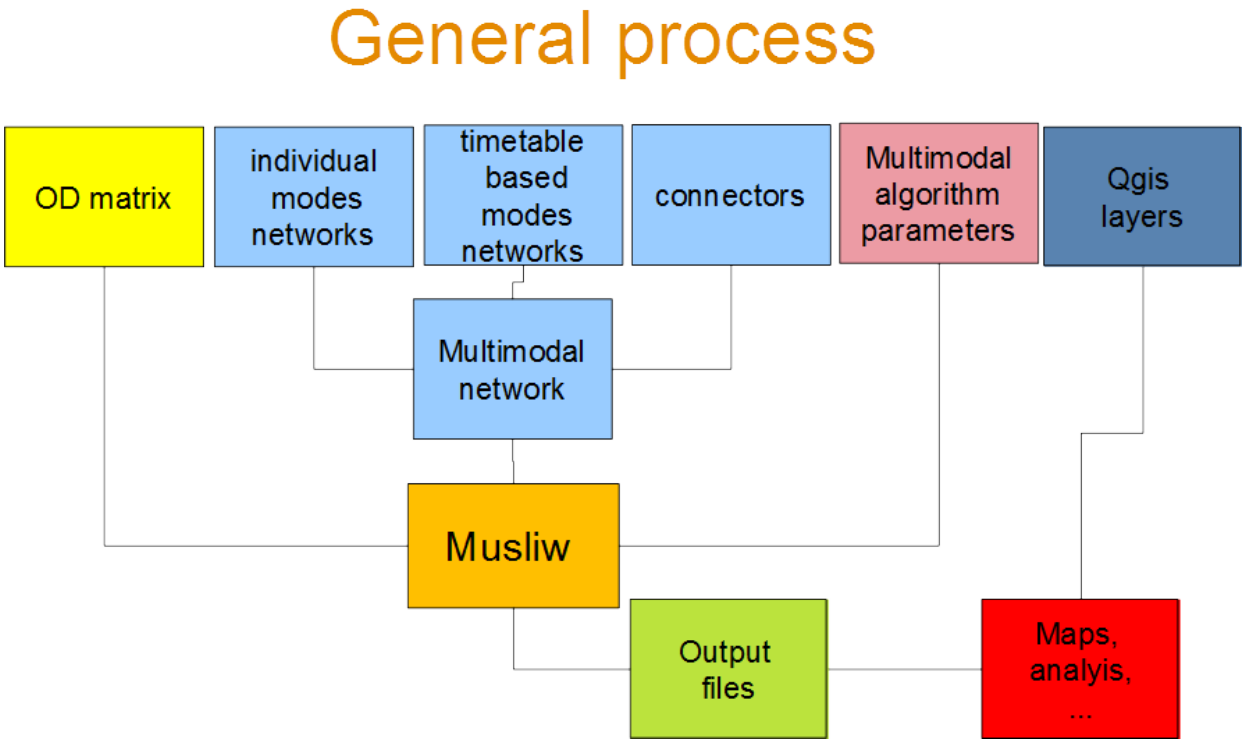
General process



Network

Individual transport modes networks (walking, cycling, car, …)

Example with OSM :

1. QuickOsm Plugin : To get OSM data (roads & streets) key='highway'
2. « **v.net.clean.advanced** » : To transform OSM layer into a planar graph (cut lines at junctions)
3. « **update field** » : To add new fields to the linear layer
    1. direction (type :char[1] value='1')
    2. length (type double value=$length)
    3. time (type double value=$length*60/4000 for a walking speed of 4km/h)
    4. spread (type char[1] value='3' if spread allowed from both sides of the road (attribute needed for isochrone with 'linear interpolation' algorithm)
    5. passable (type char[1] value='0' if every line element is passable (attribute needed for isochrone with 'linear interpolation' algorithm)

    If you are building a walking network for example, you have to set motorways as impassable with no spread allowed (→ direction='0' , spread='0' and passable='3')
4. « **reverse _arcs** »In general, it's better to have an oriented graph (easier to manage with results on links which are oriented). You have to reverse arcs and add them in the linear layer
5. « **build_graph** ». You have to build a graph with and start node id and end node id from each arc and generate a nodes layer.
6. « **Musliw individual network** ». Your have to create a musliw individual network component. You can have serveral individual networks (walking, cycling , car). Each component could have specific direction and travel time (ex : walk_direction, cyclcing_direction, walking_time, cycling attributes). When your create the musliw network component you can add a prefix before the node ids to differenciate modes (w for walking, b for bike,…)
7. « **Isolated nodes** ». If you want to keep only nodes that are connected to the main network you can use this algorithm which will not keep isolated nodes. It prevents to select non connected node later for routing algorithm

Timetable based modes networks (PT, flights, ferries, trains, …)

Timetable based modes are made from GTFS files :

1. « **Prepare GTFS** » : The script build a GTFS with adding in particular an network id (prefix) to differenciate stop_id, route_id ; trip_id,… from one network to another. The script can also manipulate stop_id string to extract your own ID (ex UIC code for rail stations). You can enter in this field any function combination you can apply on a Python string
2. « **GTFS import** » : The tool import a GTFS (made with prepare GTFS) and generates 3 layers
    1. The stops layer (with ID, name and number to PT stops during the calendar and the time period chosen)
    2. The lines layer (with correspond for each line to a succession of 2 consecutive stops)
    3. The arc layer (the same a line layer, but without the line Id information (→ no multiple arcs). This layer is used only to visualise total flows on arcs

       At the moment the algorithm doesn't deal with shapes.txt as shapes.txt are defined for trips and not for segments (two consecutive stops)
3. « **Musliw_timetable_network** ». This algorithm works from a GTFS file (made with prepare GTFS) and generates a Musliw time table component. Your have to choose a calendar period which is included in the GTFS calendar. You could several timetable networks components

Connectors

To allow intermodality, you have to generate connectors.

Connectors are virtual arcs from two Musliw network components (generally PT stops and roads nodes), but it can also be links between bike sharing stations to walking nodes or/and bike nodes to simulate the combination walking/bike share.

1. « **Build connectors** » generates arcs from a stops layer to a nodes layer (roads). It generates a Qagis layer and in the same time the corresponding Musliw network component (same name but with a txt extension).

You must take care of the mode id when generating connectors as it will enable you to disable or customize weights factors according to these ids.

Multi-modal transport network

You have to select every Musliw network component you need to build your own multimodal network (individual modes components, timetable based modes components, connectors components)

1. "**concatenate networks**". Choose the components your need. They will be added in one network file (global network). You can copy the components into a specific folder to help you selecting the right components

NB: If you want to specify custom turning movements or transfers, you need to edit the file manually as it doesn’t exist an algorithm for this at the moment.

Matrix

You need to generate a Musliw matrix file to specify the routing computation command you need.

A matrix containes information about the origin-destination nodes, the day and the desired departure or arrival time and the number of passengers for assignment.

3 algorithms are currently available. You have to choose the most relevant

- “**musliw simple matrix**”: This script generate a one od matrix by clicking the origin and destination points on a map. It can add the od to an existing matrxi file
- “**musliw matrix simple list**”: this script generates a square matrix (or diagonal) from a set of points
- “**musliw matrix double liste**”: this script generates a rectangle matrix from two sets of points

Parameters

- “**Musliw parameters**”: This algorithm generated a set of Musliw routing parameters. You must specify different weights, select different links types, customize output files and shortest paths algorithm parameters.

Multimodal routing computation

“**Musliw computation**”: This script is the core of the multimodal routing. It calls a .NET exe file “Muslic.exe” with the following arguments:

- Musliw network file
- Musliw matrix file
- Generic output file name
- Musliw parameters file
- Musliw turning movements of transfers (optional)

The algorithm works also on Linux, but you have to install the Mono runtime previously.

Several output files are produced depending of which were selected.

Result analysis

Accessibily map from links

If you want to produce an accessibility map based on , here is a process using the “networks” provider.

1. In musliw parameters, you should have selected “without timetable links” in the “output links times?” form
2. “**Update ti tj**”: This script will scan the <FILENAME>_temps.txt and will create (if they are not existing) or update fields (I often call “ti” and “tj” for time at node I, time at node j) form the links layer with the corresponding value in output file corresponding to cumulative travel times (or any other musliw output file field) at start node and end node.
3. “**Linear interpolation**”: This script will generate a grid by performing and deterministic line-based interpolation (http://geomatica.como.polimi.it/workbooks/n12/FOSS4G-eu15_submission_35.pdf).
4. “**Isovalue polygons**” generates isovalue polygons from the grid previously generated. Isovalue polygons enable indicators socioeconomic indicators based on multimodal accessibiliy

Accessibily map from nodes

You can also wants to produce an accessibility map based on nodes results. The map will be less precise, but you can others interpolation algorithms

1. In musliw parameters check “output nodes times”
2. Open in qgis the <FILENAME>_noeuds.txt (text file with “;” as delimiter)
3. Perform a join base on node ID with the nodes layer
4. Perform an interpolation algorithm (IDW, Delaunay, ….)
5. You can generate isovalue polygons with “**Isovalue polygons**”

Other analysis

Other types of analysis are available on github https://github.com/crocovert/scripts_qgis . These scripts are only in french and for Qgis 2.

They will be upgraded for Qgis 3 soon.

- calcul_socioeco.py: perform for example the sum of inhabitants in the isovalue polygons
- autoconnectors.py: generates connectors between nodes of the same layer in a certain radius
- decalage_lignes: generates attributes your can use to shift flows to produce a flow map with superposed links (PT map style, or traffic map (truck and car in different colors)

- fichier_aff.py: fill an attribute in the link layer with the total volume on links or on lines
- fichier_od.py: compute indicators for each od, when you have for have produced several calculation per od (ex: every 15 minutes during 2 hours)
- fichier_temps: the same as fichier_od.py but for each link instead of each od
- itineraires.py: computes paths (need a <filename>_temps.txt) and the link layer by selecting links and generates a routes layer
- arborescence.py: select one or several links. This will produces a flow map from all passengers who are using these links. (to know for example the routes used by people circulating on a bridge or entering into a specific station)
- trafic.py: Create flow map objects (polygons) with a nice cut consecutive lines.