

ConnectOTP : Workshop

COLLABORATORS

	<i>TITLE :</i> ConnectOTP : Workshop		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Mehdi Maazouz	March 24, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Introduction	1
2	Environment	2
3	Getting project	3
3.1	Github	3
4	Prepare Database	4
5	Prepare OpenTripPlanner server	5
6	Prepare QGIS	6
7	Generate MobilityDB trips	8
7.1	A Multi-Modal Trip	8
7.2	People moving in Brussels	11
7.3	Comparison with alternatives itineraries	12

Abstract

This workshop is about generating trajectories using [OpenTripPlanner](#) and connect [MobilityDB](#) on those trajectories. This workshop is open source and its code is available on [Github](#). The visualization will be done by [Qgis](#).

ConnectOTP is developed by the student Maazouz Mehdi in the framework of his thesis in computer science in the Université Libre de Bruxelles (ULB) under the direction of Prof. Esteban Zimányi.

This is the workshop for ConnectOTP. Feel free to use this material any way you like.

Chapter 1

Introduction

This workshop aims to show you how to connect [OpenTripPlanner](#) and [MobilityDB](#) and to display the generated trips on Qgis. The generated trips are multi-modal and focus on means of transport obtained via GTFS data.

In this document you will find the tools used for the realization of this workshop as well as the different steps for the creation and visualization of the trips.

Chapter 2

Environment

This workshop was run on a Ubuntu 18.04 LTS Linux machine, 8GB Ram, Intel Core I5-5300U CPU and 512GB SSD. In addition, the following tools made this workshop possible

- Java 11 (OpenTripPlanner 2 requires Java 11 or later)
- PostgreSQL 13.6
- Postgis 2.5.5
- MobilityDB 1.0
- Python 3.7
 - module psycopg2 (installed via PyPi)
- QGis 3.22 Białowieża and its module
 - Move (please follow instructions on [github](#))

Chapter 3

Getting project

3.1 Github

Please go to the official [github](#) page of the project and download the various files there, or simply clone the project. Here is the command to clone the repository

```
$ git clone https://github.com/MaazouzMehdi/ConnectOTP.git
```

This project contains several files, here is a brief overview :

- doc/ contains the necessary files to generate this workshop.
- src/ contains the sql and python files that will allow the generation of MobilityTrips.
- otpData/ contains the files necessary for the WEB OTP server to function properly. These files are :
 - brussels.pbf corresponds to the brussels.osm file reduced
 - stib-gtfs.zip corresponds to the GTFS data of the Stib covering the months of March 2022 to April 2022
 - tec-gtfs.zip corresponds to the GTFS data of the TEC covering the months of March 2022 to May 2022
 - build-config.json contains options and parameters that are taken into account during the graph building process

Chapter 4

Prepare Database

For the moment, we will use the OSM map of Brussels. The data was fetched via the OpenStreetMap API (Overpass API) and then reduced into a zip file. The result can be found in the root of the project in the file ./brussels.zip. Please extract this file in order to get ./brussels.osm file. Then, open a console

```
$ createdb -h localhost -p 5432 -U dbowner brusselsOTP  
-- replace localhost with your database host, 5432 with your port,  
-- and dbowner with your database user
```

this command line allows the creation of the brusselsOTP database that we will use throughout this workshop

ConnectOTP is a project that depends on Postgis, MobilityDB as well as Hstore. This is why we need to add these extensions to our database

```
-- create needed extensions  
$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP -c 'CREATE EXTENSION hstore'  
  
$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP -c 'CREATE EXTENSION postgis'  
$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP -c 'CREATE EXTENSION MobilityDB'  
-- adds the PostGIS and the MobilityDB extensions to the database
```

The last two lines can be executed in a single line by using the CASCADE parameter which install Postgis extension before installing MobilityDB extension .

```
$ psql -h localhost -p 5432 -U dbowner -d brusselsOTP -c 'CREATE EXTENSION MobilityDB ←  
CASCADE'
```

This last command will allow us to create a topology for the city of Brussels by creating two tables, namely ways as well as ways_vertices_pgr

```
$ osm2pgrouting -W 'password' -h localhost -p 5432 -U dbowner -f brussels.osm -d ←  
brusselsOTP  
-- replace password by your dbowner's password
```

the table ways_vertices_pgr brings us more than 116000 nodes located in the Brussels area. It is from these points that we will choose our source nodes and target nodes.

Chapter 5

Prepare OpenTripPlanner server

In order to contact the OpenTripPlanner API, we will set up a local web server (for now). The .jar file `otp-2.0.0-shaded.jar` is located at the root of the project and corresponds to OpenTripPlanner 2.0

When it is executed, it will use the files in `otpData/` to create the graph. By default, when the graph is created, the transit network will contain the GTFS data of the Tec and the Stib. If you want to add a GTFS file, please put it in the `otpData/` folder and add its name in the `build-config.json` file like this.

- "gtfs" : "tec-gtfs.zip|stib-gtfs.zip|my-gtfs.zip"

For example if we remove the GTFS file `tec-gtfs.zip` in our `build-config.json` we've got :

- "gtfs" : "stib-gtfs.zip"

Please be careful, for OpenTripPlanner 2 to detect a GTFS file, its name must end in `.zip` and must contain the letters. Now all we have to do is launch our server.

```
-- at the root of the project:  
$ java -Xmx4G -jar otp-2.0.0-shaded.jar --build --save ./otpData/
```

The parameter `-Xmx4G` means that you reserve 4GB of memory in order to build the graph containing our OSM and GTFS data from Stib and Tec. If you decide to use only OSM and Stib data, the parameter `-Xmx2G` would be sufficient. The parameter `--build` means the graph is build and `--save` means we store the graph onto disk. To start the server

```
-- at the root of the project:  
$ java -jar otp-2.0.0-shaded.jar --load --serve ./otpData/
```

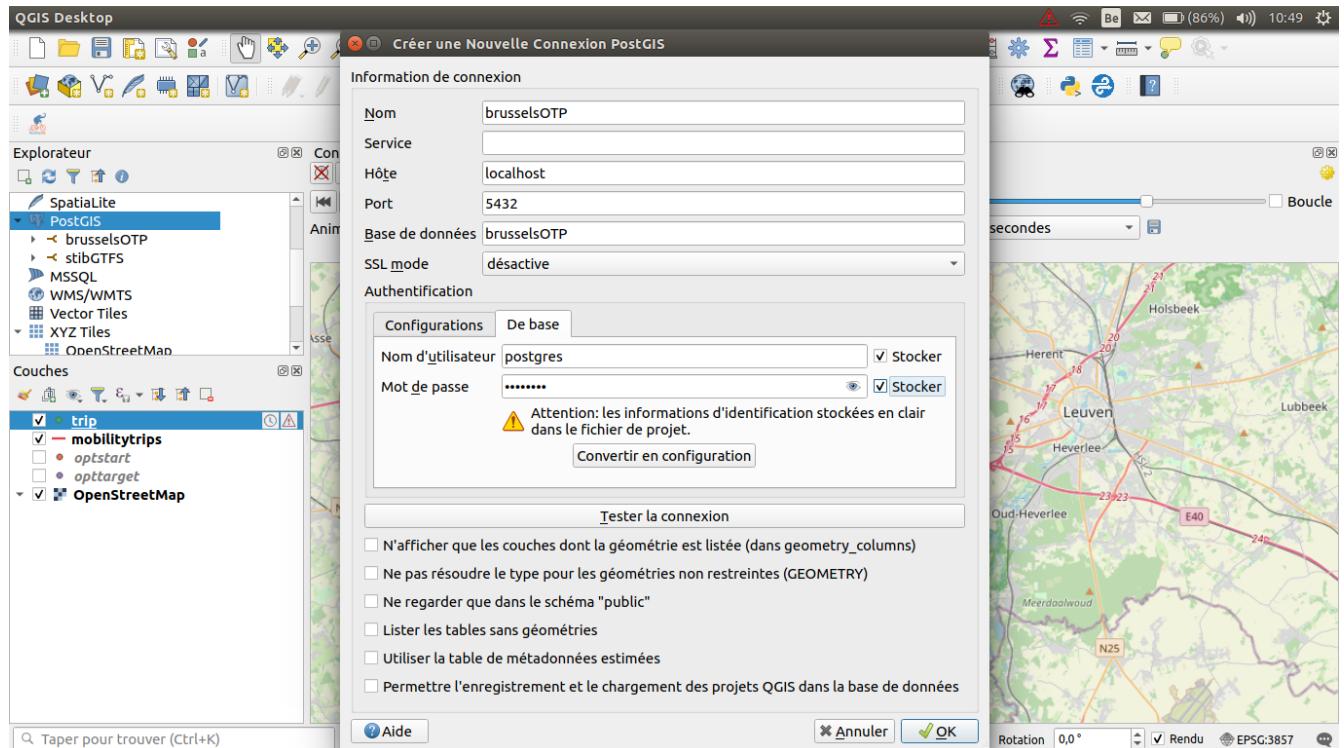
`--load` means we load the graph just built before and `--serve` simply means we run an OTP API server

Chapter 6

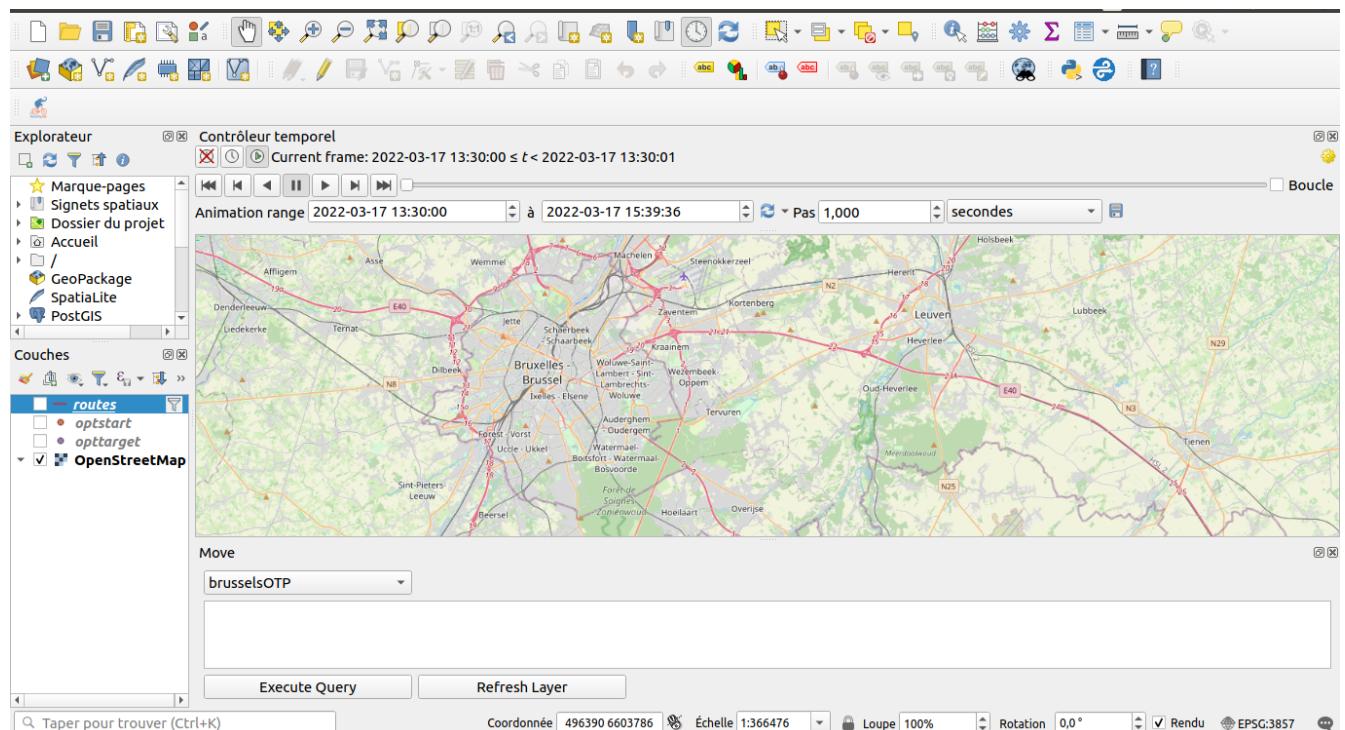
Prepare QGIS

To display the trips, the Qgis tool is used. Instructions for downloading and installing it can be found at this [address](#)

When connecting Qgis with your database, please enable basic authentication. Otherwise, there is a risk that the Move module do not work properly



After installing Qgis and its Move module, you should be able to access to the below window. If you have any problems, please refer to the [Move](#) github page



Chapter 7

Generate MobilityDB trips

Once the various tools have been installed and configured. We can start generating MobilityDB trips from trips obtained via OpenTripPlanner.

7.1 A Multi-Modal Trip

First, we will generate a multi-modal trip for a person wanting to travel from a point A to a point B in Brussels. To do this, we will randomly generate a source and a target nodes that will be used in the calculation of the trip.

```
$ psql -h localhost -U postgres -d brusselsOTP -f ./src/comboination.sql -v o=1  
-- -v o=1 means we want to generate 1 source node and 1 target node
```

We will now contact the OpenTripPlanner API to generate the trip, and parse it into a PostgreSQL database called `routes`. Please make sure you have started the OTP server before doing this.

```
-- in a console at the root of your project :  
$ python ./src/manageOTP.py localhost brusselsOTP dbowner password  
  
-- as usual, replace dbowner by your database user  
-- replace password by your database user's password
```

Running this command will prompt you to enter parameters to better define the trip you want to create. Below are some examples of parameters you might want to enter. OpenTripPlanner is full of parameters, a exhaustive list can be found at this [address](#)

the desired format is presented as follows : `key=value`

Here is an example you could enter in the console when you run `manageOTP.py`

```
please enter the desired parameters ( key=value ) : arriveBy=true optimize=QUICK
```

You could find below some parameters you put enter :

- `date=2022-03-10`
- `time=13:23`
- `mode=WALK,BIKE` (represents transport modes to consider)
- `wheelchair=true`
- `arriveBy=true` (specifies that the given time is when we plan to arrive)
- `maxWalkDistance=1000` (specifies the maximum distance in meters that you are willing to walk)

If you wish, you can press "Enter" directly. In this way, the default values will be taken into account. Note that the default date represents the current day, the default time represents the current time.

Please note that the baseline GTFS data covers the months of March 2022. If you are doing this workshop later, please enter a specific date that is taken into account by the GTFS data

For the sake of this workshop, we have changed the default value of the mode parameter. Originally WALK, it is now WALK, TRANSIT. The reason being that this workshop deals with multi-modal routing, so there is no point in using only your feet to do the trips. Moreover, we changed the default value numItineraries=1 by numItineraries=2 mainly because the default value only takes account the WALK trip.

In our case, when the prompt below appears, we click immediately on Enter

```
please enter the desired parameters ( key=value ) :
```

Now we just need to run one last sql script to generate the mobility trip.

```
-- in a console at the root of your project
$ psql -h localhost -U dbowner -d brusselsOTP -f ./src/generateMobility_Trips.sql
-- We generate MobilityDB trips
```

We now have the mobilitytrips table containing our tgeompoinst. As well as a table stibtrip containing our tgeompoinst which represents the part by public transport, a table waittrip which represents the waiting for a transfer and a table walktrip which represents walking part.

Now let's go to Qgis and open a connection to our BrusselsOTP Database. Once done, click on the optstart and opttarget tables to display the source node and target node. The generated nodes are random, so there is a good chance that you will have other nodes

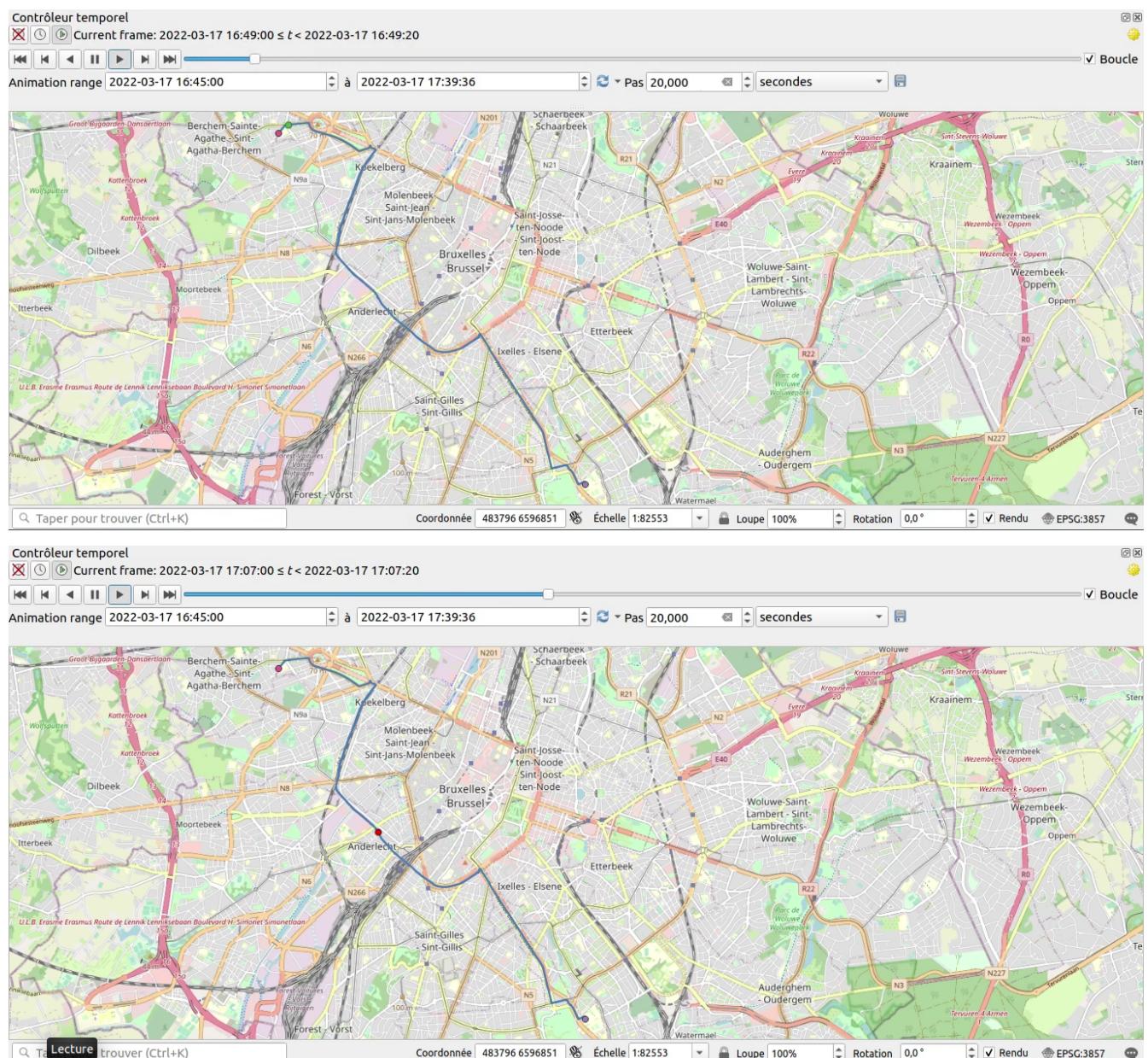
We can display our mobilitytrips table on QGis. However, in order to be able to visualize our points (people) moving in time, we will use the Move module.

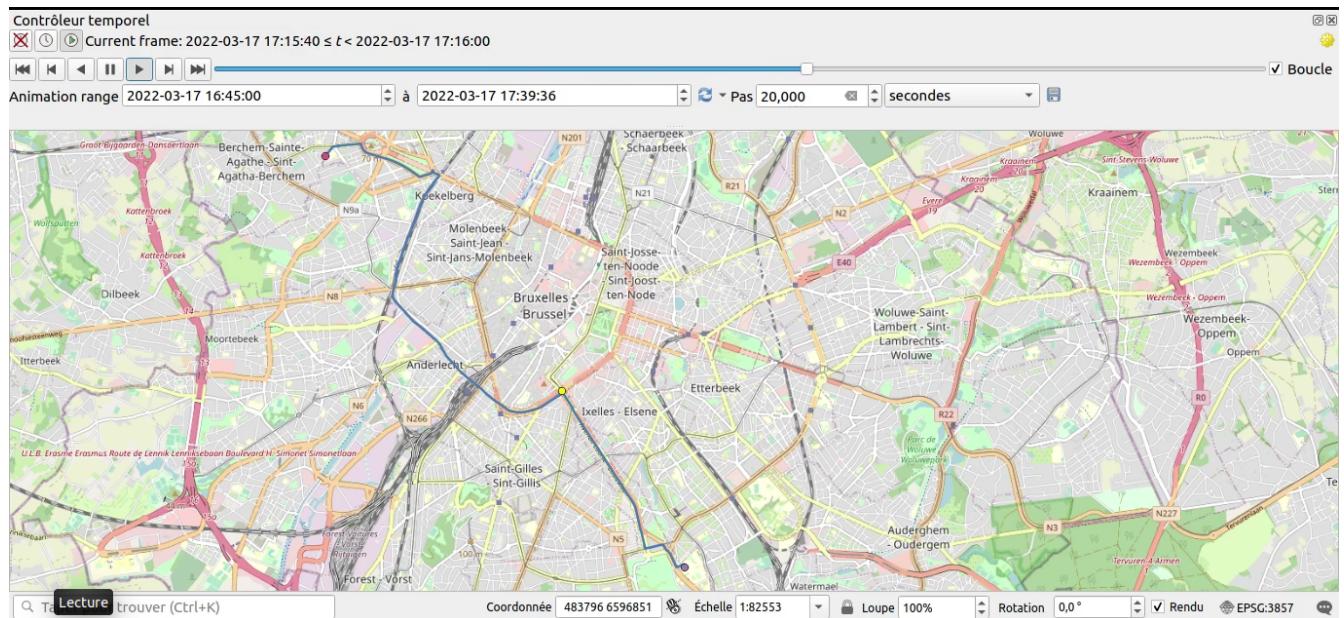
In order to visualize the points in a temporal way, we just need to use the Move plugin (installed previously). Click on the plugin to display a window at the bottom where you insert the following queries and then, click on execute. This will create temporal layers. Open the controller panel (View->Panels->Temporal Controller Panel) and you can then see the moving points move over time.

```
-- on the plugin Move in Qgis
select trip as walktrip from walktrip
select trip as waittrip from waittrip
select trip as stibtrip from stibtrip
```

Please note that in Belgium we are in UTC+1. I'm currently having trouble displaying the correct time on Move. Therefore, if you generate trips between 04:00 pm and 04:30 pm for example, the display of the points on Move will be between 03:00 pm and 03h30 pm.

You should have a similar result, the green dot representing the person walking, the red dot representing the person moving in a public transport (tram, bus, metro) and the yellow dot representing the person waiting for a transfer





7.2 People moving in Brussels

We will now generate multi-modal trips for a multitude of people. The principle is the same as before, except for the number of sources and target nodes generated.

```
$ psql -h localhost -U postgres -d brusselsOTP -f ./src/combination.sql -v o=40
-- -v o=40 means we want to generate 40 source nodes and 40 target nodes
$ python ./src/manageOTP.py localhost brusselsOTP dbowner password

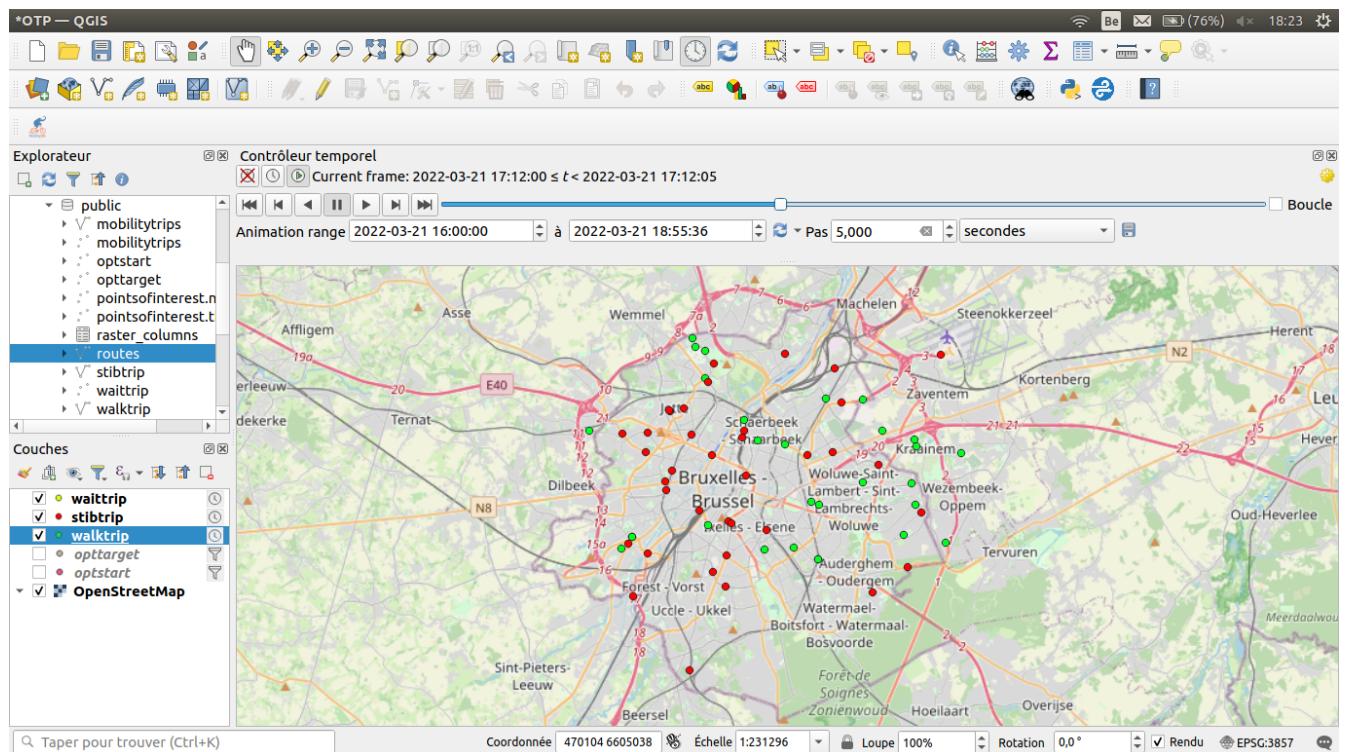
please enter the desired parameters ( key=value ) : ( Click on Enter )

$ psql -h localhost -U dbowner -d brusselsOTP -f ./src/generateMobility_Trips.sql
```

Inside Move module in Qgis

```
-- on the plugin Move in Qgis
select trip as walktrip from walktrip
select trip as waittrip from waittrip
select trip as stibtrip from stibtrip
```

Here is the result



7.3 Comparison with alternatives itineraries

You can also generate multiple itineraries for a single trip. In this workshop, we will take a simple example by comparing only 2 itineraries proposed to one person.

```
$ psql -h localhost -U postgres -d brusselsOTP -f ./src/combination.sql -v o=1
-- -- v o=1 means we want to generate 1 source nodes and 1 target nodes
$ python ./src/manageOTP.py localhost brusselsOTP dbowner password

please enter the desired parameters ( key=value ) : ( Click on Enter )

$ psql -h localhost -U dbowner -d brusselsOTP -f ./src/generateMobility_Trips.sql
```

But if you want, you could generate several itineraries for one trip by giving the value you want to the parameter numItineraries and generate several trips

Below you see the red line representing a multi-modal trip (the red dot representing a person taking a public transport) while the green line represents the trip by foot. Then, you see that the multi-modal trip is faster than the other one.

