SciGRID

Open Source Transmission Network Model

USER GUIDE

V 0.2

AUTHORS:
W. MEDJROUBI & C. MATKE

NEXT ENERGY

EWE-Forschungszentrum

für Energietechnologie e. V.

www.scigrid.de

Last Update: 20.11.2015


SciGRID

Contents

1 Licence

This document is part of SciGRID release V0.2.

SciGRID is free, open-source code and open data and builds on OpenStreetMap data. The OpenStreetMap data is available under the Open Database License (ODbL) [1] and OpenStreetMap cartography is licenced as CC BY-SA. For more information on the copyright of OpenStreetMap, please refer to [2]. All data and databases delivered with the **SciGRID** model are made available under the Open Database License [3]. Any rights in individual contents of the database are licenced under the Database Contents License [4] (see [5]). You can also redistribute and/or modify the data distributed with **SciGRID** under the same licences and copyright.

The **SciGRID** code and this documentation is licenced under the Apache License, Version 2.0. [6]. Please visit the webpage [6] for more information concerning the Apache License and for a description of the terms under which you can use the **SciGRID** code.

Limitations of liability: in no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any contributor to **SciGRID** be liable to any user for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such contributor has been advised of the possibility of such damages.

2 Introduction

2.1 Motivation and aim

Details of electrical transmission networks are currently integrated in a number of in-house energy system models which are not publicly available. The structure, assumptions, simplifications, and the degree of abstraction involved in the transmission network models used are, hence, unknown and often undocumented. Further, there is hardly any (scientific) discussion on the underlying approaches, procedures, and results. This implies, that the learning curve in the construction of (electrical) grid models is rather low. At the same time, the output of energy system models takes an important role in the decision-making process concerning future sustainable technologies and energy strategies. Recent examples of such strategies are debated and discussed for the *Energiewende* in Germany.

In this context, the availability of transmission network data and models is a critical and urgent issue which has to be addressed in order to allow the involvement of an increasing number of actors in the energy sector decision-making. Furthermore, transmission network models are important when dealing with issues concerning cross-border congestion management, transmission capacity, grid stability and extension, to cite a few examples.

In this framework, the project **SciGRID** initiated by the research center NEXT ENERGY [7] aims at building an open source model of the electrical transmission network in Europe. The idea behind making both the **SciGRID** model and data available in the open source domain is that the energy modeling sector lacks reliable data on transmission networks. Data available under appropriate (open) licences ensures that established models and the assumptions they incorporate can be published, discussed, and validated in a well-defined and self-consistent

manner. In addition, the methods which are developed for building the model will be published under suitable licences, which is expected to foster and improve existing in-house models.

The main purpose of the **SciGRID** project is to open the door to new models and ideas in energy system modelling by providing freely available and well-documented data on the European electrical transmission networks. The open source philosophy offers a high level of transparency as the involved assumptions and simplifications are open to discussion and criticism. Such a discussion and the availability of documented approaches and methods can act as a motivation for more communication and scientific scrutiny in the construction of grid and network models. Furthermore, the **SciGRID** model will also be a benefit for existing in-house models, as its tools and results can be used as both reference implementation and reference data.

2.2 SciGRID project details

Project title:	Open Source Reference Model of European Transmission Networks for Scientific Analysis (Offenes Referenzmodell europäischer Übertragungsnetze für wissenschaftliche Untersuchungen)
Acronym:	SciGRID (Scientific GRID)
Funding period:	September 2014 - August 2017
Sponsoring body:	Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung), Germany
Funding code:	03SF0471
Project partner(s):	NEXT ENERGY (individual project)
Project webpage:	www.scigrid.de

2.3 Technical approach

On the technical level, the **SciGRID** network model is mainly based on the raw transmission data available in openstreetmap.org [8] under the ODbL License [1]. The ODbL License offers the possibility to share derived and modified databases from the OSM database. The **SciGRID** project is not only the **SciGRID** abstraction code but also the datasets of the transmission network resulting from the abstraction. Both are made available and can be downloaded from the **SciGRID** project webpage [9]. Bellow are the details of the approach used in **SciGRID**.

Four steps are involved in the **SciGRID** model: OSM data download, power data filtering, data export to a database, and data abstraction (see Fig. 1). As a result, the vertices (nodes) and links (edges) of the transmission network are created and exported as `.csvdata` files. The relational database approach allows for a flexible and a modular structure of the **SciGRID** model. For example, only networks with transmission lines of a certain voltage level, managed by a certain network operator, or for a certain type of cables can be filtered and graphically displayed.

This is the documentation for version 0.2 of the **SciGRID** transmission network model. This document includes a detailed user guide of the **SciGRID** model as well as the assumptions and simplifications considered in building the model. Additionally, a detailed user guide of the **SciGRID** Graphical User Interface is included.

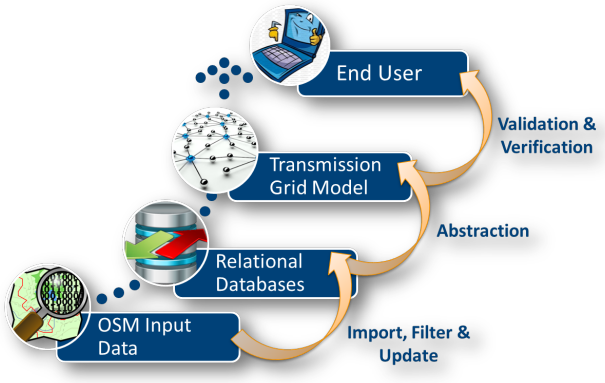


*Figure 1: Schematic representation of the different stages involved in the **SciGRID** model.*

The **SciGRID** model, user guide and data can be downloaded from the **SciGRID** webpage under the download section [9].

2.4 Data sources: OpenStreetMap

The **SciGRID** model is based on data available from the collaborative project OpenStreetMap (OSM). The data from OSM is licenced under ODbL, which states that also derived data can be published. This offers the possibility to share the **SciGRID** model along with its output data.

OSM is a collaborative project to create a free editable map of the world. OSM data is available on the webpage of OpenStreetMap [8] and is rendered directly as a map. The data can also be downloaded in an XML format, which follows a certain schema definition.

OSM data is based on three data types, called "data primitives", which are nodes (defining points in space), ways (defining an ordered collection of nodes that form linear paths and area boundaries) and relations (defining an ordered collection of nodes, ways, or other relations and provide logical or geographical relationships between OSM "data primitives"). The geographical locations of the nodes are defined by their latitude and longitude (see Fig.2). Datasets relevant for the power grid are filtered by using the "power" tag, which in OSM identifies a wide range of facilities and features related to the generation, the transmission and the distribution of the electrical power. The relations having the key/value with route/power are filtered out, their members (ways and nodes) identified and their geographical locations extracted (see Fig.2). The filtering is performed in an automated manner and the different steps it involves will be explained in this user guide. A map of OSM data showing in particular power-related details can be found on itoworld [10].

OSM data also offers the possibility to isolate the different components of the transmission network by using sub-tags associated with these components. Some practical examples are the tags: "substation" indicating electrical substations and "line" indicating transmission lines. Other sub-tags provide technical details about the components of the transmission network: the tag "voltage" indicates the voltage level(s) of a transmission line or a substation, the tag "cables" indicates the number of power-carrying cables represented by a transmission line, and the tag "tower" represents the towers carrying the electricity cables (see Fig.3).

To build the **SciGRID** model using OSM data, the transmission network components need to

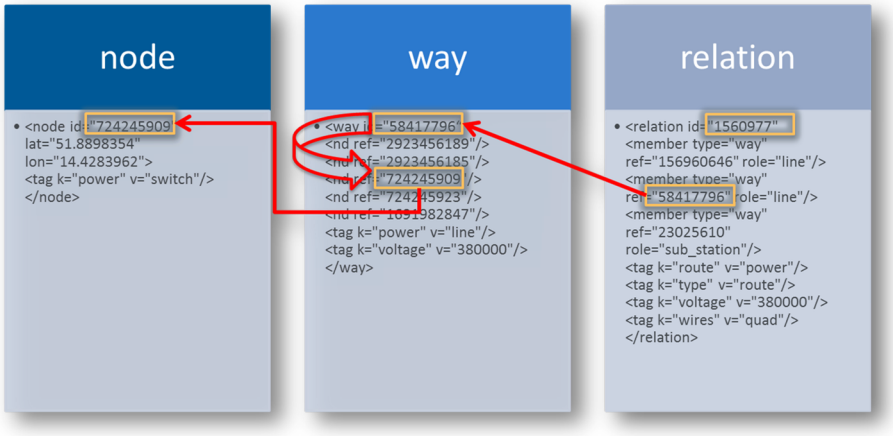


Figure 2: Schematic representation of the OSM data types. Relations with the "power" tag are chosen and also the subsequent ways and nodes belonging to the relations. The identification of relations and their members is done by using their "ID" tag.

```
<relation id="1637161">
    <member type="way" ref="245128569" role="line"/>
    <member type="way" ref="118942023" role="line"/>
    <member type="way" ref="30181272" role="line"/>
    <member type="way" ref="177829920" role="sub_station"/>
    <member type="way" ref="106684941" role="line"/>
    <member type="way" ref="23837631" role="sub_station"/>
    <tag k="cables" v="3"/>
    <tag k="color" v="white"/>
    <tag k="colour" v="white"/>
    <tag k="frequency" v="50"/>
    <tag k="from" v="Conneforde"/>
    <tag k="operator" v="TenneT"/>
    <tag k="ref" v="304"/>
    <tag k="route" v="power"/>
    <tag k="to" v="Diele"/>
    <tag k="type" v="route"/>
    <tag k="voltage" v="380000"/>
    <tag k="wires" v="double"/>
  </relation>
```

Figure 3: Example of the keys, values and tags available for a power relation in OSM. The relation displayed has the OSM-ID=1637161 [8], credits: ©OpenStreetMap contributors.

be accurately and correctly defined, isolated, and reproduced in a consistent and automatized fashion.

3 How to use SciGRID

3.1 Requirements

The **SciGRID** model scripts are developed and tested on Linux, but should also work with other UNIX systems. The different tools and software used in building the **SciGRID** model and their versions are listed below.

Operating system: Ubuntu precise (12.04.5 LTS)

osmosis version:	0.44
PostgreSQL version:	9.1.15 (64-bit)
PostGIS version:	1.5.3
osm2pgsql version:	0.82
pgAdmin III version:	1.14.0
GNU Make version:	3.81
GNU bash:	4.2.25(1)-release
Python:	2.7.3
QGIS version:	2.7.0-Master

For more information on how to install some of the software listed above, we refer to Section 7, and consult the file REQUIREMENTS available with the release folder.

3.2 Getting started

The **SciGRID** network model in its current version is based solely on raw OSM data. The structure of the **SciGRID** folder which can be downloaded from [9], is shown on Fig. 4. The **SciGRID** folder includes: the **SciGRID** abstraction code, the **SciGRID** GUI, the network data (vertices and links), OSM power data for Germany and this user guide.

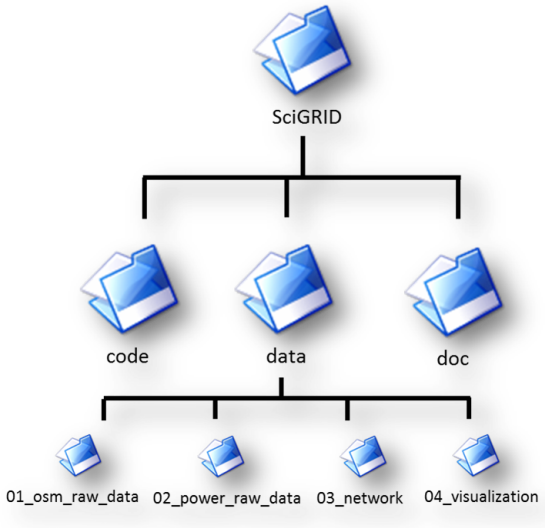


*Figure 4: The folder structure of the **SciGRID** model folder, which can be downloaded from [9].*

In the following the different steps involved in the **SciGRID** model are introduced in details.

3.3 OSM data download and filtering

The latest file containing the OSM data of the entire world, called "planet-latest.osm" can be downloaded for example from the website [11]. The term "latest" indicates that it is the latest available data of the whole planet. Older versions of the planet data are provided with a "date stamp" and are named as: planet-{date stamp}.osm. Other sources are available for downloading OSM data, for more information refer to [12]. The OSM data can be extracted using the java tool *osmosis* as smaller datasets , e.g. for a continent, a region, country or a city, depending on the user needs.

The OSM data is available under different formats, we have chosen to work with the `.pbf` (Protocolbuffer Binary Format) format [13]. The `.pbf` file representing the OSM planet data contains all information available in the OSM world map. This represents a huge number of nodes, ways and relations (On 27.10.2015 OSM planet data included 3 Billion nodes [13]).

The raw OSM data is filtered *thematically* using the "power" tag and *spatially* by only including the region for which the transmission network will be abstracted. As an example, in this user guide the region of interest will be Germany. However, the procedures introduced here can be adapted and applied for any other region, as long as the relation representation of the the transmission network is available. The *osmosis* tool used to filter the data with respect to the "power" tag will also be introduced.

OSM data download

The planet OSM data file is downloaded as `.pbf` file to the folder `SciGRID/data/01_osm_raw_data`. The size of the `.pbf` file is about 30 Gb (Status: November 2015). Depending on the internet connection speed, the user will need about two hours to download the whole planet OSM `.pbf` file. It is not necessary that the user downloads the planet OSM data file. In the **SciGRID** model folder, the resulting data extracted and filtered using *osmosis* is provided in the folder `SciGRID/data/02_osm_raw_power_data` under the name `de_power_151109.osm.pbf` (representing the filtered data status on 09.11.2015).

OSM data filtering: the "power" data

The tag "power" is assumed to represent all data necessary to build the **SciGRID** network model. Power data is represented by relations, ways and nodes having the "power" tag (either as a key or as a value). More precisely, it is assumed that the relations having a tag with the key "route" and value "power" cover all the available information about transmission circuits in OSM. For more information about the tags with the key "power" and how the transmission network is represented in OSM, please refer to the reference [13]. [1]

The OSM data with the "power" tag for the region "Germany", defined by a bounding polygon, is extracted using the command line java application *osmosis*. *osmosis* is a useful open source application which allows for OSM data manipulation and processing. For more information about *osmosis* and how to install it, refer to the web-page [14] or Section 7.1 of this user guide.

[1] Note that not all the transmission network details are covered by relations. This is the case for example for most of the European countries, where the relation representation of the transmission network is poor. In Germany however, the relation representation is widely used and covers a high percentage of the transmission network.

osmosis version 0.44 is used for the present version of the **SciGRID** model.

When filtering the `planet-latest.osm.pbf` file, *osmosis* stores a large amount of temporary data. In order to provide sufficient space, a temporary folder is set with the following command:

Listing 1: set osmosis tmp folder

```
export JAVACMD_OPTIONS="-Djava.io.tmpdir=osmosis_tmp_folder"
```

where `osmosis_tmp_folder` indicates the folder's path. If no folder is indicated, *osmosis* will store the data by default in /tmp.

The OSM data is filtered for Germany and for the data primitives containing a tag "power". The filtering is performed for the power tag (key=power, value=*) for nodes, ways and relations. In addition, the data is filtered by adding the power relations containing a tag (key=route, value=power).

This is accomplished by the following *osmosis* command:

Listing 2: Osmosis power data filtering

```

osmosis \
--read-pbf file=SciGRID/data/01_osm_raw_data/planet-151109.osm.pbf \
--tag-filter accept-relations route=power \
--used-way --used-node \
--bounding-polygon file=SciGRID/data/01_osm_raw_data/germany.poly \
  completeRelations=yes --buffer outPipe.0=route \
--read-pbf file=SciGRID/data/01_osm_raw_data/planet-151109.osm.pbf \
--tag-filter accept-relations power=* \
--used-way --used-node \
--bounding-polygon file=SciGRID/data/01_osm_raw_data/germany.poly
  completeRelations=yes --buffer outPipe.0=power \
--read-pbf file=SciGRID/data/01_osm_raw_data/planet-151109.osm.pbf \
--tag-filter reject-relations \
--tag-filter accept-ways power=* \
--used-node \
--bounding-polygon file=SciGRID/data/01_osm_raw_data/germany.poly \
  completeWays=yes --buffer outPipe.0=pways \
--read-pbf file=SciGRID/data/01_osm_raw_data/planet-151109.osm.pbf \
--tag-filter reject-relations \
--tag-filter reject-ways \
--tag-filter accept-nodes power=* \
--bounding-polygon file=SciGRID/data/01_osm_raw_data/germany.poly \
--buffer outPipe.0=pnodes \
--merge inPipe.0=route inPipe.1=power \
--buffer outPipe.0=mone \
--merge inPipe.0=pways inPipe.1=pnodes \
--buffer outPipe.0=mtwo \
--merge inPipe.0=mone inPipe.1=mtwo \
--write-pbf file=SciGRID/data/02_osm_raw_power_data/de_power_151109.osm.pbf
```

Note that the symbol \ at the end of each command line is a line-breaking symbol and is not part of the *osmosis* command. Note also that the "151109" used in the name of the database and the files throughout this user guide indicates the "date" stamp of the OSM data used. In this case the data used correspond to the 9th of November 2015.

The option "outPipe.0=name" produces a data stream and stores it temporarily under "name". This option is used four times in Listing 2 to temporarily contain the data resulting from the "power" filtering applied on relations, ways and nodes. As an example, the following command produces a data stream containing the relations having a "power" tag and temporarily stores it under the name "route":

Listing 3: data stream output

```
--read-pbf file=SciGRID/data/01_osm_raw_data/planet-151109.osm.pbf \
--tag-filter accept-relations route=power \
--buffer outPipe.0=route \
```

The same procedure is used for the nodes, ways and power relations data. To allow the pipeline processing to be run in parallel (on multiple threads) the option `--buffer` is used.

The different data streams produced when using the outPipe option are then merged together using the option `--merge`. As an example, the data resulting from filtering the power tagged relations (both power=* and route=power data) are merged as follows:

Listing 4: osmosis pipe merge

```
--merge inPipe.0=route inPipe.1=power --buffer outPipe.0=mone
```

The same procedure is repeated to merge all data streams and finally store the filtered power data in the file `de_power_151109.osm.pbf`

A detailed description of the *osmosis* commands used in Listings 2:

- `SciGRID/data/01_osm_raw_data` and `SciGRID/data/02_osm_raw_power_data` are the folders where the raw OSM planet data and the output of the OSM data filtering are stored, respectively.
- `--read-pbf` enables reading OSM data files in `.pbf` format. If the file format is .osm the option should be changed to `--read-xml`. However, it is recommended to use `.pbf` files. It is also possible to use the file formats .osm.gz or .osm.bz2. This is done by unpacking the file using a pipeline: `bzcat planet-latest.osm.bz2 | osmosis file=- ...`.
- `--tag-filter` option is used to filter elements (relations, ways and nodes) based on their type and optionally based on their tag values. It is useful to accept or reject elements that match the filter specification.
- The combination `--tag-filter accept-relations route=power` is used to include the relations with the key "route" and the value "power".

- To allow the filtering of ways having the key "power", the combination `--tag-filter accept-ways power=*` is used, where * means that any value is accepted. The same syntax is used for nodes and relations, using accept-nodes and accept-relations keywords, as in Listing 2.
- To export nodes which belong only to the filtered ways and relations, the option `--used-node` is used. This option guarantees that other nodes which do not belong to the filtered relations and ways will not be included. This reduces dramatically the number of exported nodes.
- `--buffer` allows the pipeline processing to be split across multiple threads. This is useful if multiple CPUs are available, as multiple tasks consume significant CPU time.
- `--outPipe.0=name` produces a data stream and stores it in the temporary file "name". For example, `--outPipe.0=route` stores the data stream under "route". Using pipes allows for merging multiple tag filters in one *osmosis* command. This is done by directing the data stream resulting from each tag filtering and then merging all streams two by two using the option `--merge`.
- The data is *spatially* restricted by using a `.poly` file for Germany. This is accomplished with the option `--bounding-polygon`. The `.poly` files consist of a list with arbitrary many points, which builds the exterior of a region of interest. The `.poly` file for Germany (`germany.poly`) is available in the `SciGRID/data/01_osm_raw_data` folder.
- Using `--completeRelations=yes` in combination with the previous bounding polygon option allows for including all available relations which are members of relations which have at least one member in the bounding box. This option also implies that the ways are also completed. This is important in the case where in a relation some relation members (transmission lines, substations) are in Germany but some members are outside of Germany. This is the case for example for cross-border transmission lines (see Figure 5), which extend between two countries. The option `--completeRelations=yes` guaranties that all relation members are exported even if some of them are "physically" outside of the bounding box.
- The option `--write-pbf` allows to write the results of the filtering into the indicated `.pbf` file.

After executing the *osmosis* command in "Listing 2, the resulting data file containing the filtered "power" OSM data is labelled `de_power_151109.osm.pbf` and is stored in the `SciGRID/data/02_osm_raw_power_data` folder. A copy of this file is provided with the **SciGRID** model folder. The dataset includes only relations, ways, and nodes having the key "power" for the region Germany. This data will be labelled "power data" throughout this user guide.

For more information about other options available in *osmosis*, please refer to the OSM wiki webpage [13] or type the command `osmosis --help` in a shell terminal.

Using bounding box for OSM data filtering

When filtering for a certain region, the user can make use of a bounding box, which is less precise than a `.poly` file in defining borders. This is done by using the *osmosis* option `--bounding-box`

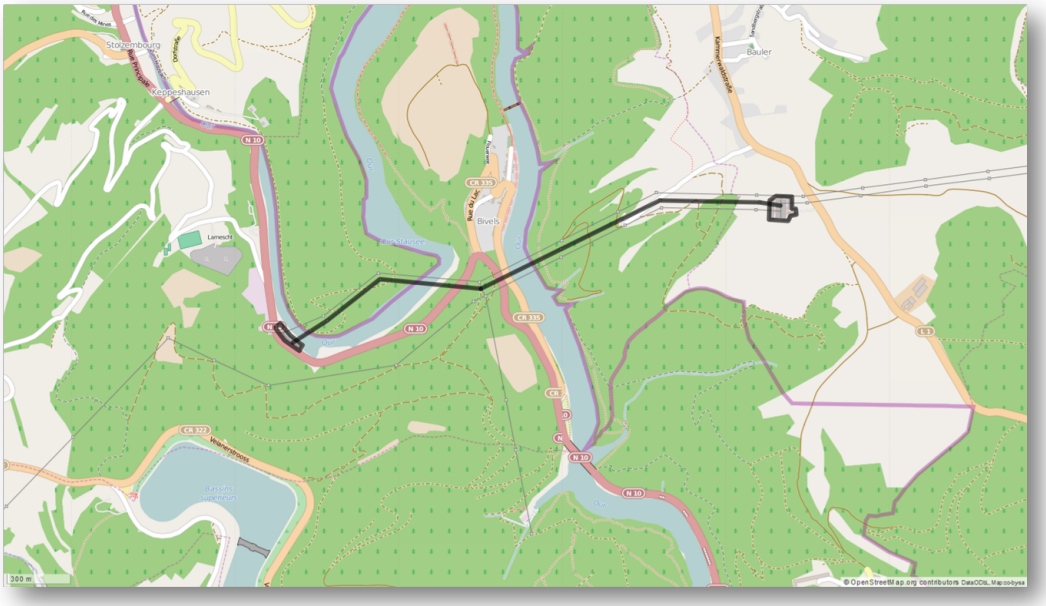


Figure 5: Example of a relation representing a cross-border transmission line. One transmission substation and the transmission lines of the relation are in Germany (right side of the figure) and the second transmission substation is in Luxembourg (left side of the figure). Credits: ©OpenStreetMap contributors.

instead of `--bounding-polygon`. The user needs to indicate the latitudes of the top, bottom bounding box, and the longitudes of the left and right edges of the bounding box; respectively. For Germany for example, the *osmosis* command using a bounding box will be:
`--bounding-box top=56 bottom=46 left=5 right=16`.

3.4 Power data export to the database

To store and analyse the "power data", it needs to be exported to a database. This is accomplished by using the command-line based program *osm2pgsql* [13], which converts OSM data to *PostGIS*-enabled *PostgreSQL* databases. The open source *PostgreSQL* [15] is an object-relational database management system. *PostGIS* [16] is a spatial database extender for *PostgreSQL* object-relational databases. *PostGIS* is very useful in the context of OSM data as it enables support for geographic objects allowing location queries to be run in *PostgreSQL*.

What is *osm2pgsql* actually doing? First, it is provided with a so called `.style` file, where all the settings of how *osm2pgsql* deals with the OSM raw data are stored. These settings indicate which columns are created for the tables containing the "power data" in the *PostgreSQL* database. It is also possible to define which information in the "power data" are ignored and will not be stored in the database. In the context of **SciGRID**, it is important to use a customized `.style` file, and not the default one available in the *osm2pgsql* folder because the latter does not consider the "power" related tags. A customised `.style` file called `power.style` is provided with the **SciGRID** model in the folder `SciGRID/data/02_osm_raw_power_data`.

In the following, the different steps involved in exporting the "power data" to the *PostGIS* enabled *PostgreSQL* database are described in details. Note that, the following steps are also included in the **SciGRID** makefile provided in the folder `SciGRID/code`.

- Step1: Create a *PostgreSQL* database:

```
createdb --username=postgres_user --port=postgres_port \
--host=postgres_host postgres_database
```

The name of the database is indicated as `postgres_database`. The options `--username`, `--host`, `--port` indicates the username, the server host, and the port number of the *PostgreSQL* server; respectively. The default values for the username, hostname and the port number are: postgres, 127.0.0.1 and 5432; respectively.

- Step2: Load the *PostGIS* object and function definitions into the database by loading the `postgis.sql` definitions file:

```
psql --dbname=postgres_database \
--username=postgres_user --host=postgres_host \
-f /usr/share/postgresql/9.1/contrib/postgis-1.5/postgis.sql
```

The option `-f ../../postgis.sql` indicates that a file containing *SQL* commands will be executed, in this case the file `postgis.sql`. Change the location of `postgis.sql` file, if it is different in your system, in the `default_config.mk` file provided in the **SciGRID** code folder. The `postgis.sql` file is located in Linux systems by default in the folder [prefix]/share/contrib). To find where the file `postgis.sql` is located in your system, run the following command in a terminal:

```
find . -name 'postgis.sql'
```

- Step3: Install the spatial reference system for *PostGIS*, necessary for a complete set of EPSG coordinate system definition identifiers. The `spatial_ref_sys`.sql definitions file has to be loaded and will populate the `spatial_ref_sys` table. This is done with the command:

```
psql --dbname=postgres_database \
--username=postgres_user --host=postgres_host \
-f /usr/share/postgresql/9.1/contrib/postgis-1.5/spatial_ref_sys.sql
```

This will permit the use of very useful *PostGIS* functions, for example the `ST_Transform()` operations on geometries. For more information about the *psql* command line and the available options, consult the *psql* section in the *PostgreSQL* documentation [15].

Change the location of `spatial_ref_sys.sql` file, if it is different in you system, in the `default_config.mk` file. To find where the file `spatial_ref_sys.sql` is located on your system, run the following command in a terminal:

```
find . -name 'spatial_ref_sys.sql'
```

- Step4: Create the "hstore" extension for the database. The "hstore" data type permits storing sets of key/value pairs within a single *PostgreSQL* value. This is useful in various cases, such as rows with many attributes that are rarely examined, or semi-structured data. Keys and values are simply text strings. This is accomplished by the command:

```
psql --dbname=postgres_database --username=postgres_user \
--host=postgres_host --port=postgres_port -c "CREATE EXTENSION hstore;"
```

The option `-c SQL_command` specifies that *psql* is to execute one command string, `SQL_command`, and then exit.

- Step5: Export the data in '.2adflstv]-[`de_power_151109.osm.pbf` to the database `de_power_151109` using *osm2pgsql*. This is accomplished by using the following command:

Listing 5: Osm2pgSQL data export to database

```
osm2pgsql -r pbf \
--username=postgres_user --host=postgres_host \
--port=postgres_port --database=postgres_database \
--style /data/02_osm_raw_power_data/power.style -s  -C cash_size_in_MB  \
--number-processes nb-processors \
/data/02_osm_raw_power_data/de_power_151109.osm.pbf
```

note that the symbol \ at the end each command line is a line-breaking symbol and is not part of the *osm2pgsql* command. The different options used in Listing 5 are explained in the following:

  - The first option `-r pbf` indicates the input-reader format, in this case the OSM binary format `.pbf` is used. The path and the name of the power data file to be exported is indicated on the last line of the command as: `SciGRID/data/02_osm_raw_power_data/de_power_151109.osm.pbf`. This file is provided in the folder `SciGRID/data/02_osm_raw_power_data`.
  - The database to which the data is exported is defined using `--database=postgres_database`, which in this case is `de_power_151109`, where "de" stands for Germany (or Deutschland).
  - The option `--style /data/02_osm_raw_power_data/power.style` indicates the location and name of the style file. If not set by the user, the default `.style` file location is `/usr/local/share/osm2pgsql/default.style`. As indicated above, to be able to correctly include the power data in the database a customised style file need to be used. In **SciGRID** an appropriate `.style` file named `power.style` is provided in the folder `SciGRID/data/02_osm_raw_power_data`.
  - To reduce the RAM usage it is useful to use the "slim mode" indicated by `-s` in the export command. The data is then stored temporary in the database, although the data export is slower.
    An important advantage of using the slim mode option is that the relations will be

exported to the database as well, which is not the case when using the default export mode. As the relations constitute the backbone of **SciGRID** model it is mandatory to use the slim mode option when exporting the "power data" to the database.

- When using the slim mode, the option `-C cache_size_in_MB` is mandatory. The default cache size is 800 MB of RAM.
- The user can specify the number of processors to run the data export in parallel by making use of the option `--number-processes nb_processors`.

After executing the command in Listing 5 , the "power data" for Germany is exported into the PostgreSQL database `de_power_151109`, created in Step 2. The `de_power_151109` is extended with *PostGIS* and hstore extensions.

3.5 Abstraction

After downloading, filtering and exporting the OSM "power data" for Germany into the database `de_power_151109`, the input data necessary for the **SciGRID** network model is available.

The **SciGRID** network model is based on the "power" relations, i.e. relations with a key/value pair route/power. Relations with the key "route" and value "power" are typically constituted of one or more substations and one or more transmission lines. The relations considered in this second version of the **SciGRID** model are:

- Relation with only two substations and one/several transmission lines linking them, an example of such relations is shown on Fig. 6. Substations are defined in OSM by the key "power" and the values "substation", "sub_station", "station", "plant", "generator". Transmission lines are on the other side defined by the key "power" and the values "line" and "cable".

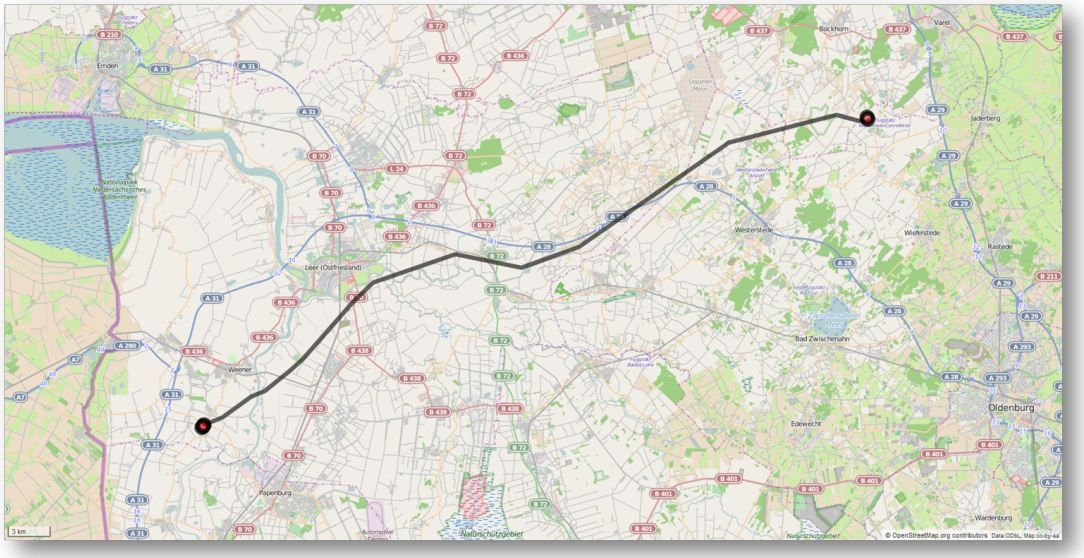


Figure 6: Example of a relation constituted of two substations. The two red circles are the two substations contained in the relation (OSM-ID=1637161) and the black line are the transmission lines (this relation contains four transmission lines). Figure obtained using overpass-turbo.eu, credits: ©OpenStreetMap contributors.

- Relations with three substations and with a T-junction. T-junctions are connections where transmission lines branch, an example of a relation with 3 substations and a T-junction is shown in Fig. 7. The nodes at which the lines branch are called T-nodes, an example is shown on Fig. 8.

These simplifications are used in this version of **SciGRID** as it is quite straightforward to extract the network model when only considering such relations. Relations which have more than three substations are not considered in a first approximation due to the difficulty of calculating the transmission lines length. This is due to the fact that the relation members (nodes and ways) may not belong to each possible connection between two substations. This makes it difficult to define which transmission lines are connected to which substation, so that the length of the transmission lines is not possible to calculate without the use of an elaborated algorithm or a routing routine. Relations with zero or one substation are also not considered in **SciGRID** as they constitute incomplete electrical circuits. In **SciGRID** only circuits with "at least" two substations and one or more transmission lines are considered.

The abstraction involves extracting the vertices (nodes) and links (edges) of the transmission network without considering the path followed by the transmission lines. The vertices of the network are represented by the geographical center positions of the substations which are members of the "power" relations. This procedure guarantees that possibly multiple relations with a shared substation will be abstracted with the same vertex, since the geometric polygon of the substation will be abstracted to its geometric center. The links (or edges) of the network are represented by the transmission lines as straight lines, without including the information about their paths. Therefore, the network produced by the **SciGRID** model is said to be an *abstracted* transmission network, as it does not reproduce the transmission lines actual paths. It is however straightforward to conserve the information about the topology of the transmission lines if needed.

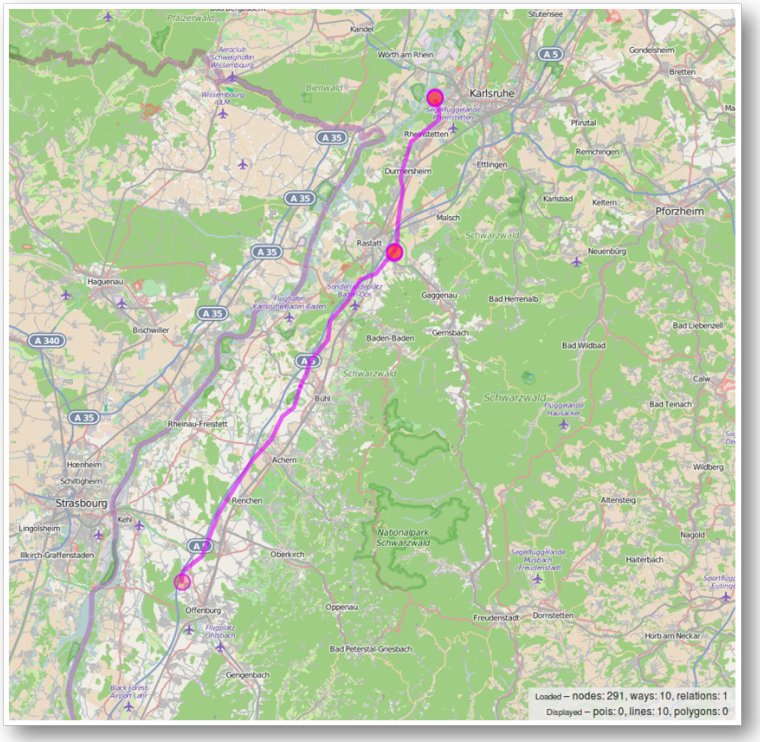


Figure 7: Example of a relation (OSM-ID=339244) containing three substations (circles), a T-junction and seven transmission lines. Figure obtained using overpass-turbo.eu, credits: ©OpenStreetMap contributors..

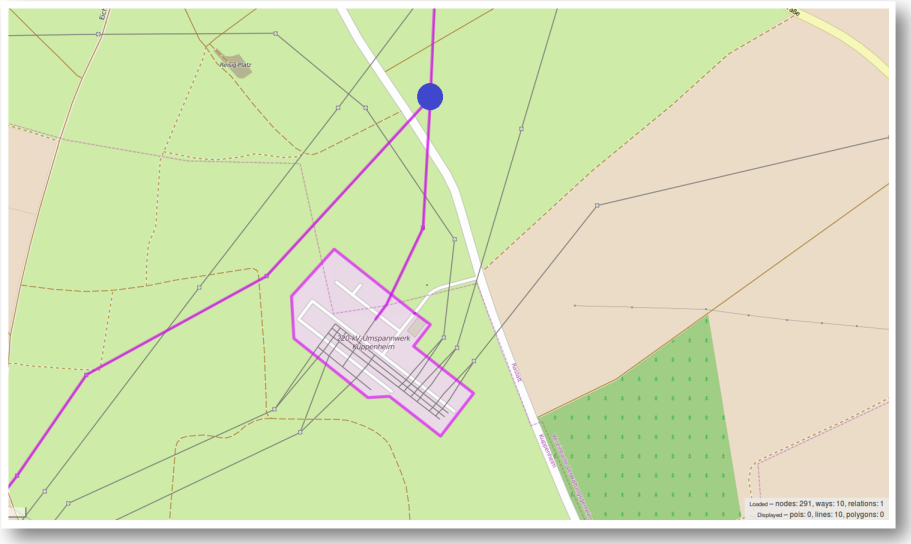


Figure 8: T-connection from relation (OSM-ID=339244). The T-node represented by the blue filled circle, is the node at which the transmission line at the top part of the figure branch into two parts. Figure obtained using overpass-turbo.eu, credits: ©OpenStreetMap contributors.

The abstraction step in **SciGRID** will produce the vertices and the links of the abstracted transmission grid, which can be stored as `.csvdata` files. The abstraction is divided in several sub-steps and is executed by running the python script `SciGRID.py`.

In the following is a listing of the different steps involved in the abstraction process: first for relations with two substations (Section 3.5.1) then for relations with 3 substations and a T-junction (Section 3.5.2).

3.5.1 Relations with 2 substations

The abstraction is performed using a python script called SciGRID.py located in the `SciGRID/code` folder. In the following, the different steps involved in the abstraction are introduced.

Abstraction step 1: "power" relations analysis

To be able to abstract relations with two substations, an analysis of the available "power" relations is performed on the data in the database `de_power_151109`. The analysis will concern the number of the substations (and also transmission lines) contained in each "power" relation. Note that, only relations where the voltage tag has a value of 220kV and higher are considered, as in **SciGRID** only the extra-high voltage transmission system is modelled.

The analysis of the relations is performed by the function `relation_analysis.py` called by the `SciGRID.py` script. Several *SQL* functions are defined in `relation_analysis.py` to list all relations in the database `de_power_151109` and to check for different attributes. The relations have a tag "parts", where all parts of a relation (including relations, ways, and nodes) are listed with their respective IDs. The relations "parts" or elements usually consists of transmission lines and substations. The relation "parts" can also have extra tags other than the "power" tag. If a part of a relation has the tags "power=construction", "power=planned" or "power=fixme" or has no power tag, it is listed as a "discarded" part and the relation is excluded from **SciGRID**.

The number of substations and transmission lines involved in each relation is listed and stored in the table `_analysis_rels`. The functions defined in `relation_analysis.py` use the relation ID as a variable (input) value and each function call analyses the relation in terms of:

- the relation electrical properties (voltage, cables, wires, frequency)
- the (relation,way,node) IDs of substation parts
- the (relation,way,node) IDs of transmission line parts
- the (relation,way,node) IDs of discarded parts
- number of substation parts
- number of transmission line parts
- number of discarded parts
- the total number of parts
- an analysis for T-junctions IDs

A summary of the relations analysis for Germany is shown in Table 1 and 2.

relations to be fixed / being planned / being under construction	52
relations with 0 substation	4
relations with 1 substation	23
relations with 2 substations	643
relations with 3 substations	67
relations with 4 substations	3
total number of "power" relations in Germany	792

Table 1: Number of relations with key/value pair route/power in the dataset of Germany in OSM (Status: 18.05.2015). The total number of relations is subdivided into sets of relations with different numbers of substations or categorized as discarded.

relations to be fixed / being planned / being under construction	54
relations with 0 substation	2
relations with 1 substation	11
relations with 2 substations	690
relations with 3 substations	65
relations with 4 substations	4
total number of "power" relations in Germany	826

Table 2: Number of relations with key/value pair route/power in the dataset of Germany in OSM (Status: 09.11.2015). The total number of relations is subdivided into sets of relations with different numbers of substations or categorized as discarded.

Abstraction step 2: creating the vertices table

As stated earlier in this section, relations containing only 2 substations can be trivially connected. Using these relations, the polygons of the substations are abstracted to their geometric center. The centres of all substations will then build the vertices table of the **SciGRID** network model. As an example, see Fig.9, where the substations of relation OSM-ID:3756858 are abstracted to their geometric centres (in orange). The abstracted centres represent the transmission model vertices.

After analysing the relations as explained previously, three tables are needed to obtain the list of vertices derived from relations with exactly 2 substations. They are created using the three *SQL* commands in Listings 6-8, which are part of the script `create_db_tables.py`.

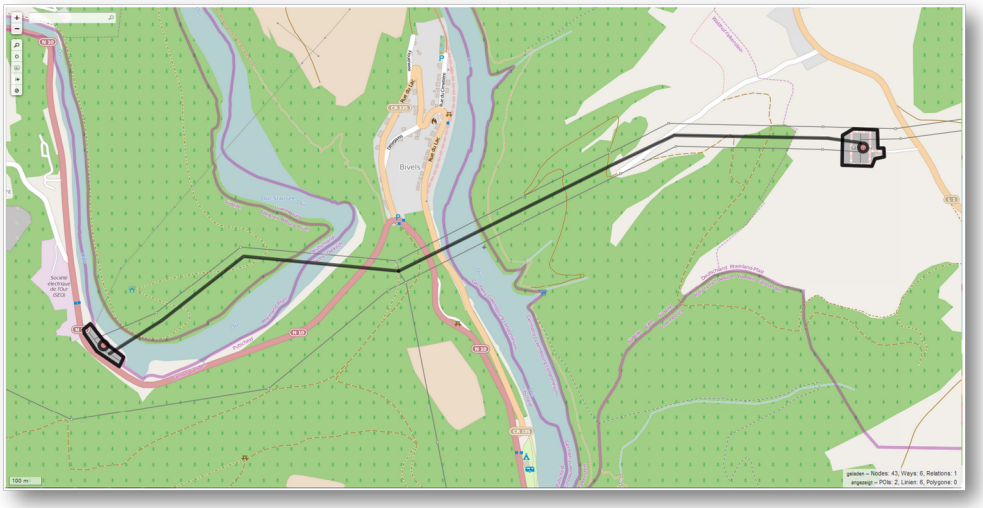


Figure 9: Abstraction of the substations in the relation OSM-ID:3756858. The centres of the substations in orange represent the network vertices, and the lines in black are the original transmission lines. Figure obtained using overpass-turbo.eu, credits: ©OpenStreetMap contributors.

Listing 6: Intermediate vertices table

```
CREATE TABLE _vertices (
    id              serial PRIMARY KEY NOT NULL,
    osm_id          bigint,
    osm_id_typ      char,
    geo_center      geometry,
    longitude       float,
    latitude        float,
    role            text,
    voltage         text,
    from_relation   bigint);
```

The script `create_db_tables.py`, called by the `SciGRID.py` script creates all tables needed for the abstraction process. The table `_vertices` in Listing 6 is an intermediate list of vertices, which also contains the relation's ID, given by the table `_analysis_rels` from Step 3.5.1. Since different relations can share the same substation (vertex), the table `_vertices` can contain repeated vertices. Therefore an intermediate step is necessary to "clean" these repeated vertices so that the table `vertices` contains non-repeated "unique" vertices. A list of unique vertices is obtained by adding vertices only once to the table `vertices` (see Listing 8). These vertices get a new ID `v_id`, which identifies them within the **SciGRID** model. This new `v_id` is necessary, since vertices can be derived from different data types in OSM which may share the same `osm_id`. The OSM IDs are only unique within a data type, namely nodes, ways, or relations. This were the table `vertices_ref_id` (see Listing 7) comes into play. This table links the original OSM ID `osm_id` and the unique **SciGRID** `v_id`. This table plays also an important role in the update process of the **SciGRID** dataset, see Section 3.8.

Listing 7: Intermediate table of abstracted substations IDs

```
CREATE TABLE vertices_ref_id (
    v_id           serial PRIMARY KEY NOT NULL,
    osm_id         bigint,
    osm_id_typ     char,
    visible        smallint);
```

Listing 8: Table of abstracted substations

```
CREATE TABLE vertices (
    v_id           bigint PRIMARY KEY,
    lon            float,
    lat            float,
    typ            text,
    voltage        text,
    frequency      text,
    name           text,
    operator        text,
    ref            text,
    WKT_SRID_4326  text);
```

In table `vertices`, additional data about the voltage level, frequency, name, operator, and reference (ref) of the substation are also included.

Abstraction step 3: creating the links table

Similarly to obtaining a table of vertices, two tables are needed to obtain the links table. These tables are created using the following two *SQL* commands listed in Listings 9-10. These two commands are part of the script `create_db_tables.py`.

The relation IDs are given by the table `_analysis_rels` from Step 3.5.1. For each relation, a function which abstracts the connections between two substations to a link is applied. The link is defined as a straight line connecting two abstracted substations (see Fig.10). This step is accomplished by the function `abstract_rel_with_2subs` included in the python script `relation_abstraction.py`.

The properties of each link, including its true length, the number of wires, cables, frequency, name, ref, and operator are added to the table `links`. The table `_links` is an intermediate table, since the begin and end of a link need to have a unique ID as introduced with the ID `v_id`. Therefore, an intermediate step is necessary to transform the original beginning (ending) `osm_id` of a link to the corresponding `v_id` at the start (end) of the link. These links are assigned a new ID `l_id` identifying links in the **SciGRID** dataset. The table `links` uses the referenced `v_id` obtained from table `vertices_ref_id` to identify the vertices IDs.

Listing 9: Intermediate links table

```
CREATE TABLE _links (
    id                  serial PRIMARY KEY NOT NULL,
    osm_id_1            bigint,
    osm_id_1_typ        char,
    osm_id_2            bigint,
    osm_id_2_typ        char,
    length_m            integer,
    way                 geometry,
    voltage             integer,
    cables              integer,
    wires               text,
    wires_nb            integer,
    frequency           text,
    from_relation       bigint,
    from_transmissions  bigint[]);
```

Listing 10: Table of abstracted connections between substations

```
CREATE TABLE links (
    l_id                serial PRIMARY KEY NOT NULL,
    v_id_1              bigint,
    v_id_2              bigint,
    voltage             integer,
    cables              integer,
    wires               integer,
    frequency           text,
    name                text,
    operator            text,
    ref                 text,
    length_m            integer,
    r_ohmKm             float,
    x_ohmKm             float,
    c_nFKm              float,
    i_th_max_A          float,
    from_relation       bigint,
    WKT_SRID_4326       text);
```

3.5.2 Relations with 3 substations and a T-junction

As mentioned earlier, T-junctions are located where transmission lines branch (see Fig. 7 and Fig. 8). In a first approximation, only relations with 3 substations and a T-junction are going to be abstracted in this version of the **SciGRID** model. As for the relations with exactly 2 substations previously introduced, the abstraction of relations with three substations and a T-junction is accomplished by using the `SciGRID.py` script. For simplicity, these relations are labelled "T-junction relations" in the remainder of this user guide.

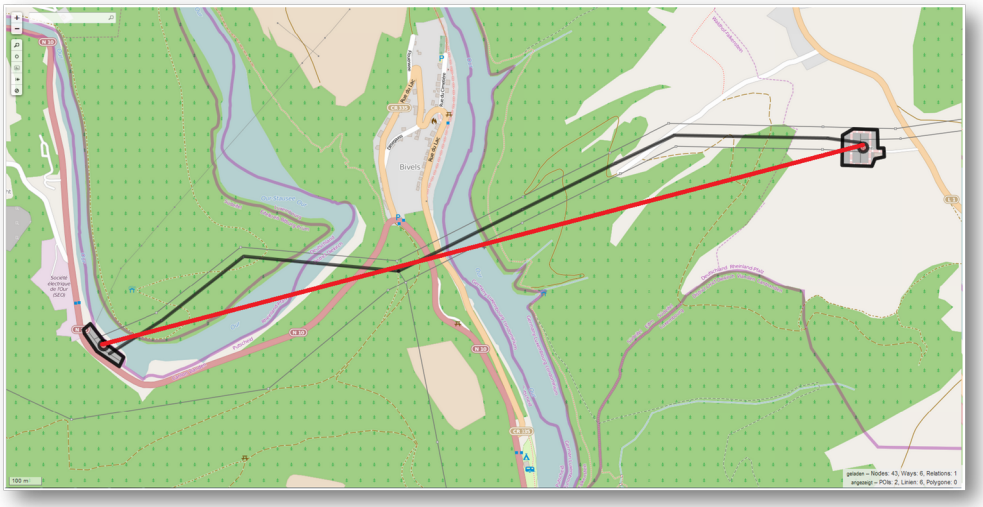


Figure 10: Abstraction of the relation OSM-ID:3756858: the path followed by the transmission lines (in black) is not considered in the ***SciGRID*** *abstracted model and the network link is represented by a straight line instead (in red). Figure edited using overpass-turbo.eu, credits: ©OpenStreetMap contributors.*

The abstraction of T-junction relations follows the same sub-steps introduced in Section 3.5.1, which are: extracting the desired relations from the relation analysis results, abstracting the vertices and links and exporting the results to the *vertices* and *links* tables.

After running the analysis script `relation_analysis.py`, the relations with 3 substations are sorted out and their IDs identified. This is achieved in the python script `relation_abstraction`, (called by the `SciGRID.py` main script) by the function `abstract_rel_with_T_node`.

Not all relations with 3 substations contain a T-junction. In order to find T-junctions, relations with three substations are analysed. First, a list of all nodes is selected from all relation parts associated with transmission lines. In this nodes list, the nodes are counted. Since nodes within one associated part of transmission line occur only once, the nodes which appear 3 times in the list of nodes over all associated part of transmission lines belong to three different transmission line segments. Thus, these three segments share the same node which is then identified as T-node.

Since, relations with 3 substations and more than 3 cables can have more than one T-node, one must also check if a connection from each of the three segments start at the T-node and end in a different substation. This is achieved by the function `abstract_3subs_T_node` in `relation_abstraction`. The relations for which T-nodes exist are then abstracted by using the T-node as an "auxiliary vertex" linking the three substations present in the relation (see Fig. 11 and Fig. 12). The relations with 3 substations which have no T-junction are not included in the present **SciGRID** version.

To define which transmission lines lies between the 4 abstracted vertices (3 as the substation nodes and the T-junction node), the function `separate_parts` is used by the abstraction script `relation_abstraction`. Then, the function `insert_segments` is used to insert and update the tables `vertices` and `links` for the vertices and links obtained.

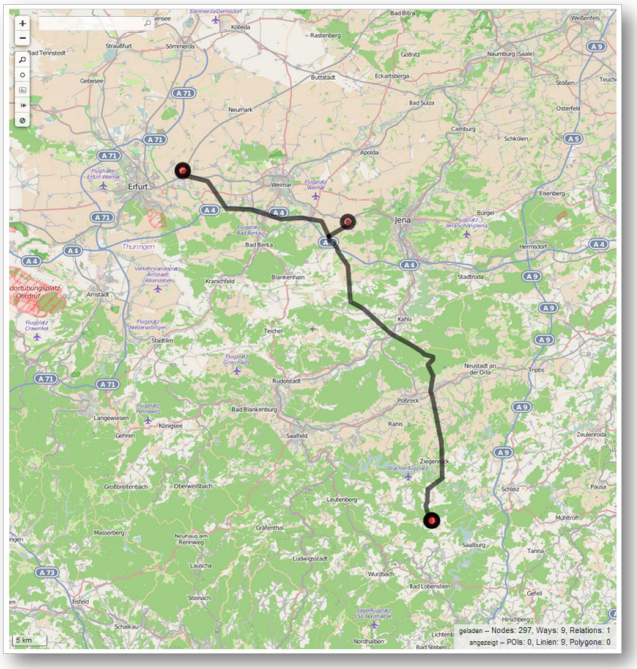


Figure 11: Example of power relation (OSM-ID:918569) with 3 substations and a T-junction. The transmission lines are in black and the stations geometrical centres in orange. Figure obtained using overpass-turbo.eu, credits: ©OpenStreetMap contributors.

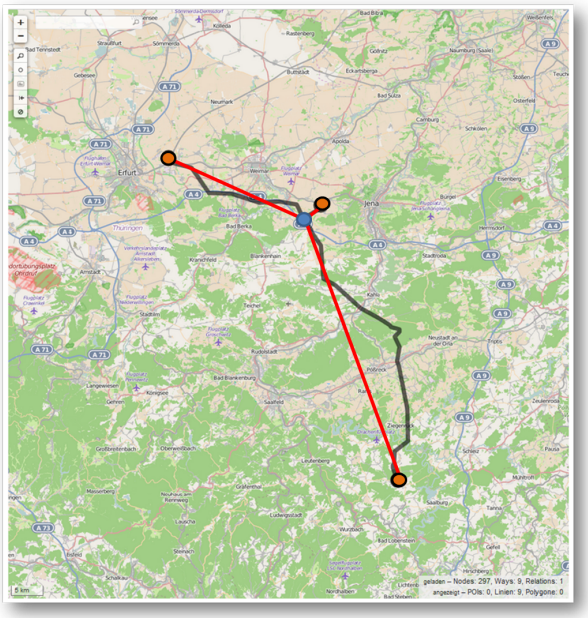


Figure 12: Example of the abstraction of a T-junction relation (OSM-ID:918569) with 3 substations and a T-junction. The transmission lines in black represent the real path and the red lines the abstracted lines. They rely the T-node in blue, which is used as an auxiliary vertex, and the 3 substations vertices in orange. Figure edited using overpass-turbo.eu, credits: ©OpenStreetMap contributors.

Vertices and links tables: The SciGRID model output

The result of executing the abstraction process described above are two new tables added to the `de_power_151109` database. First, the table `vertices`, defined in Listing 8, contains the vertices of the transmission network. Second, the table `links`, defined in Listing 10, which contains the links of the transmission network.

Additionally, in the `links` table the electrical properties of the transmission lines are calculated by the `electrical_properties.py` which is called by `SciGRID.py`. The electrical properties calculated are: resistance r, reactance x, capacitance c, and maximum current thermal limit $I_{\max}$. They are inserted in the `links` table as columns. The values of r, x, c and $I_{\max}$ depend on: the voltage level, the number of wires in a transmission line, and the number of cables in a circuit. In **SciGRID**, the electrical properties are calculated in per km units as follows:

$$r_{ohmKm} = C_r/(\frac{wires}{wires_{typical}})/(\frac{cables}{3}) \tag{1}$$

$$x_{ohmKm} = C_x/(\frac{wires}{wires_{typical}})/(\frac{cables}{3}) \tag{2}$$

$$c_{nFKm} = C_c \cdot (\frac{wires}{wires_{typical}}) \cdot (\frac{cables}{3}) \tag{3}$$

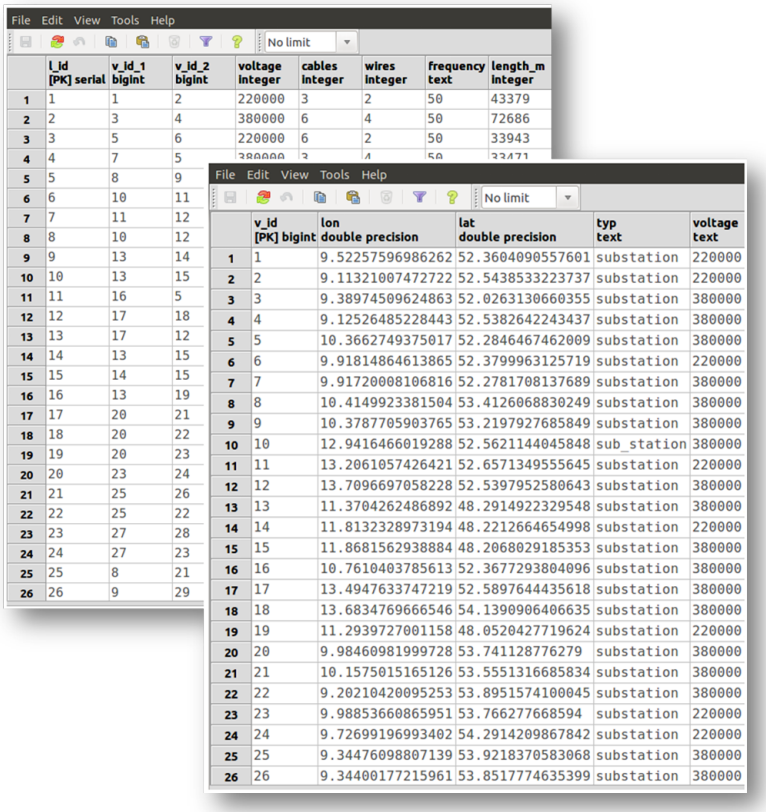
$$I_{th\_max\_A} = C_I \cdot (\frac{wires}{wires_{typical}}) \cdot (\frac{cables}{3}) \tag{4}$$

where, $wires_{typical}$ is the number of wires in a transmission cable. $wires_{typical}$ as is typically 2 for transmission lines of $220kV$ and 4 for transmission lines of $380kV$. The coefficients used in Eq. 1-4 are listed in table 3 from reference [17].

Voltage level	C_r (ohm/km)	C_x (ohm/km)	C_c (nF/km)	C_I (A)
380 kV	0.025	0.25	0.0137	2.6
220 kV	0.080	0.32	0.0115	1.3

Table 3: Electrical properties coefficients from reference [17].

The `vertices` and `links` tables are part of a "relational database", i.e. their rows have a unique key. Since each row in a relational database has its own unique key, rows in other tables that are related to it can be linked to it by storing the original row's unique key as an attribute of the secondary row. Fig. 13 displays an example of the tables `vertices` and `links` obtained when executing the abstraction script `SciGRID.py`.



	l_id [PK] serial	v_id_1 bigint	v_id_2 bigint	voltage integer	cables integer	wires integer	frequency text	length_m integer
1	1	1	2	220000	3	2	50	43379
2	2	3	4	380000	6	4	50	72686
3	3	5	6	220000	6	2	50	33943
4	4	7	5	380000	3	4	50	33471
5	5	8	9					
6	6	10	11					
7	7	11	12					
8	8	10	12					
9	9	13	14					
10	10	13	15					
11	11	16	5					
12	12	17	18					
13	13	17	12					
14	14	13	15					
15	15	14	15					
16	16	13	19					
17	17	20	21					
18	18	20	22					
19	19	20	23					
20	20	23	24					
21	21	25	26					
22	22	25	22					
23	23	27	28					
24	24	27	23					
25	25	8	21					
26	26	9	29					

	v_id [PK] bigint	lon double precision	lat double precision	typ text	voltage text
1	1	9.52257596986262	52.3604090557601	substation	220000
2	2	9.11321007472722	52.5438533223737	substation	220000
3	3	9.38974509624863	52.0263130660355	substation	380000
4	4	9.12526485228443	52.5382642243437	substation	380000
5	5	10.3662749375017	52.2846467462009	substation	380000
6	6	9.91814864613865	52.3799963125719	substation	220000
7	7	9.91720008106816	52.2781708137689	substation	380000
8	8	10.4149923381504	53.4126068830249	substation	380000
9	9	10.3787705903765	53.2197927685849	substation	380000
10	10	12.9416466019288	52.5621144045848	sub_station	380000
11	11	13.2061057426421	52.6571349555645	substation	220000
12	12	13.7096697058228	52.5397952580643	substation	380000
13	13	11.3704262486892	48.2914922329548	substation	380000
14	14	11.8132328973194	48.2212664654998	substation	220000
15	15	11.8681562938884	48.2068029185353	substation	380000
16	16	10.7610403785613	52.3677293804096	substation	380000
17	17	13.4947633747219	52.5897644435618	substation	380000
18	18	13.6834769666546	54.1390906406635	substation	380000
19	19	11.2939727001158	48.0520427719624	substation	220000
20	20	9.98460981999728	53.741128776279	substation	380000
21	21	10.1575015165126	53.5551316685834	substation	380000
22	22	9.20210420095253	53.8951574100045	substation	380000
23	23	9.98853660865951	53.766277668594	substation	220000
24	24	9.72699196993402	54.2914209867842	substation	220000
25	25	9.34476098807139	53.9218370583068	substation	380000
26	26	9.34400177215961	53.8517774635399	substation	380000

*Figure 13: Example of the **SciGRID** model network vertices and links tables obtained using the abstraction command.*

The data obtained when running the abstraction script constitutes the output of the **SciGRID** model. The `vertices` and `links` tables (only with relations containing 2 substation and T-junction relations, for the moment) can be used and/or edited as a table or a `.csvdata` file. The `.csvdata` files of the vertices and links are both provided in the `SciGRID/data/03_network` folder.

The user can adapt the previous steps, of download, filtering and abstraction for any region. The same apply for the open source tools used. The next step in **SciGRID** model development will be to also include the rest of the relations, which contain more than three substations.

3.6 Visualization

There are different ways to visualize the **SciGRID** abstracted network model. When running the `SciGRID.py` abstraction script, the last step of the abstraction is to create a plot of the abstracted network. This is done by the function `create_plots.py`. The plot is stored as a `.png` file in the folder `SciGRID/data/04_visualization`. An example is shown in Fig. 14.

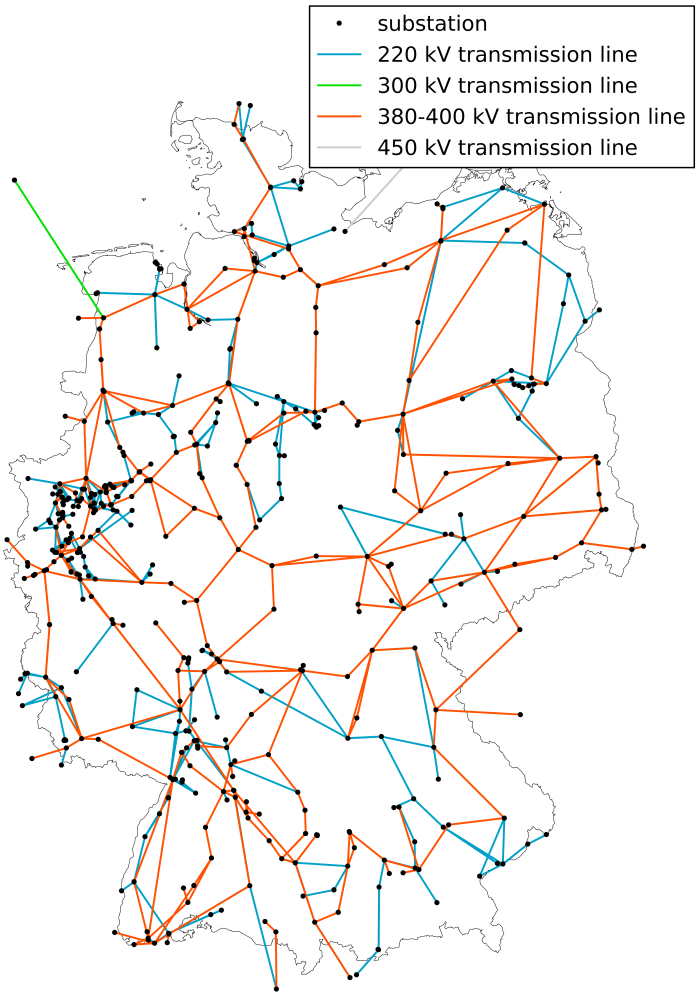


*Figure 14: Example of the plotted **SciGRID** abstracted model network using the plot function available in `SciGRID.py`, status 09.11.2015. Credits: ©OpenStreetMap contributors.*

Another option to visualize the resulting **SciGRID** abstracted model is to use the QGIS application [18], which is a free and open-source desktop Geographic Information System (GIS) providing data viewing, editing, and analysis capabilities for GIS enabled tables and databases under different formats. For more information about QGIS and how to install it, refer to [18].

Visualization using QGIS

In the following, the different steps necessary to open and visualize the **SciGRID** csvdata files are presented.

After installing QGIS, load/open it by typing the command `./qgis` in a shell terminal. Click on "Add Delimited Text Layer" on the menu on the left of the QGIS GUI (see Fig. 15.

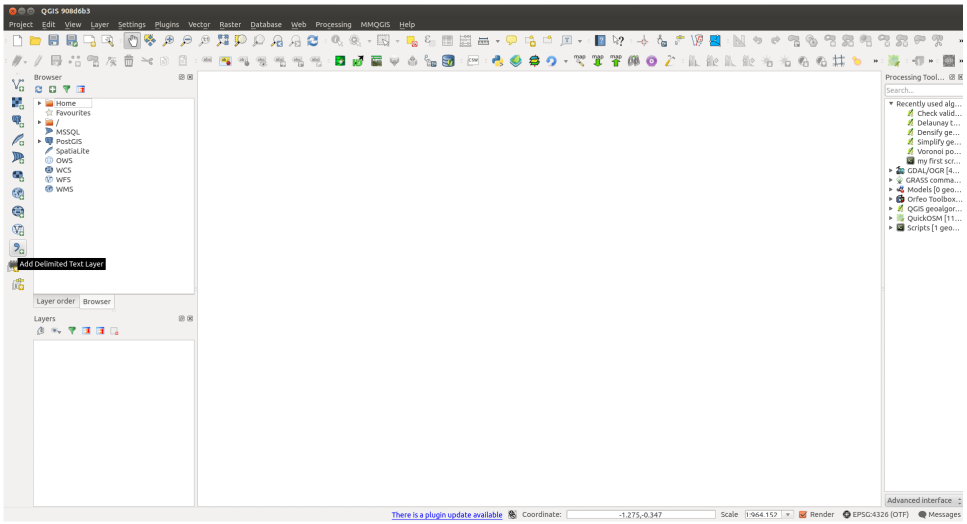


Figure 15: "Add Delimited Text Layer" option in QGIS.

A menu appears (see Fig.16, click on the Browse button at the top right corner of the menu and load the .csvdata file (see Fig. 17.

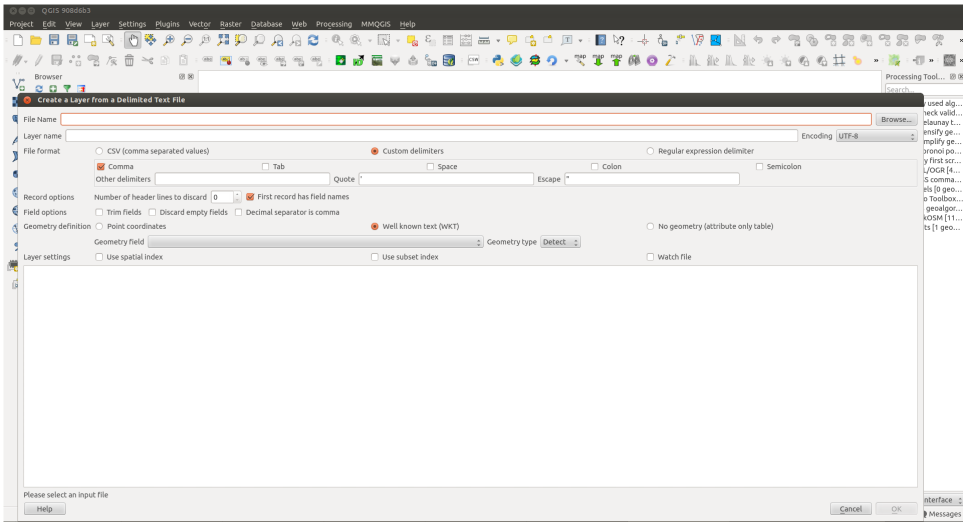


Figure 16: Add Delimited Text Layer menu in QGIS.

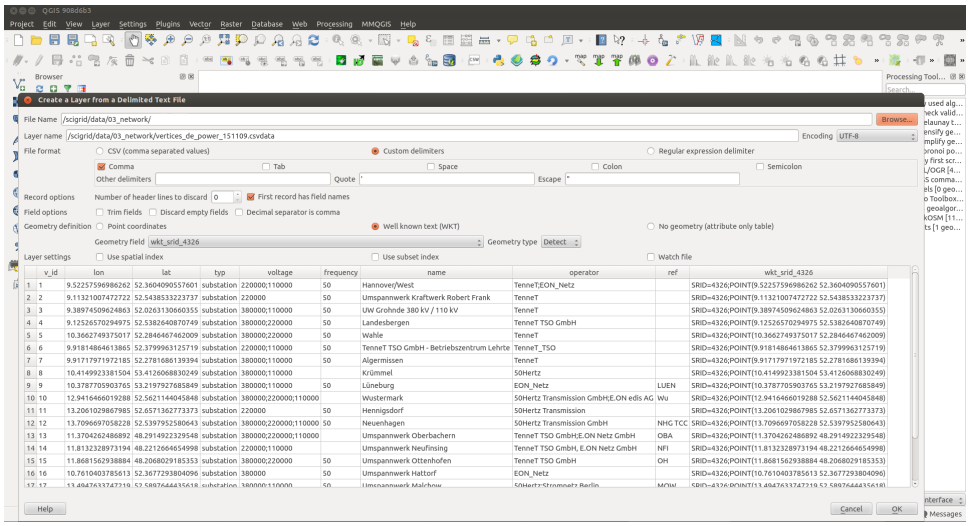


Figure 17: Browse the vertices csvdata file using the menu in QGIS.

Click on "OK" when the file is loaded and the data is displayed in QGIS, an example is shown for the file `vertice_de_power_151109.csvdata` on Fig. 18.

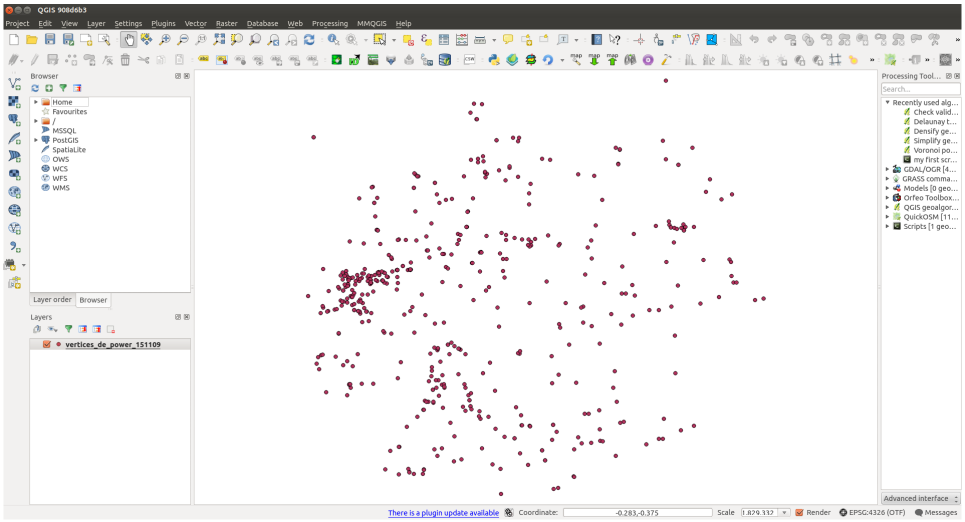


*Figure 18: Visualization of the **SciGRID** vertices in QGIS.*

Further analysis of the data can be conducted using QGIS, for example *nearest neighbour analysis*. For more information about the available data analysis tools in QGIS refer to [18].

3.7 Running SciGRID with the makefile

The different steps necessary to build the **SciGRID** model, introduced in the previous sections, can be run using the makefile. The makefile extracts the necessary information from the file `default_config.mk`. In the following, the makefile and `default_config.mk` file are introduced.

The makefile is included in the `SciGRID/code` folder. The advantage of using the makefile is to automate processing steps and to enable parameter extraction from a configuration file by

using environment variables. Moreover, dependency/targets tracking of files is possible, which enables tasks execution control. [2]

Before running the makefile make sure that you already installed *PostgreSQL* and *osmosis*. The user needs also to make sure that all necessary parameters have been correctly provided in the configuration file *default_config.mk*. These parameters are:

1. The URL of the planet OSM data
2. The location of **SciGRID** data folder paths (data, network and visualization).
3. The location of the temporary storage folder for *osmosis*.
4. The cache value and the number of processors to be used by *osm2pgsql*.
5. The paths to the executable binaries of *osmosis* and *oms2pgsql*.
6. The connection parameters to the *PostgreSQL* database.

To run the makefile , type "make config=default_config.mk task" in a bash terminal while in the `SciGRID/code` folder. The user can indicate any `.mk` file for the configuration option and *task* takes a different value depending on which task to execute. The following tasks are available:

1. `make test`: will test the **SciGRID** model using the `.pbf` data for the state of Bremen (Germany). Using the test option, the `.pbf` file of Bremen is downloaded from Geofabrik [19], filtered and abstracted while assuming default PostgreSQL connection parameters (server-cluster=9.1/main, host=127.0.0.1, user=postgres, port=5432, and no password).
2. `make config=default_config.mk download`: the download option will download the OSM data from the provided URL.
3. `make config=default_config.mk drop_database`: will drop an existing power database (requires a user confirmation before dropping the database).
4. `make config=default_config.mk filter_OSM`: is used to filter the OSM data for the power tag using *osmosis*, as introduced in Section 3.3.
5. `make config=default_config.mk scigrid`: executes database creation, export power data to the database, and data abstraction as introduced in Section 3.5.
6. `make config=default_config.mk clean`: deletes milestones (`.done`) located in the folder log and .pyc (cached bytecode files produced by python) files.
7. `make config=default_config.mk clean_all`: deletes all files as the *clean* option and additionally deletes the `.log` files, `.csvdata`, network plot (`.png`), as well as raw and power data files (`.pbf` files).
8. `make config=default_config.mk all`: will execute all steps starting with downloading planet OSM data, filtering for power data, abstraction of data and storing network and visualization files.

[2] In **SciGRID** release V0.1 there was a possibility to use either a bash script or a makefile. To avoid redundancy, the bash script is no longer supported and hence not provided in release V0.2.

3.8 Update of the SciGRID dataset

The OSM database is continuously being updated and everyday new nodes, ways and relations are added and/or deleted. An updated version of the planet OSM file is released every week and made available on different mirror websites [20]. To keep the **SciGRID** vertices and links databases up-to-date an *update* procedure is implemented. The table `vertices_ref_id` (see Section 3.5, Listing 7) is the main component to update **SciGRID**'s links and vertices tables when a new `.pbf` file is downloaded and the abstraction is executed. This procedure is introduced in the following.

Consider the following scenario: A user downloads a `.pbf` file for the 1.1.2015 (using `make download`), filters the data (using `make filter_OSM`) and then executes the **SciGRID** abstraction (using `make scigrid`). When the abstraction is executed, the table `vertices_ref_id` is created. It contains the ID of the vertices, their type and their OSM ID. Additionally, a column `visible` is present and has the value 1 for all entries. This entry means that the vertices are *abstracted*. A copy of the `vertices_ref_id` table is stored in the folder `SciGRID/data/03_-netwok`, with the database name it is derived from.

Two months later, the user obtained/downloaded a newer `.pbf` file and wants to abstract the new dataset using **SciGRID**. This is done by running the filtering and the **SciGRID** abstraction again on the new data file. However, during the abstraction, the makefile detects a copy of the `vertices_ref_id` table. The makefile then sets all `visible` entries in this table to 0. When the abstraction is executed (new `vertices` and `links` tables are created) the IDs of the vertices in the new `vertices` table are checked by reading the table `vertices_ref_id` and comparing the entries. If a vertex already existed in the previous OSM database the `visible` column gets a value 1. If it is a new vertex in the OSM database, the vertex is added at the end of the `vertices_ref_id` table and the `visible` column gets a value 1. If the vertex has been deleted from the OSM database the vertex is still stored in the `vertices_ref_id` table but the `visible` column gets a value of 0. If at any point later this deleted vertex is reused again, it will have the same ID in **SciGRID** and the `visible` column gets a value of 1

In this fashion, the vertices and links datasets are updated and their IDs remain unique in their respective tables.

4 SciGRID GUI

The **SciGRID** model is provided with a Graphical User Interface (GUI) written in python's standard GUI package: TkInter [21]. The source code of the GUI, *pyGUI.py*, is included in the folder `SciGRID/code`. Using a GUI allows for an easy and simple interaction with the **SciRGID** model and enables the user to input and adjust the different parameters of the model and to execute all steps included in the makefile. A snapshot of the GUI is shown on Fig. 19.

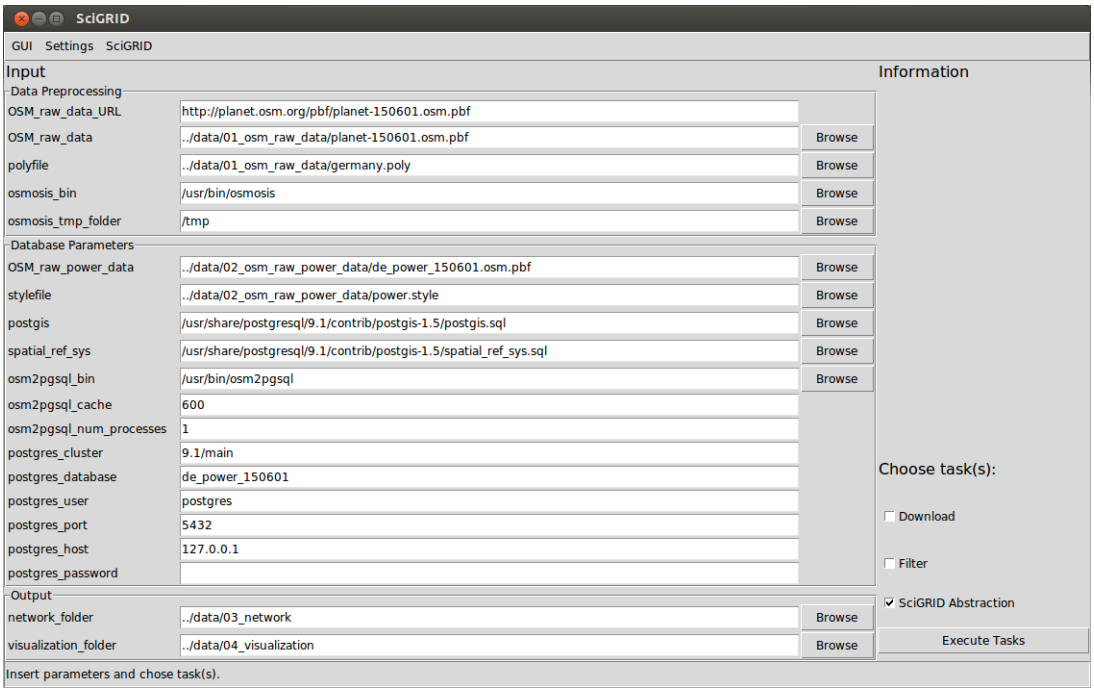


*Figure 19: A snapshot of the **SciGRID** GUI.*

The GUI is based on the structure of the **SciRGID** model [9]. [3] The GUI is divided into four sections: the *Input/Output* section, the *Information* section, the *Menu and Status Bar* and the *Tasks* section.

In the *Input/Output* section the configuration parameters are set, including data folders locations, OSM data URL, database name, etc. The *Information* widget and the *Menu and Status Bar* provide helpful information about the different fields available for settings and the progress of the tasks in execution. In the *Tasks* section, the user can chose from the tasks download, filter and **SciGRID** abstraction. In order to execute the tasks a click on the Button "Execute Tasks" is followed.

4.1 Input/Output section

The *Input/Output* fields are situated on the left side of the GUI and they allow for the input of the parameters necessary to run the **SciGRID** model by filling different entry fields (see Fig. 20).

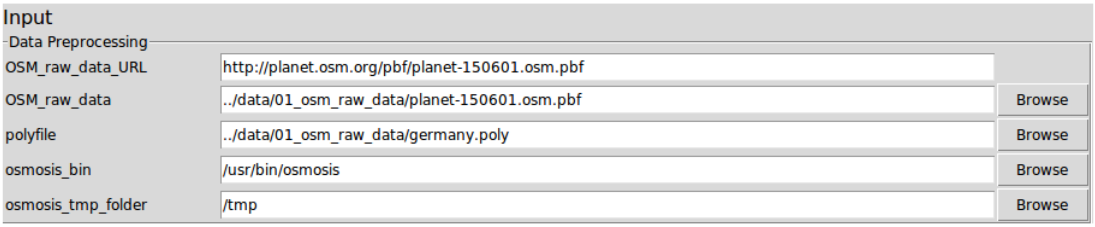


*Figure 20: Input section of the **SciGRID** GUI. Only the fields used for OSM Data Preprocessing are shown.*

[3] The requirements needed to run the **SciRGID** model (see Section 3.1) are necessary to use the GUI.

Under the subsection *Data Preprocessing*, the user provides the OSM raw data URL in the field `OSM_raw_data_url`. The URL should include the website and name of the `.pbf` file to be downloaded. Usually, `.pbf` files containing planet OSM data are named as follows: `planet-date.osm.pbf`, where *date* indicates the *timestamp* of the planet OSM data (for example `planet-150907.osm.pbf` has the timestamp 07 September 2015). The most recent `.pbf` file is named *planet-latest.osm.pbf* instead of a *date*. If the OSM raw data file has already been downloaded, the `OSM_raw_data_url` input field is obsolete. Then, the `OSM_raw_data` field should indicate the location of the existing raw data file.

File locations can be set by typing them in the corresponding fields or by browsing the folders using the *Browse* button at the left of corresponding entry fields (see Fig. 21).

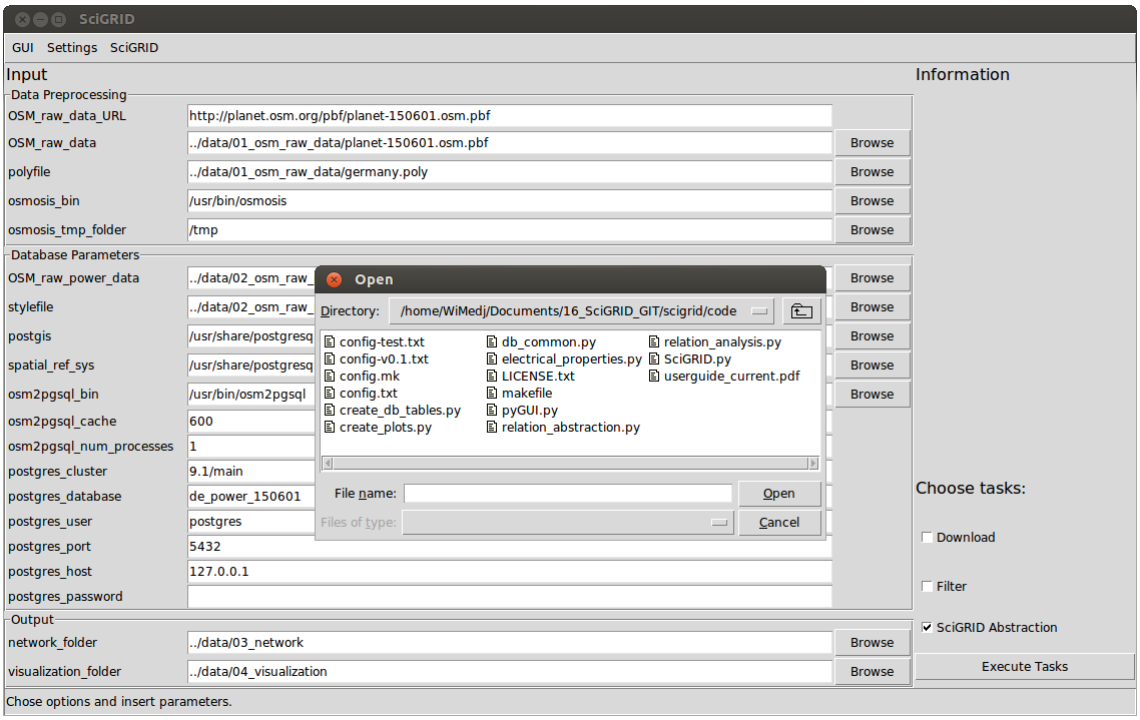


Figure 21: File browsing in the SciGRID GUI to indicate a file location. The browser is called by clicking on the "Browse" button at the left of each entry field.

The `polyfile` field is used to set the location and name of the *.polyfile* used. To set the locations of the `osmosis` binary and the temporary folder used by *osmosis* during the filtering process, the entries `osmosis_bin` and `osmosis_tmp_folder` are used.

In the *Database Parameters* subsection, the user indicates the different parameters necessary to create the power database and to establish the data export of filtered data (see Fig. 22). This includes among other parameters, the user name, the port number and the host address of the *PostgreSQL* server.

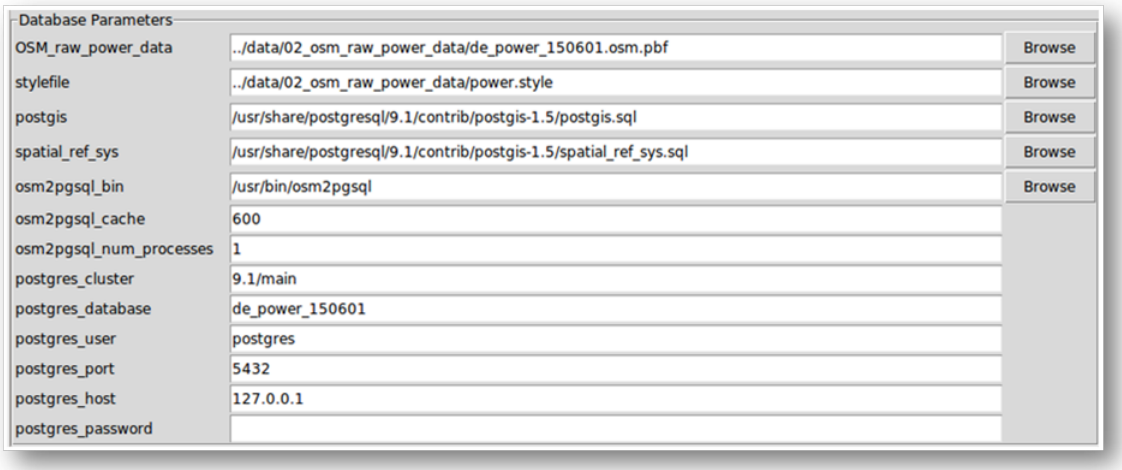


Figure 22: GUI section for setting the database and PostgreSQL parameters.

The filtered power data extracted by *osmosis* is stored at the location indicated in the field `OSM_power` and the *stylefile* location is indicated in the field `stylefile`. To enable the *PostGIS* and *spatial reference system*, the user need to complete the `postgis` and `spatial_ref_sys` fields. The location of the *osm2pgsql* binary executable is set by `osm2pgsql_bin` field. To make the data export faster, the user can specify the size of cache used in MB by varying `osm2pgsql_cache` value. Running the export in parallel is possible and the number of processors to be used can be indicated by setting the `osm2pgsql_num_processors` value.

PostgreSQL parameters are also entered using the GUI, where the database name to be created is indicated in the `Postgres_database` field, the user name in `Postgres_user` field, the default port number in `postgres_port` field . The *PostgreSQL* server host is set in the entry `postgres_host`, server name in entry `postgres_cluster`, while the password can be typed in the field `postgres_password` as is replaced by *-symbols.

In the *Output* subsection, the folders to store the links and vertices `.csvdata` files are indicated in `network`. The entry `visualisation` holds the location of the folder to store the network plot (see Fig. 23).

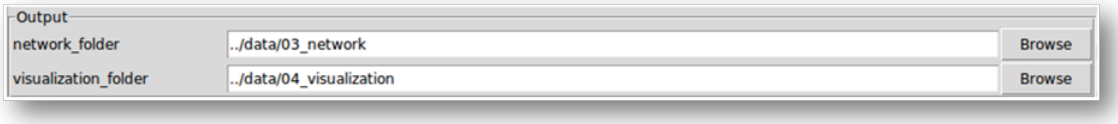


*Figure 23: Output section in the **SciGRID** GUI where the network `.csvdata` files and plot are to be stored.*

In the GUI, all entry fields are linked to a dictionary called *self.input_fields*, which manages the **SciGRID** GUI parameters. Changes in the entry-fields are traced by a text variable, which updates the *self.input_fields* dictionary.

4.2 Menu and status bar section

The menu bar of the GUI provides three options (see Fig. 24): the option *GUI*, *Settings* and *SciGRID*, see Fig. 24.

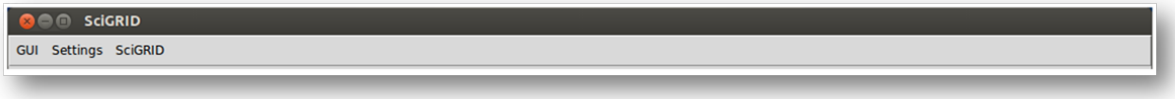


Figure 24: SciGRID GUI menu bar including three sections: GUI, Settings, SciGRID.

Under the option *GUI* the user can *Exit* the GUI.

The option *Settings* can be used to *open* or *save* the configuration file used. The option *Reset to SciGRID default* returns the default values for the input parameters and displays them visually in the respective entry fields.

The third dialogue in the menu bar, *SciGRID*, provides a link to the **SciGRID** website (*About SciGRID*), a link to the contact information, opens the **SciGRID** documentation as pdf (*Documentation*), and *licence* opens the licence file.

4.3 Information section

The information widget, located on the right side of the GUI (see Fig. 25, displays information when the mouse cursor hovers over an entry field. Helpful messages are displayed to provide information to the user about how to handle the specific entry.

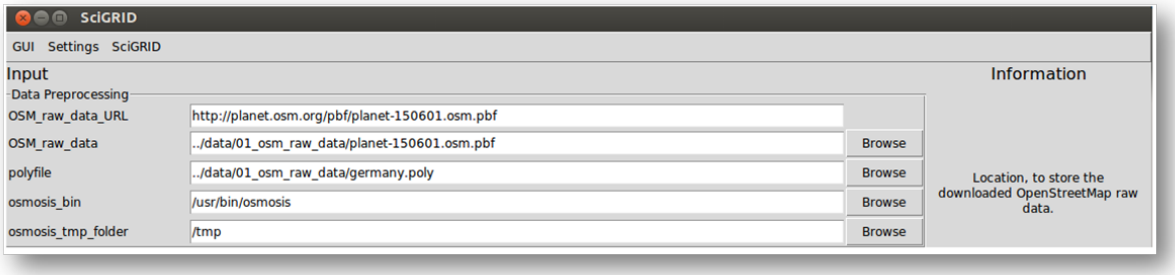


Figure 25: SciGRID GUI information section on the left side of the figure, displayed when the mouse is over the entry field osm_raw_data.

A text message appears in the status bar informing the user about the task being executed and its status (done or aborted). An example is shown in Fig. 26.

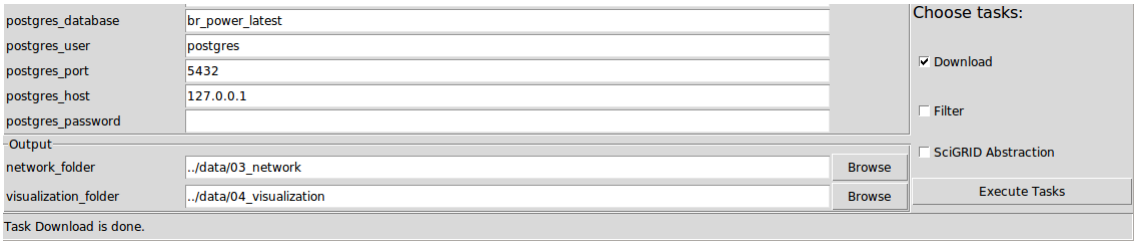


Figure 26: SciGRID GUI status bar message (lower left corner) indicating the successful execution of the download task.

4.4 Tasks section

In the lower right corner of the GUI three check boxes are provided: *Download*, *Filter* and *SciGRID Abstraction*. When checking one of the boxes and pressing the button *Execute Tasks* the respective command is executed, see Fig. 27.

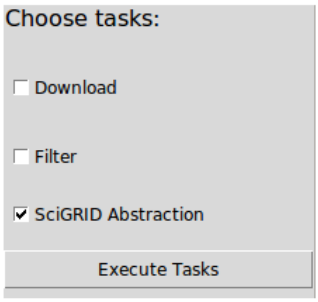


Figure 27: SciGRID GUI execute section with the different tasks available.

When checking the *Download* box, the `.pbf` file indicated in the *Data Preprocessing* section is downloaded and stored in the location indicated in `OSM_raw_data` field. A log file, *download.log*, containing the output message of the download process is created in the `SciGRID/code/log` folder.

If the file already exists, a message box appears (see Fig. 28) and the user is asked if the existing `.pbf` file should be replaced. When the answer is *yes*, the download process starts and the `.pbf` file is replaced. If the answer is *no*, the download process is aborted and the entry field `OSM_raw_data` is highlighted in red (see Fig. 29) to indicate the already existing file name and its location. This checking process avoids unnecessary data download and over-writing data files when data already exits. This can save time especially when dealing with OSM planet files which are large files (more than 30 Gb).

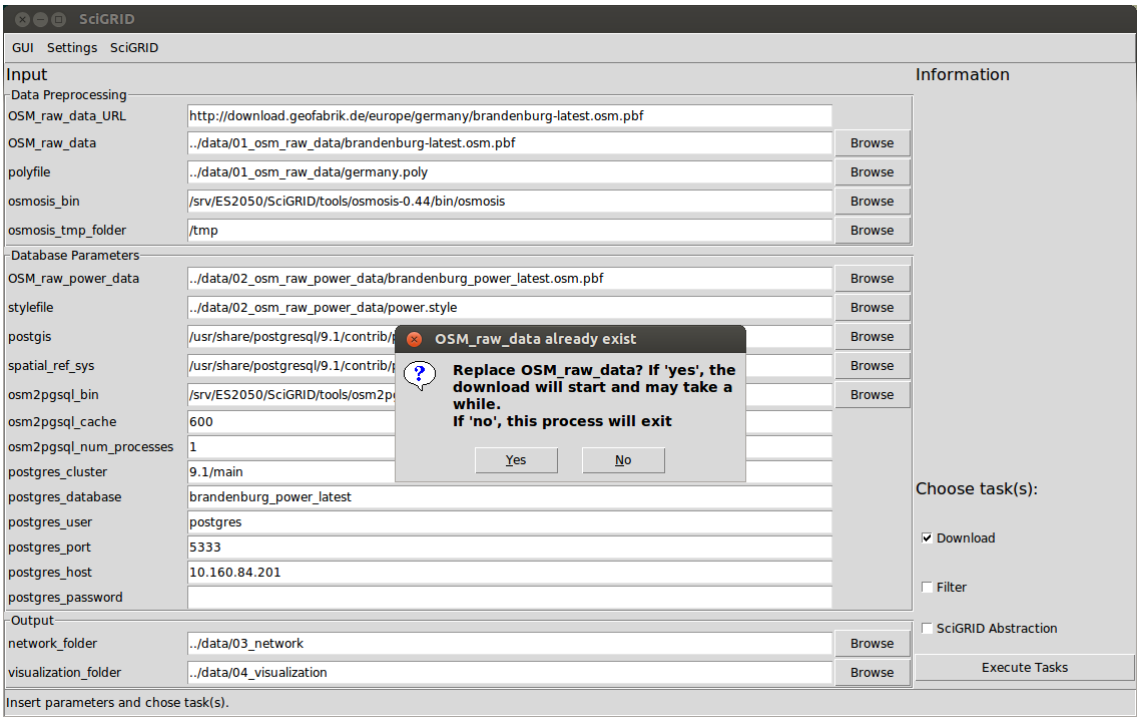


Figure 28: Message box appearing when the planet file already exists and the download process is executed.

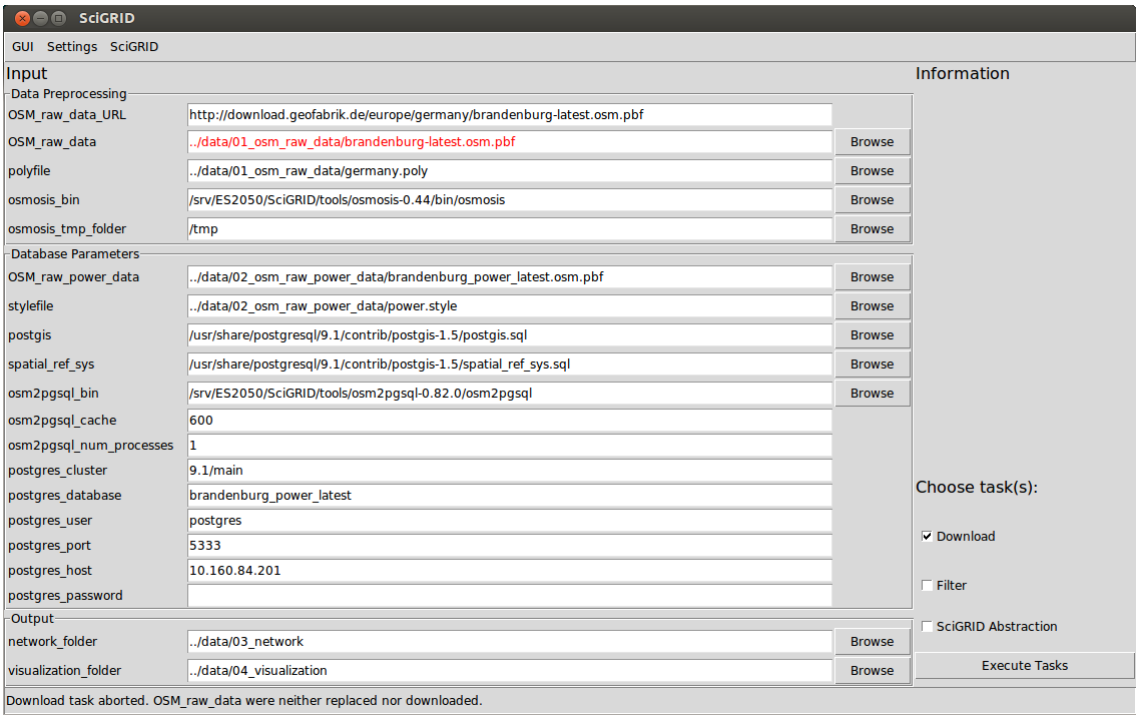


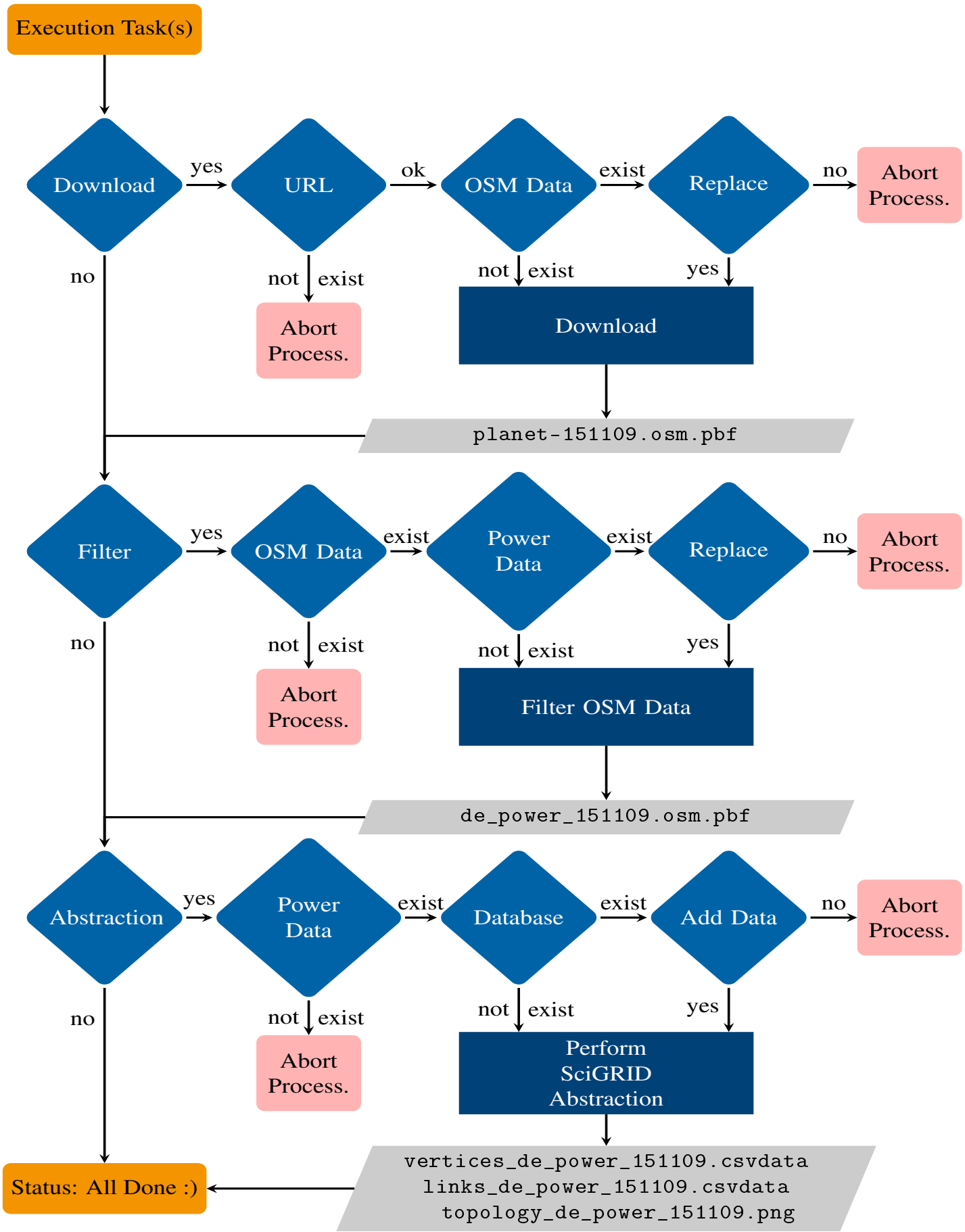
Figure 29: Red highlighted entry field to indicate the already existing file location.

The *Filter* task executes the *osmosis* filtering command introduced in Section 3.3. A log file, *osmosis.log*, containing the output message of the *osmosis* command is created in the

`SciGRID/code/log` folder. If the filtered power data file already exists, a message box appears and the user can either keep the file or execute the filtering command again. When keeping the existing power data file the process of filtering is aborted.

When choosing two or more tasks to execute, a prompt asks if the user wants to replace the corresponding file (if already exists). For example, if the tasks *Filter* and *SciGRID Abstraction* are both selected and a filtering is already found, a message box appears and the user can choose to either keep the already filtered data or to abort the execution of the tasks.

The above cited two examples are an example of the dependency tracking option implemented in the **SciGRID** GUI. For a list of all dependencies tracking see the following chart:


Execution Task(s)
Download
yes
URL
ok
OSM Data
exist
Replace
no
Abort Process.
no
not
exist
Abort Process.
not
exist
yes
Download
planet-151109.osm.pbf
Filter
yes
OSM Data
exist
Power Data
exist
Replace
no
Abort Process.
no
not
exist
Abort Process.
not
exist
yes
Filter OSM Data
de_power_151109.osm.pbf
Abstraction
yes
Power Data
exist
Database
exist
Add Data
no
Abort Process.
no
not
exist
Abort Process.
not
exist
yes
Perform SciGRID Abstraction
Status: All Done :)
vertices_de_power_151109.csvdata
links_de_power_151109.csvdata
topology_de_power_151109.png

The task *SciGRID abstraction* export of power data into the database and executes the abstraction of the power data introduced in Section 3.5. The log files *osm2pgsql.log* and *abstraction.log* are created in the `SciGRID/code/log` folder. Additionally, two files *database_import.done* and *abstraction.done* are created when the database import and abstraction steps are successfully executed. The `.log` and `.done` files serve a debugging purpose and help identifying errors when executing task(s).

5 Acknowledgements

The authors and developers of **SciGRID** acknowledge the funding by the German Federal Ministry of Education and Research (BMBF) through the funding initiative "Zukunftsfähige Stromnetze" (funding code 03SF0471). Furthermore, the authors would like to thank the users of **SciGRID** for their constructive feedback. Special thanks to Jonas Hörsch for the fixes he sent, which were included in this release v0.2.

6 Troubleshooting

- Avoid naming databases or data files using the '-' symbol. This will create an error when using psql and *osm2pgsql*.
- Avoid using capital letters to name databases or data files. This will create an error when using psql and *osm2pgsql*.
- If you are using pgAdmin III while running the **SciGRID** model, an error may occur when the databases are selected in pgAdmin III. You need to close pgAdmin III while running the **SciGRID** model.
- When executing the **SciGRID** model on Mac, the paths to `postgis.sql` and `spatial_ref_sys.sql` (both in the *Contrib* folder of PostgreSQL folder) may be different that the one on a Linux machine. The *PostgreSQL* folder on Mac is usually in the `/Library` folder.
- To localize the files `potgis.sql` and `spatial_ref_sys.sql` type the following commands in a terminal:

```
find . -name 'postgis.sql'
```

```
find . -name 'spatial_ref_sys.sql'
```

7 How to?

7.1 How to install osmosis?

7.1.1 On Linux

The installation of *osmosis* in the latest version is done in three steps.
Open a shell script, and type the following command to download the *osmosis* build-in package for Linux:

```
$ wget http://bretth.dev.openstreetmap.org/osmosis-build/osmosis-latest.tgz
```

Then, unpack the downloaded *osmosis* build-in package by typing:

```
$ tar xvfz osmosis-latest.tgz
```

Finally, to be able to use osmosis, change the rights of the *osmosis* binary file by typing:

```
$ chmod a+x bin/osmosis
```

7.1.2 On Mac OS

The easiest way to install *osmosis* on a Mac is to use homebrew [22]. In a shell terminal type:

```
$ brew install osmosis
```

7.2 How to install PostgreSQL?

7.2.1 On Linux

To install the built-in PostgreSQL and PostGIS for Linux, type the following command in a shell terminal:

```
$ sudo apt-get install postgresql-9.1 postgis
```

7.2.2 On Mac OS

To install PostgreSQL on Mac, a graphical installer is available, which includes PostgreSQL, and the StackBuilder utility for installation of additional packages. For PostgreSQL 9.0 and 9.1, Mac OS X 10.5 and above are supported, on 32 and 64-bit Intel CPUs, and PostgreSQL 9.2 and later support Mac OS X 10.6 and above on 32 and 64-bit Intel CPUs. The installer can be

downloaded from the webpage [23].

There is also a possibility to install PostgreSQL by downloading the Postgres.app, which is a simple, native Mac OS X app that runs in the menu bar without the need of an installer. After downloading the Postgres.app [24], open it, and a PostgreSQL server is then ready and awaiting new connections. To shut the server down, you just need to close the app.

7.3 How to install osm2pgsql?

A detailed webpage [25] on the OpenStreetMap wiki page exists where it is explained how to install *osm2pgsql* on Linux and Mac OS systems.

7.4 How to obtain gpx files?

Here we explain how to obtain a gpx file representing the landmass (or borders) of a country.

- Look up the country in the webpage [26] and extract the relation number representing the country's landmass. For example, the relation ID=62781 represent the German landmass.
- Use overpass-turbo.eu to visualize the relation, using the command:

```
[out:json];
( rel(62781) );
out ;
>;
out ;
```

- Use overpass-turbo.eu "Export" button in the menu bar and export the resulting relation data as a gpx file.
- The resulting `.gpx` files contain a very large number of points, for example the `.gpx` file of Germany contains 212089 points. Therefore, the `.gpx` file is simplified by deleting some points but still keeping enough points to obtain a sharp landmass representation. This is done by the python script `simplify_landmass.py` provided in the `SciGRID/data/04_-visualization` folder. For example, the output of the script `simplify_landmass.py` for the `de_landmass.gpx` file is the `de_landmass.txt` file which then is used by the `SciGRID/code/create_plots.py` script to plot the landmass border of Germany.

References

[1] Open Data Commons Open Database License. Odbl. `http://opendatacommons.org/licenses/odbl/`.

[2] OpenStreetMap. Copyright and license. `http://www.openstreetmap.org/copyright`.

[3] Open Database License. (odbl) v1.0. `http://opendatacommons.org/licenses/odbl/1.0/`.

[4] Database Contents License. (dbcl) v1.0. http://opendatacommons.org/licenses/dbcl/1.0/.

[5] Open Data Commons Open Database License. (odbl). http://opendatacommons.org/licenses/odbl/#sthash.C4HJvcBW.dpuf.

[6] Apache License. Version 2. http://www.apache.org/licenses/LICENSE-2.0.

[7] Research Center Next Energy. EWE-Forschungszentrum fur Energietechnologie e. V. http://www.next-energy.de.

[8] OpenStreetMap. http://www.openstreetmap.org.

[9] SciGRID webpage. Scigrid developers. http://www.scigrid.de.

[10] Ito world website. http://www.itoworld.com.

[11] Index of mirrors openstreetmap.org. openstreetmap.org planet data. http://ftp.heanet.ie/mirrors/openstreetmap.org.

[12] openstreetmap.org planet data. http://wiki.openstreetmap.org/wiki/Planet.osm.

[13] OpenStreetMap. project wiki webpage. http://wiki.openstreetmap.org.

[14] Osmosis. Detailed information about osmosis. http://wiki.openstreetmap.org/wiki/Osmosis#Detailed_usage.

[15] Databases and data access APIs. openstreetmap wiki webpage. http://www.postgresql.org.

[16] Postgis, a spatial database extender for postgresql. http://postgis.net/.

[17] Deutsche Energy-Agentur (dena). Ausbau- und Innovationsbedarf der Stromverteilnetze in Deutschland bis 2030. Technical report, 2012.

[18] QGIS. A free and open source geographic information system. http://www2.qgis.org.

[19] Geofabrik Downloads. Openstreetmap data extracts. http://download.geofabrik.de.

[20] Planet.OSM. http://wiki.openstreetmap.org/wiki/Planet.osm.

[21] OpenStreetMap. Countries of europe landmass relations. http://wiki.openstreetmap.org/wiki/Template:Countries_of_Europe.

[22] HomeBrew. http://brew.sh/.

[23] EnterpriseDB. Download postgresql. http://www.enterprisedb.com/products-services-training/pgdownload#osx.

[24] Postgres.app. Download postgres.app. http://postgresapp.com/.

[25] OpenStreetMap Wiki Page. Install osm2pgsql. http://wiki.openstreetmap.org/wiki/Osm2pgsql#Installation.

[26] Python GUI package. Tkinter. https://wiki.python.org/moin/TkInter.