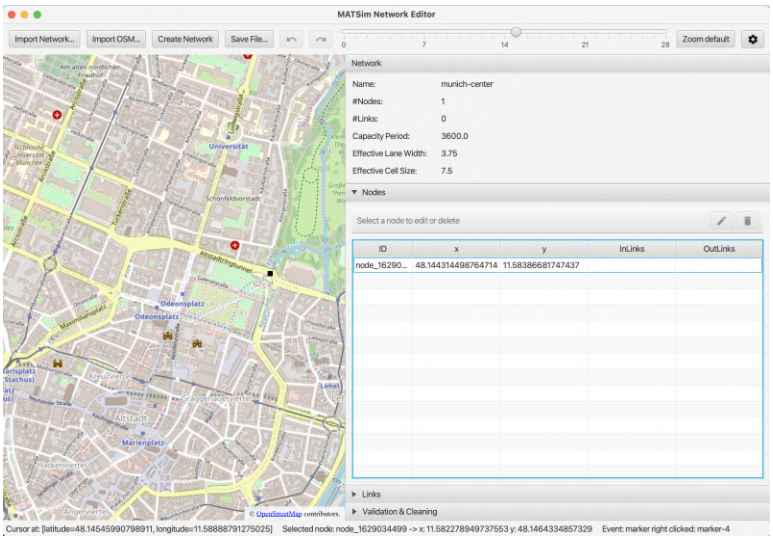

Figure 26: The editor after the node is deleted.

In the same way, we can edit and delete a link from the "Links" side panel.

5.6 Scenario 6: Saving a file

After we have finished working on our network, it is time to save it. To do that, we click on the save button on the top panel of the editor.

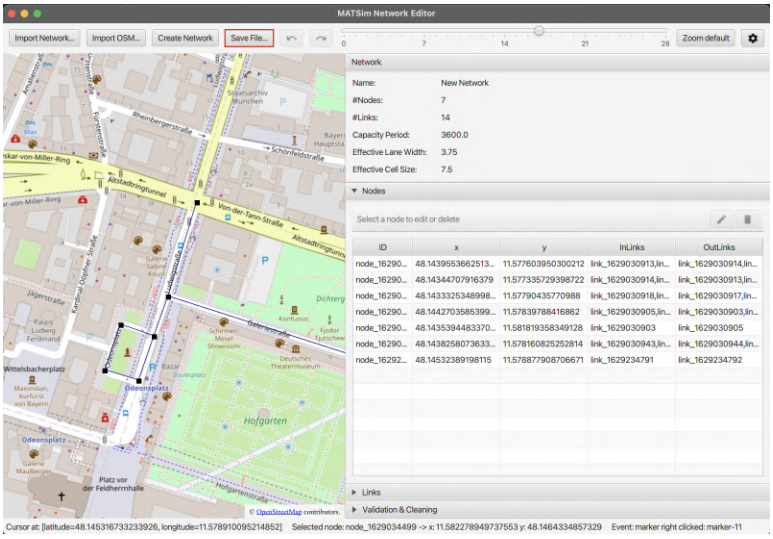

Figure 27: The editor before we save the network

.

The file selector of the operating system appears, we change the name of our file, and click on "Save".

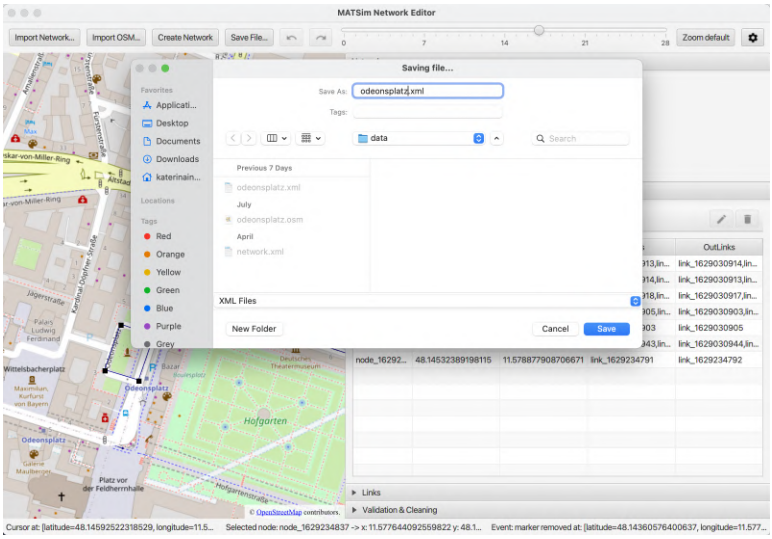

Figure 28: The file selector dialog to save the file.

5.7 Scenario 7: Run validation

To validate that our network can be used in a MATSim simulation, we click on the tick mark button on the "Validation & Cleaning" panel on the right.

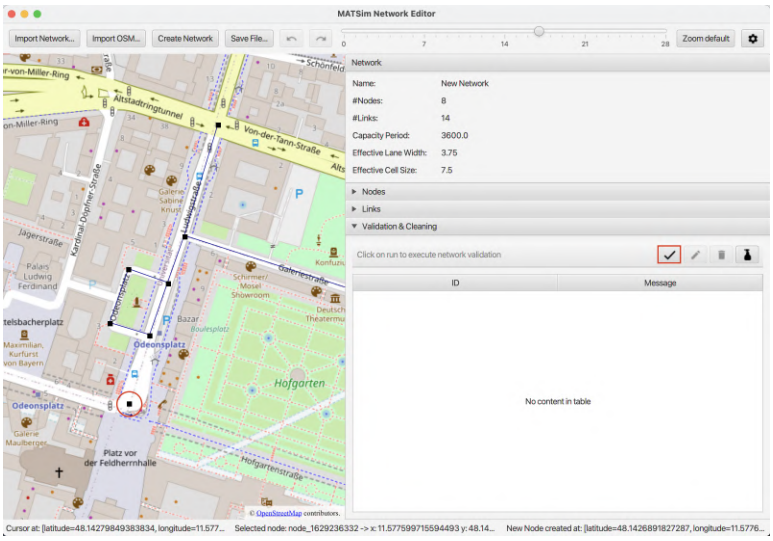

Figure 29: The network containing a dangling node, before the validation is run.

Any possible issues appear now in the panel.

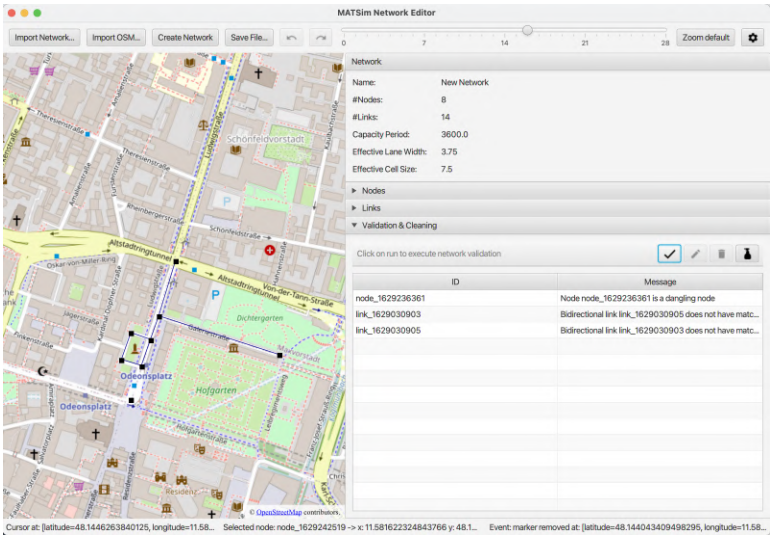

Figure 30: The editor containing the validation results.

Now we can edit or delete the nodes and links involved as we've seen before.

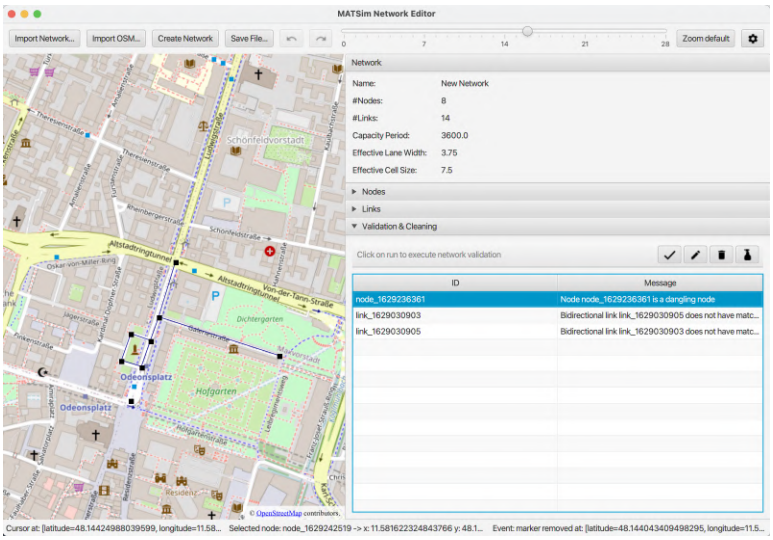

Figure 31: Result items of the validation can be edited and deleted.

5.8 Scenario 8: Clean a network

To clean our network, we click on the spray bottle button on the "Validation & Cleaning" panel on the right.

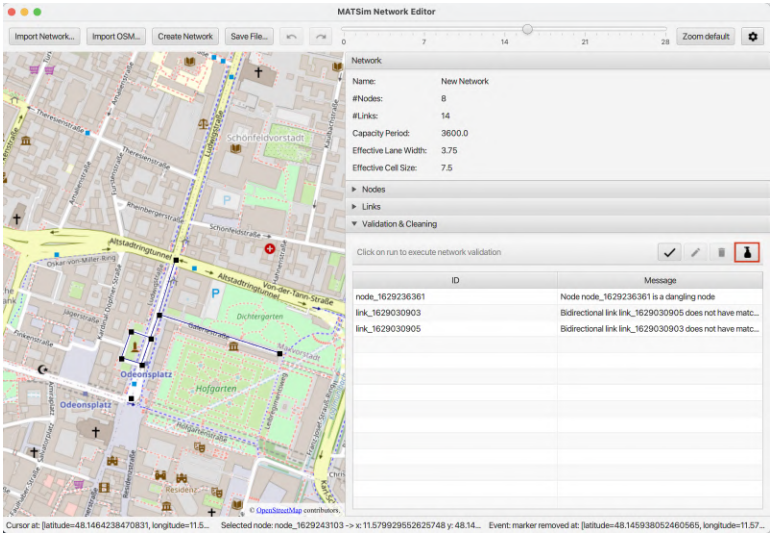

Figure 32: The editor before the network cleaning.

The cleaner starts processing the network and after a while, the cleaned

network replaces the current network on the map.

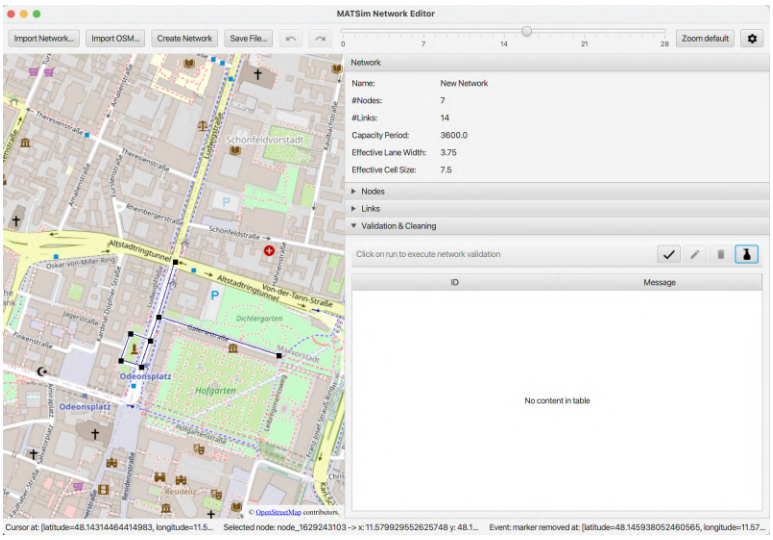

Figure 33: The editor after the cleaning. The dangling node has been removed from the network.

6 Future work

In the future, the editor can be enhanced with further functionality for efficiency and a more comfortable experience in creating and editing networks. These functionalities and ideas are mentioned in this section.

As mentioned in a previous section, there are undo and redo buttons in the control panel on the top of the editor. As of now, these buttons are not connected to the respective functionality. It would be beneficial to implement an undo and redo function to lessen the risk of making detrimental mistakes while editing the network.

Another important functionality to be added in the editor is highlighting the nodes on the map when they are selected from the table in the side panel and vice versa, to be able to easily identify nodes and links.

In further versions in the future, a selection mode could also be implemented, to be able to move and drag nodes and links on the map, without adding, editing or deleting them. Additionally, the implementation of the whole application could be enhanced by using a store-on-disk concept, to accommodate working on very large networks.

References

[1] Andreas Horni, Kai Nagel, and Kay Axhausen, editors. *Multi-Agent Transport Simulation MATSim.* Ubiquity Press, London, Aug 2016.