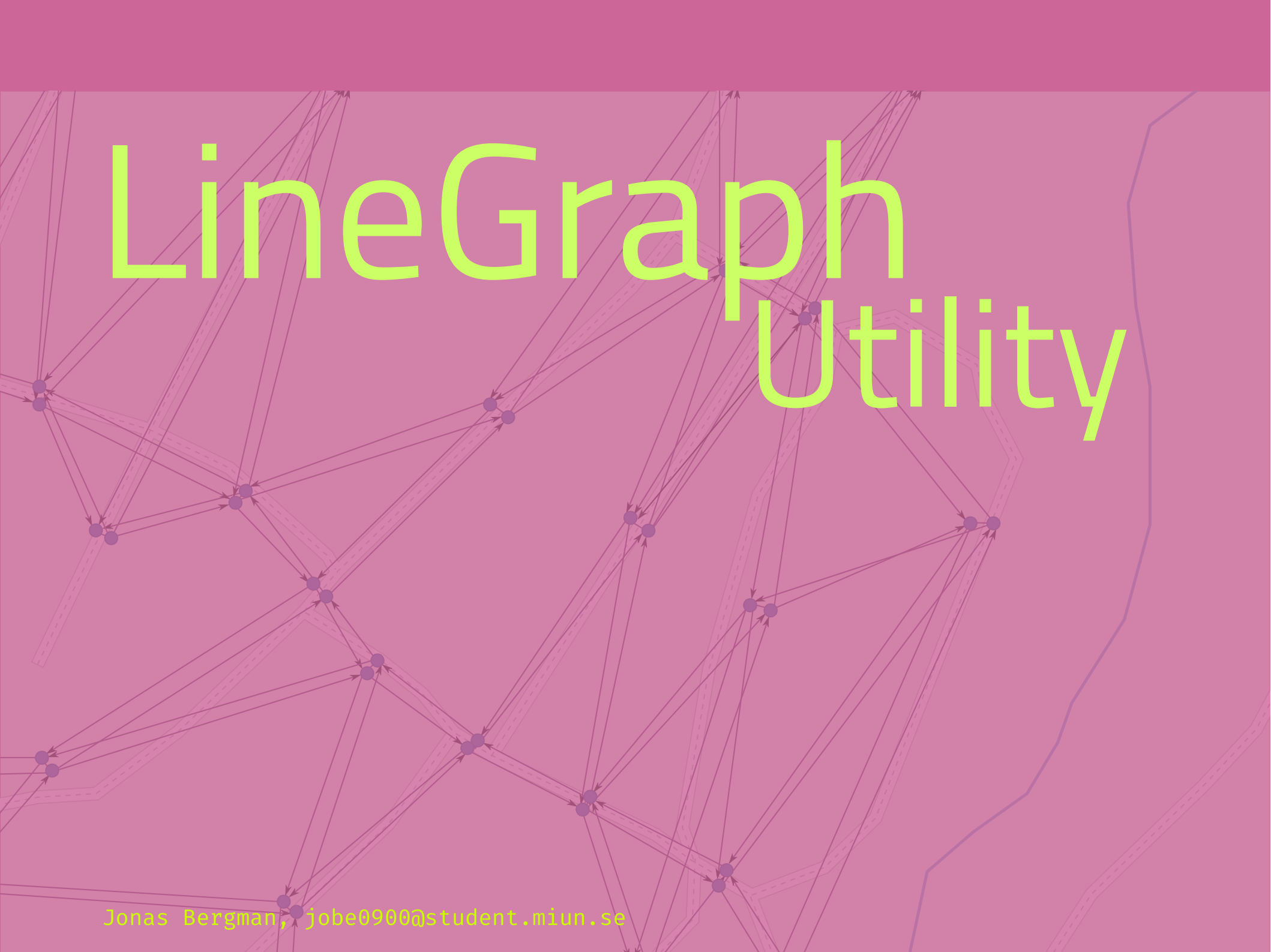
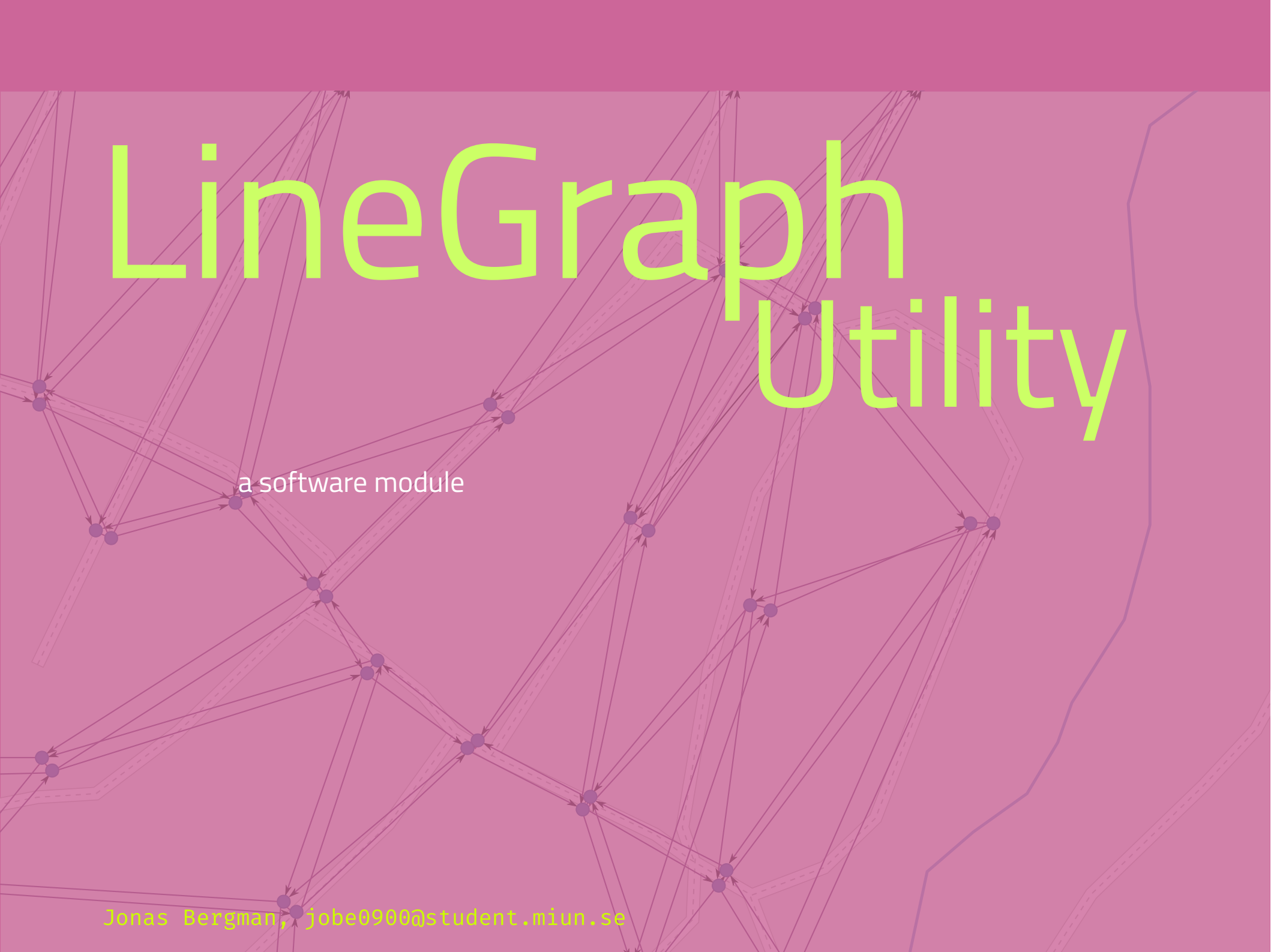


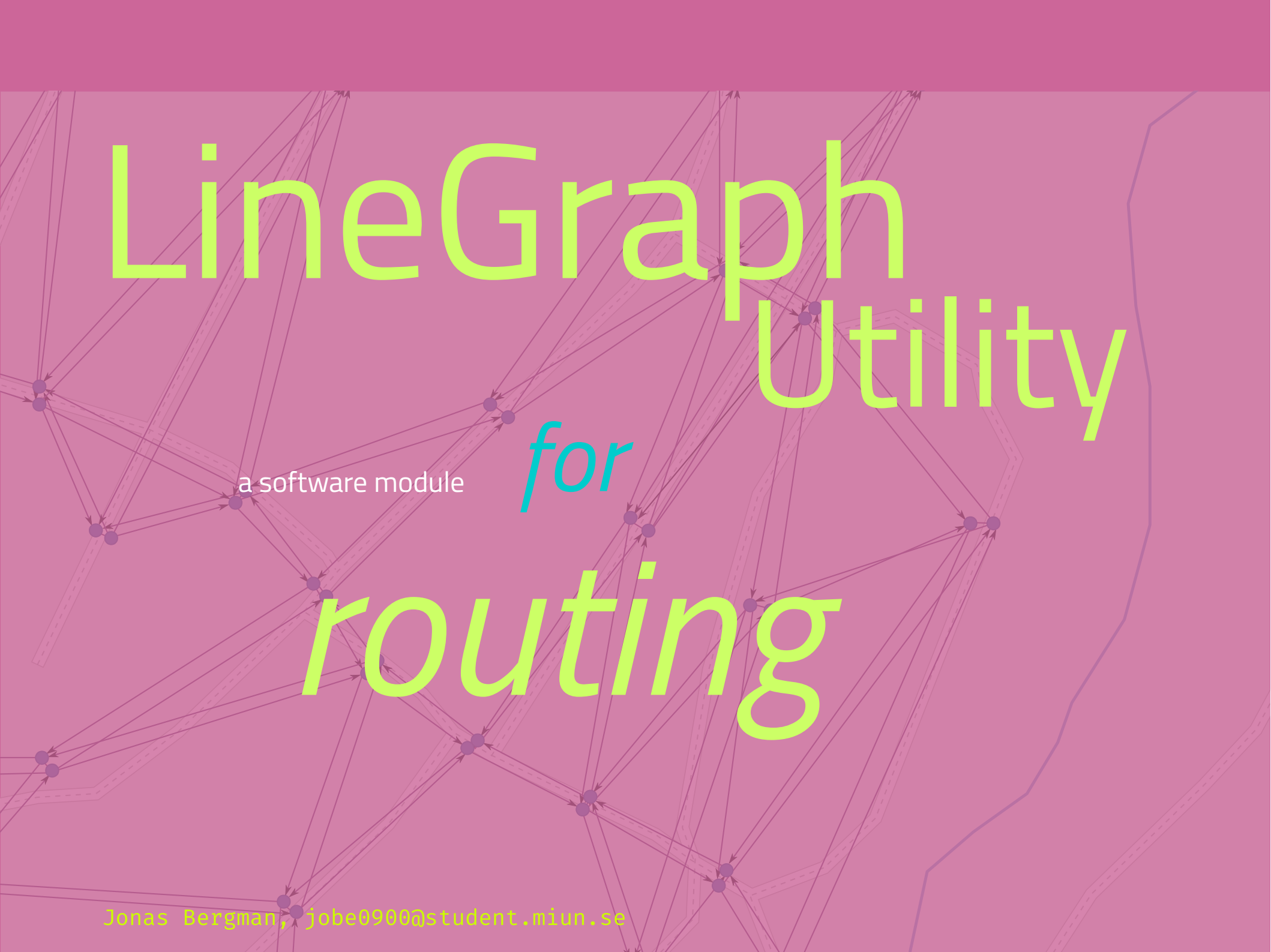
LineGraph Utility

The background of the slide features a light purple map of a road network. Overlaid on this map is a complex graph structure. The graph consists of numerous small, dark purple circular nodes. These nodes are interconnected by a dense web of thin, dark purple lines representing edges. Some of these edges are directed, as indicated by small arrowheads. The graph appears to be a utility or network graph, possibly representing a road network or a data flow system. The overall aesthetic is technical and modern.

LineGraph Utility

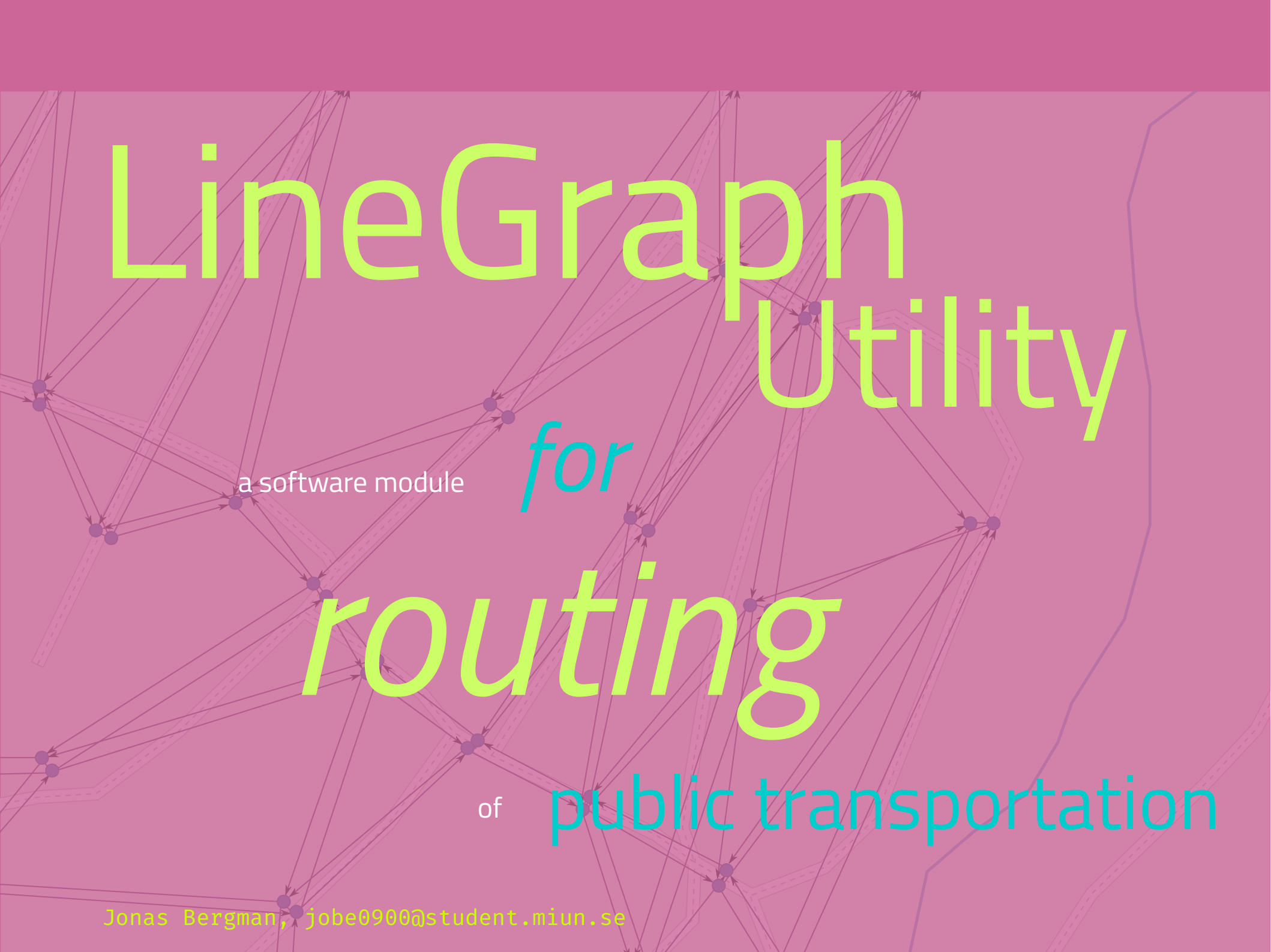
The background of the slide features a complex directed graph. Nodes are represented by small purple circles, and edges are thin black lines with arrowheads indicating direction. The graph is dense and interconnected, with some nodes having multiple incoming and outgoing edges. The overall aesthetic is technical and network-oriented.

a software module

A complex network diagram with numerous nodes (purple dots) and directed edges (purple arrows) is overlaid on a light purple background. The diagram represents a network topology, possibly for routing or data flow. The text 'LineGraph Utility' is in a large, bold, yellow-green font. The word 'for' is in a smaller, italicized, teal font. The word 'routing' is in a large, bold, yellow-green font, italicized. The phrase 'a software module' is in a smaller, white font.

LineGraph Utility

a software module *for* *routing*



LineGraph Utility *for* *routing* of public transportation

Jonas Bergman, jobe0900@student.mun.se

The background features a complex network of thin green lines and small green circular nodes. These lines and nodes form a web-like structure that spans the entire page, with some lines being thicker and more prominent than others. The overall aesthetic is technical and abstract, suggesting a network or a data structure.

1

background

1. background

managing flexible *public transportation*

1. background

flexible public transportation:

no
timetables

1. background

flexible public transportation:

- no timetables

commission
rides

1. background

flexible public transportation:

- no timetables
- commission rides

calculate
routes

1. background

flexible public transportation:

- no timetables
- commission rides
- calculate routes

update driving
instructions

flexible public transportation:

- no timetables
- commission rides
- calculate routes
- update driving instructions

gain?

1. background

flexible public transportation: **gain?**

less
idle vehicles

1. background

flexible public transportation: **gain?**

+ environment

less

idle vehicles

1. background

flexible public transportation: **gain?**

+ economy

+ environment

less

idle vehicles

1. background

flexible public transportation: **gain?**

- less idle vehicles

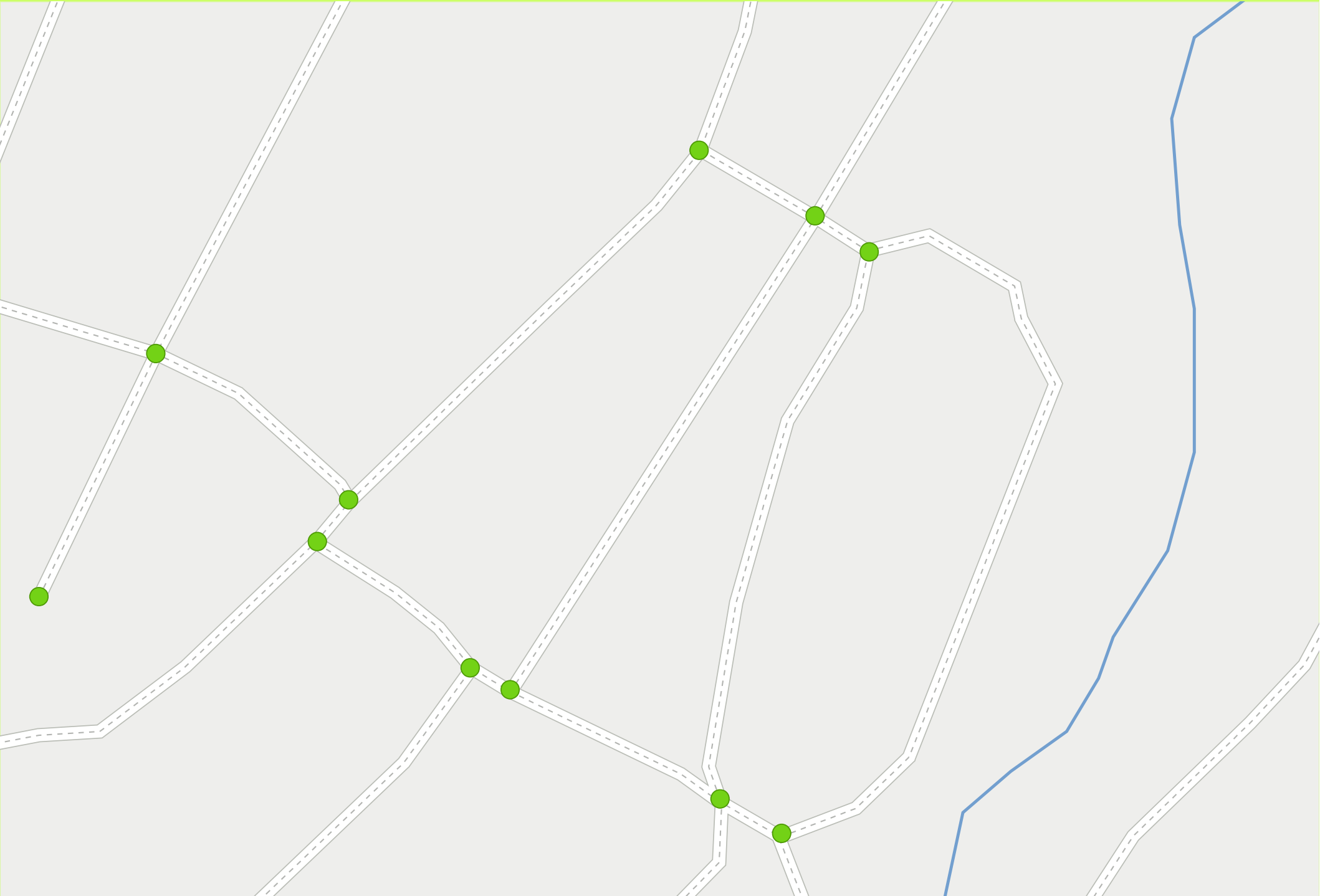
less

waiting

1. background > maps & graphs



1. background > maps & graphs



1. background > maps & graphs > topology



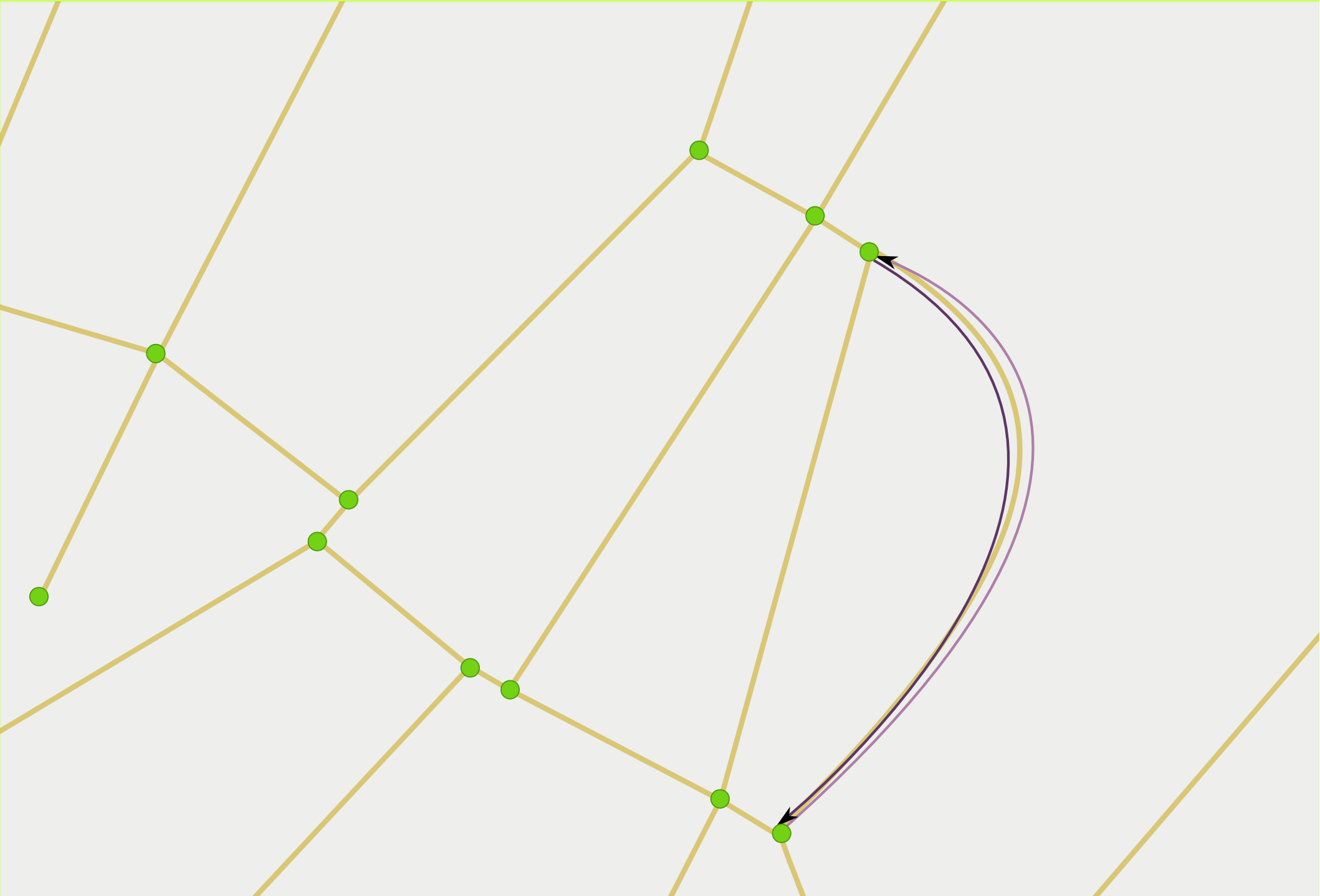
1. background > maps & graphs > topology



1. background > maps & graphs > directed graph



1. background > maps & graphs > directed graph



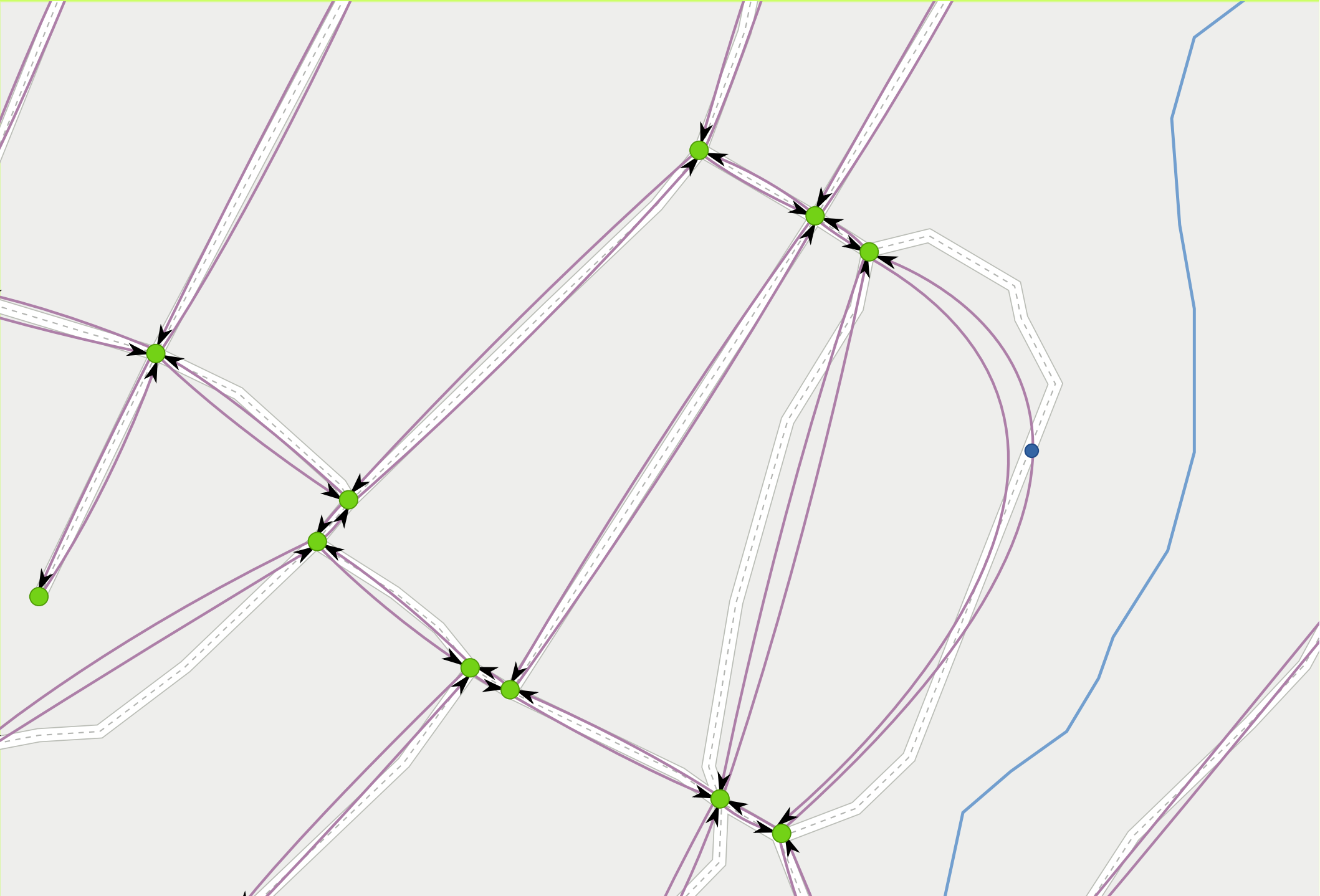
1. background > maps & graphs > directed graph



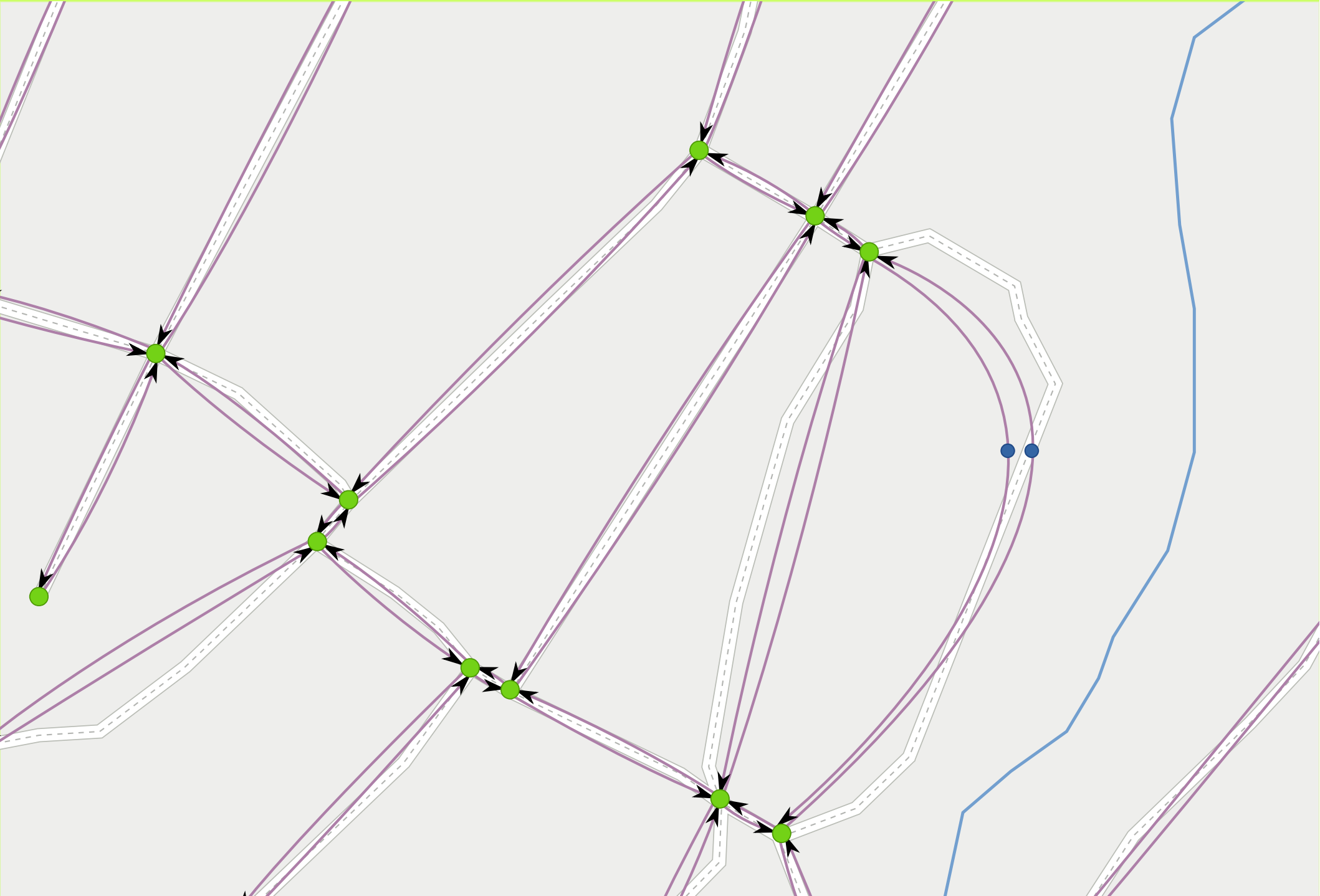
1. background > maps & graphs > directed graph



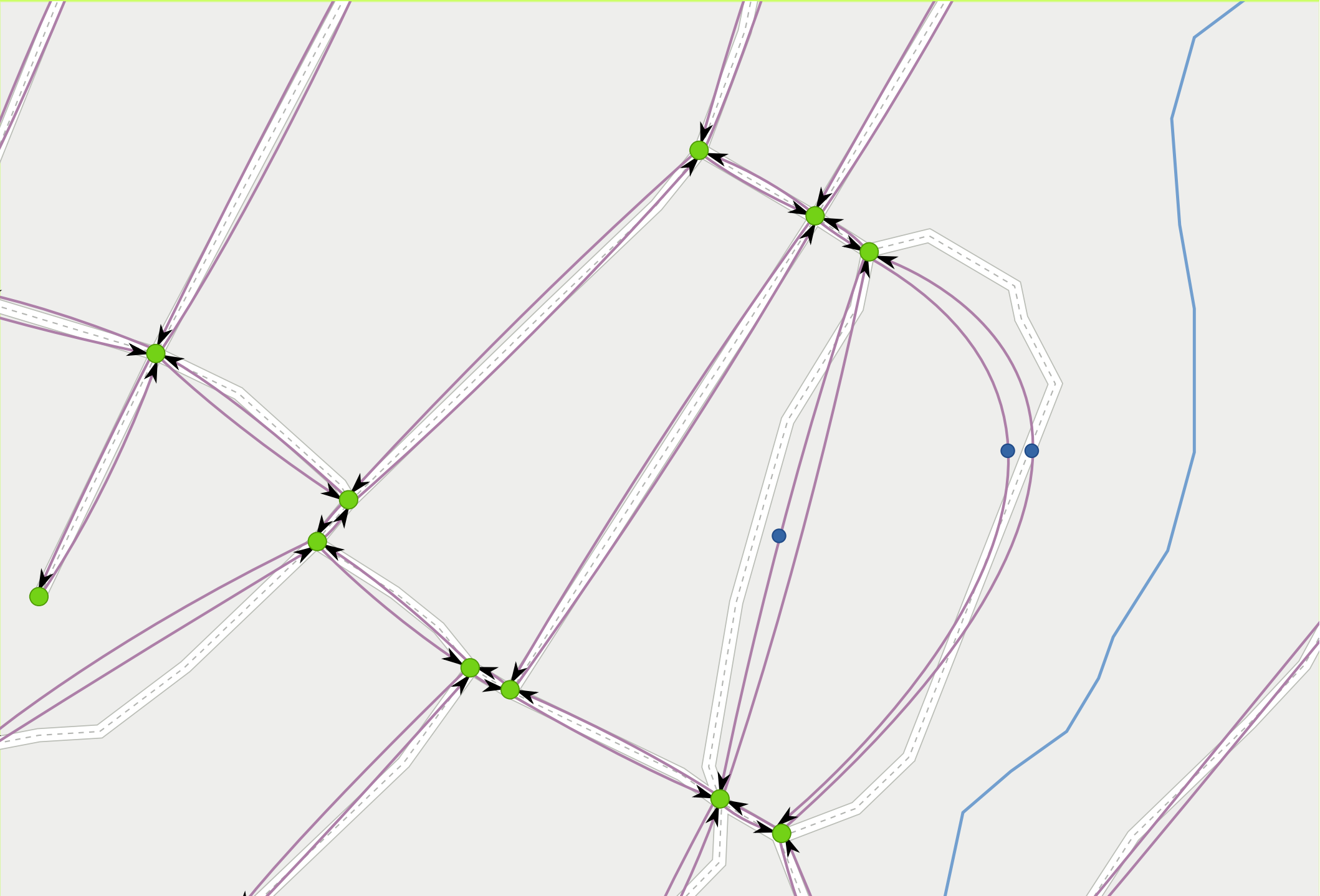
1. background > maps & graphs > line graph



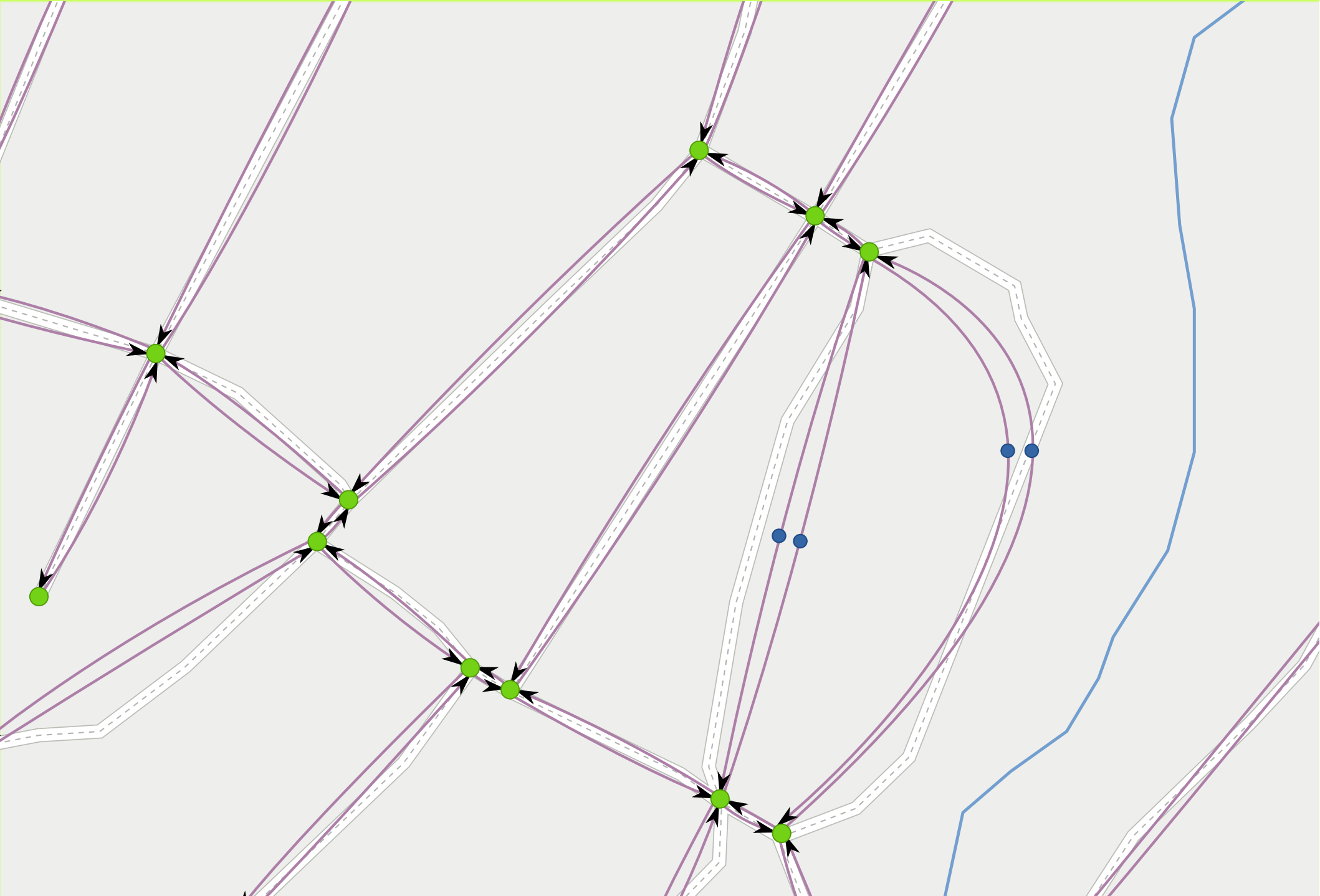
1. background > maps & graphs > line graph



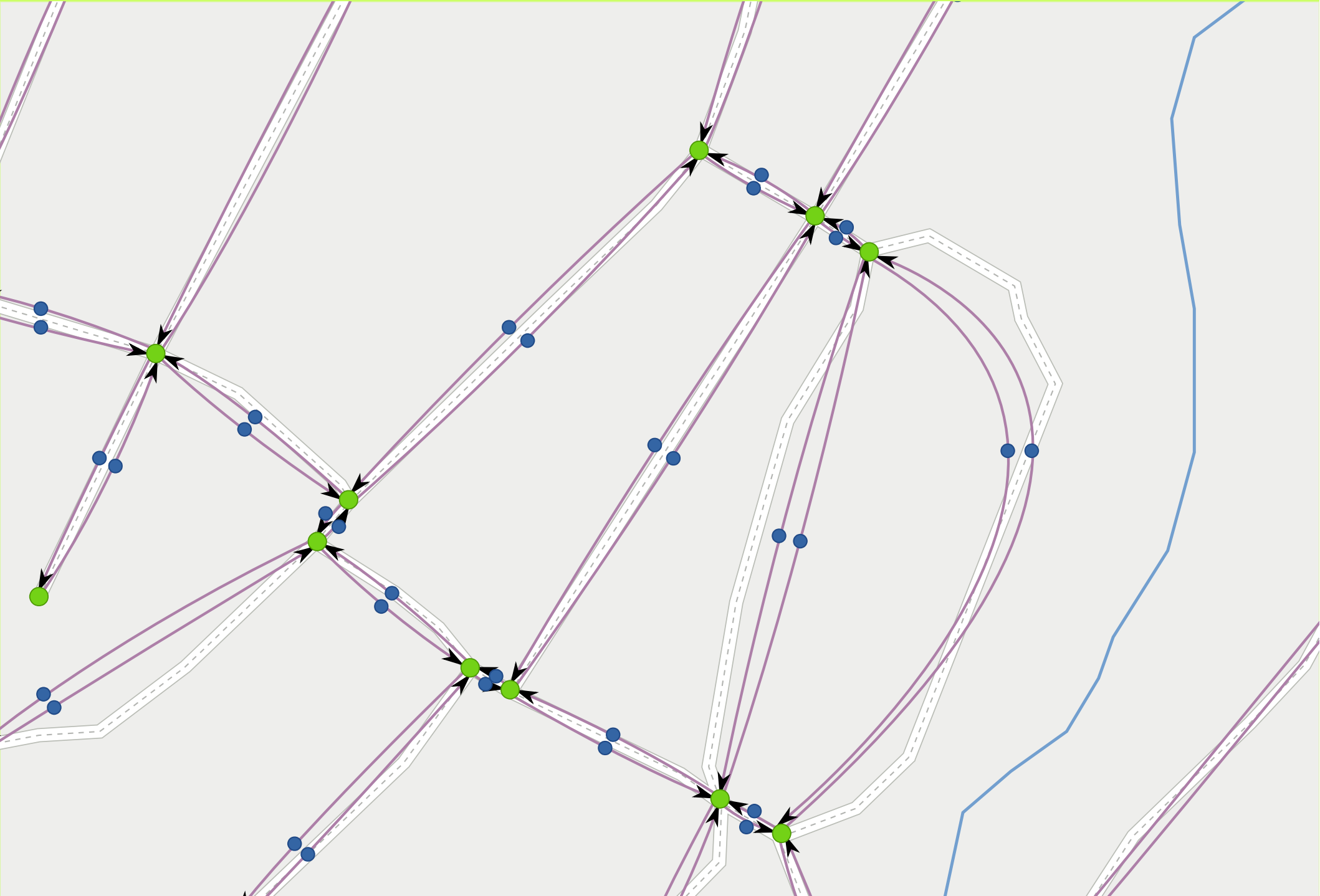
1. background > maps & graphs > line graph



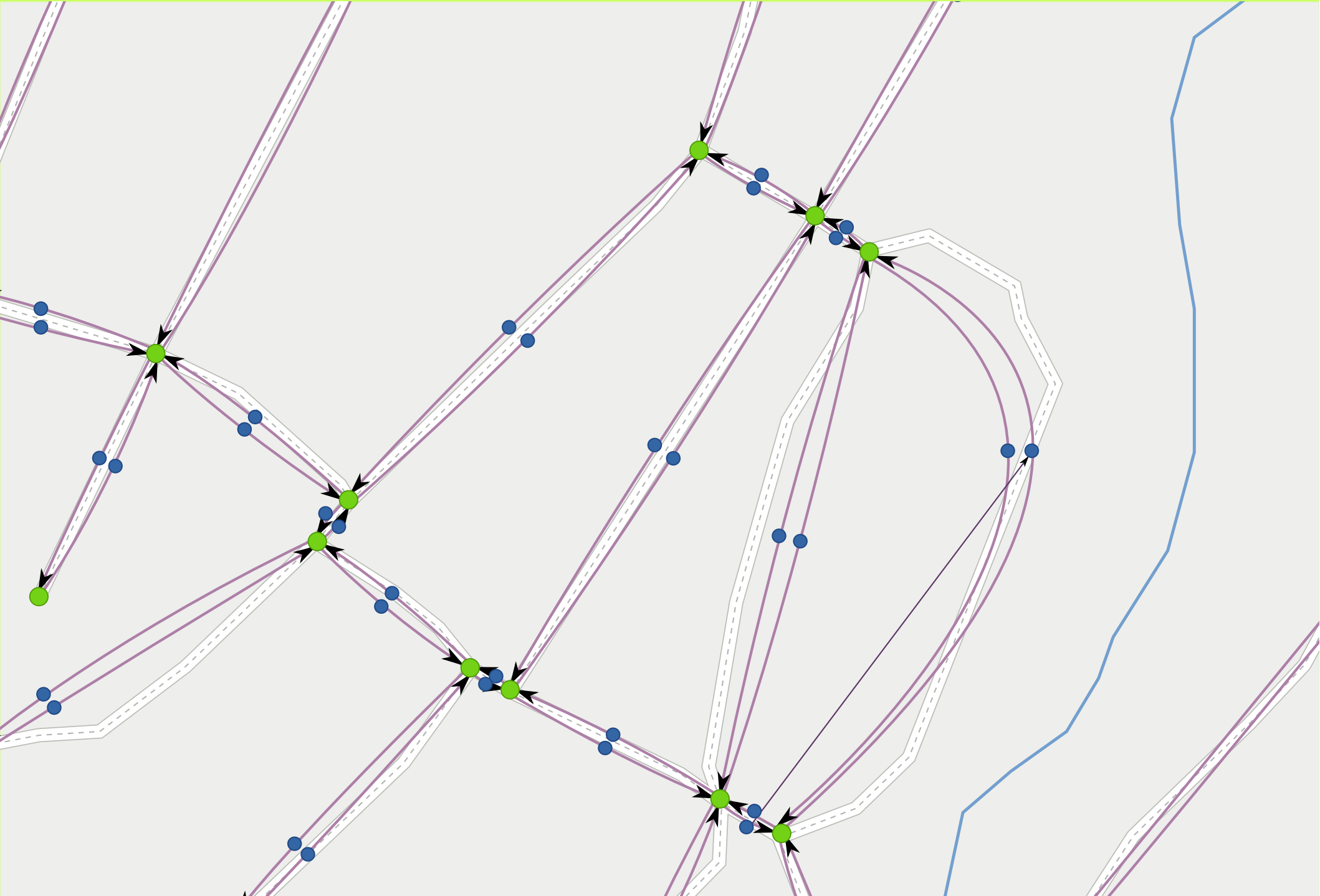
1. background > maps & graphs > line graph



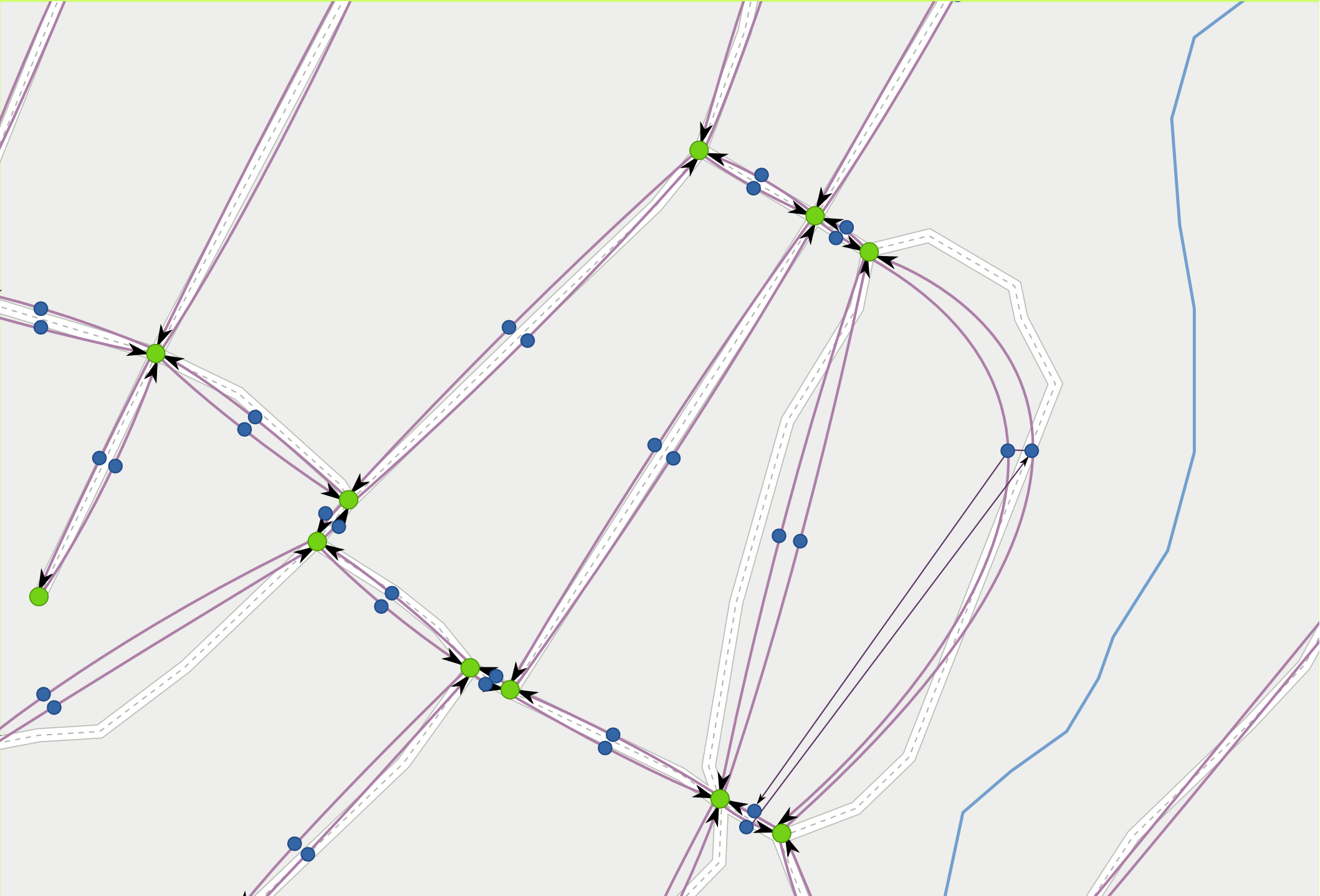
1. background > maps & graphs > line graph



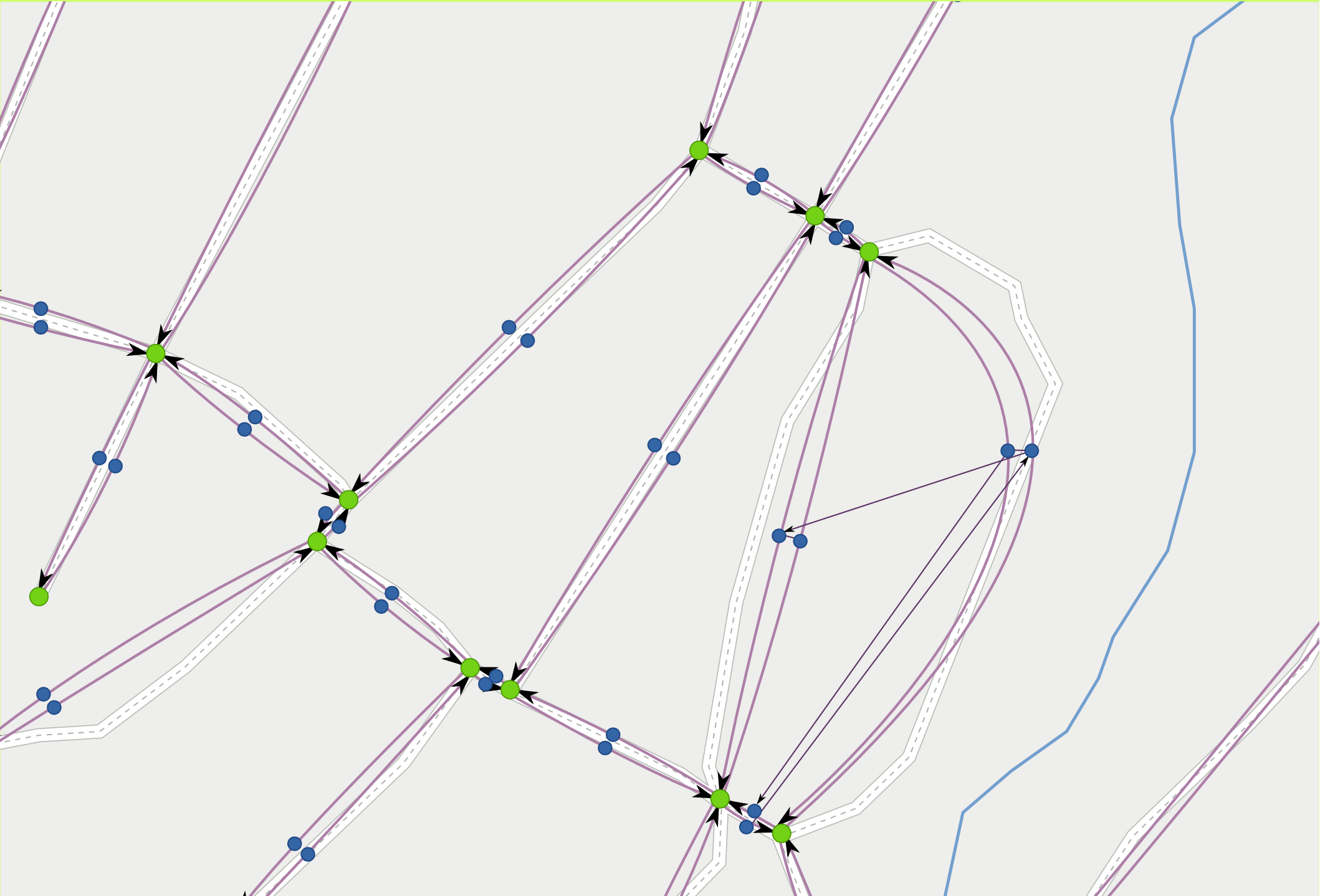
1. background > maps & graphs > line graph



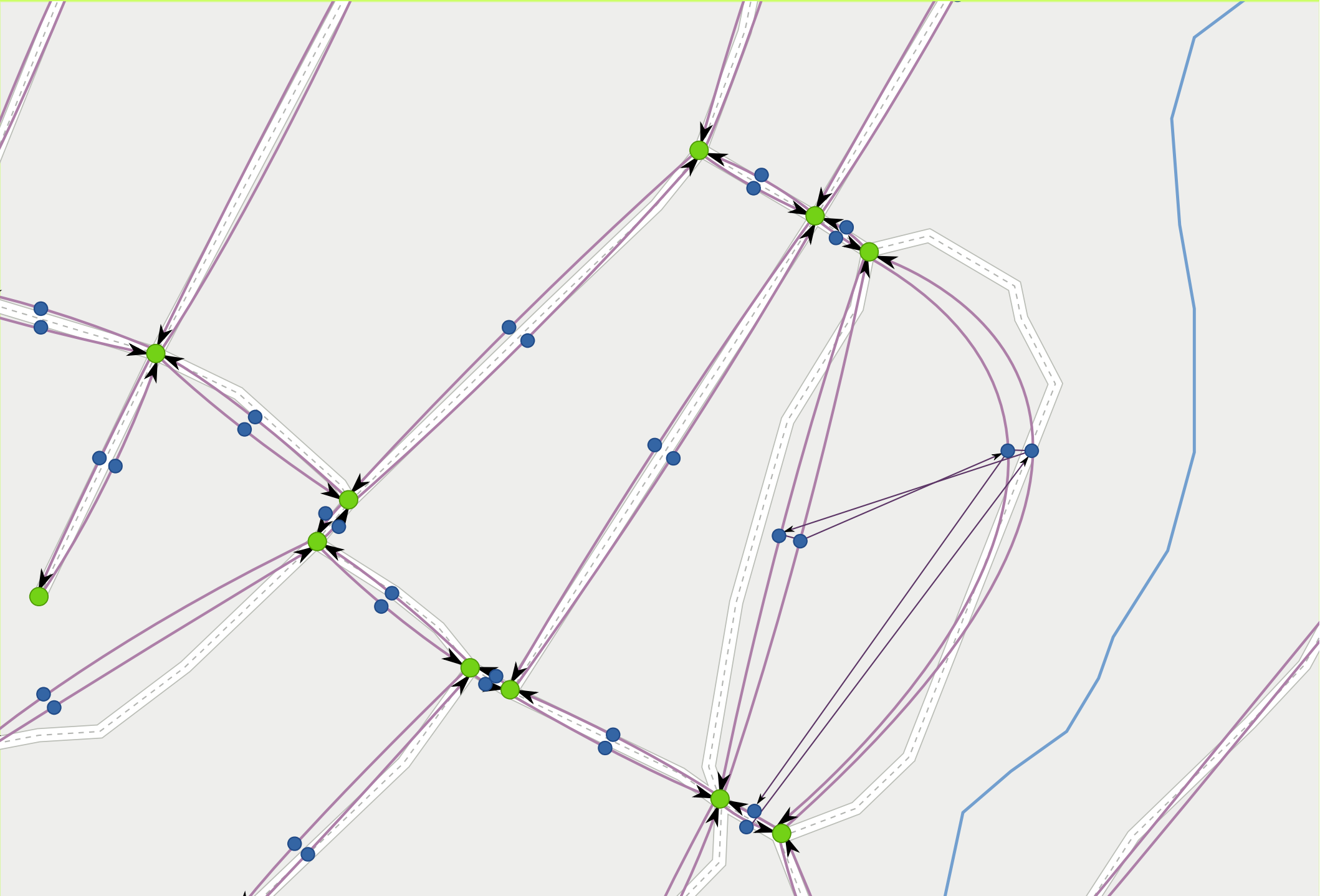
1. background > maps & graphs > line graph



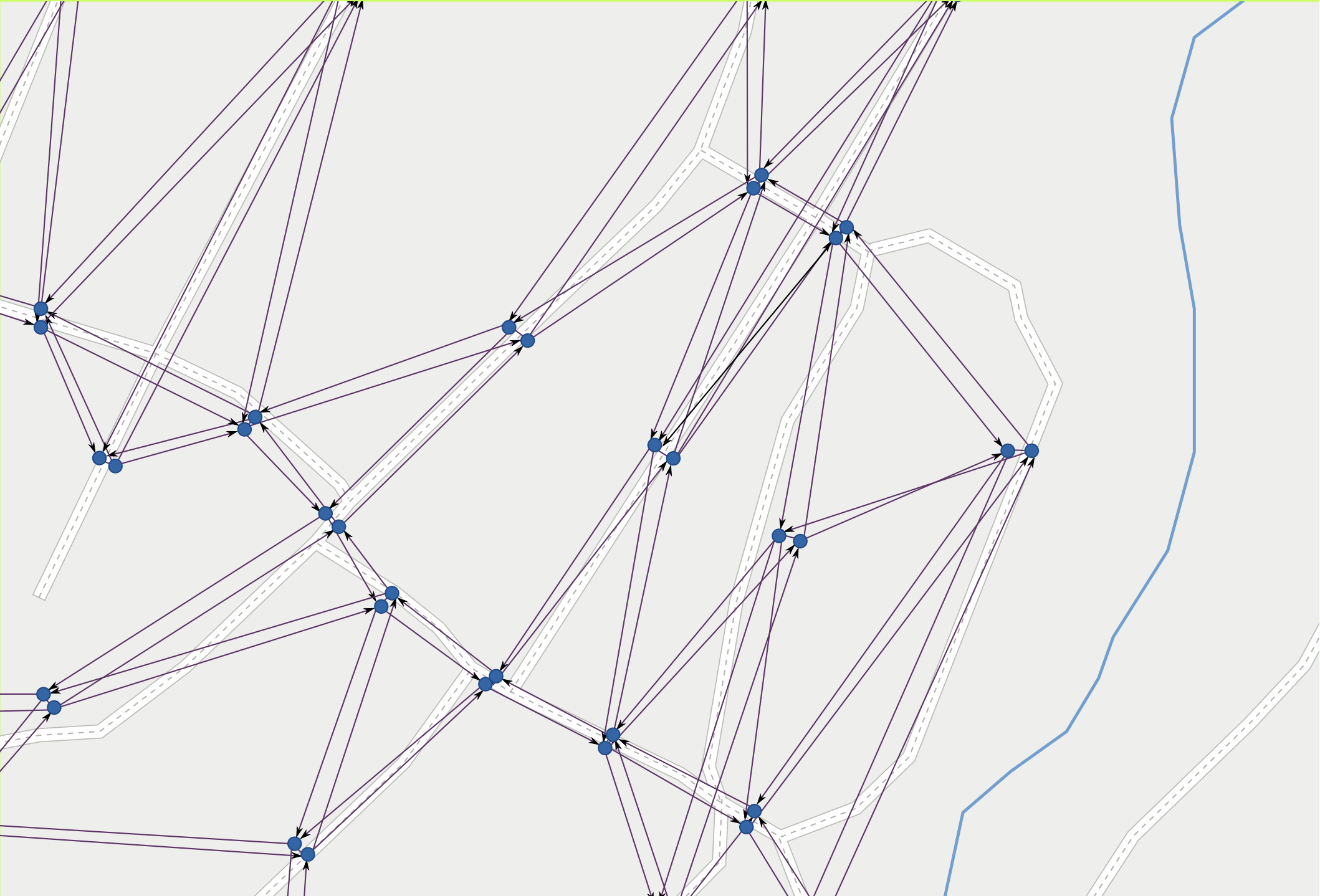
1. background > maps & graphs > line graph



1. background > maps & graphs > line graph



1. background > maps & graphs > line graph





2

problem

statement

2. problem statement

software *module*

2. problem statement

software *module*
exposing a **function**

2. problem statement

software *module*
exposing a function

data *returning* a
structure

2. problem statement

software *module*
exposing a function

data *returning* a
structure
for *routing*

2. problem statement

software *module*
exposing a function

data *returning* a
structure

for *routing*

in *soft* **real-time**

2. problem statement

software *module*
exposing a function

data *returning* a
structure

for *routing*

in *soft* real-time

2. problem statement

sequential operation:

2. problem statement

sequential operation:

load *map data*

2. problem statement

sequential operation:

load *map data*
build topology

2. problem statement

sequential operation:

load *map data*
build topology

apply **restrictions**

2. problem statement

sequential operation:

load *map data*
build topology

apply **restrictions**

build ***directed** graph*

2. problem statement

sequential operation:

load *map data*
build topology

apply **restrictions**

build *directed* graph

return **line graph**

2. problem statement

sequential operation:

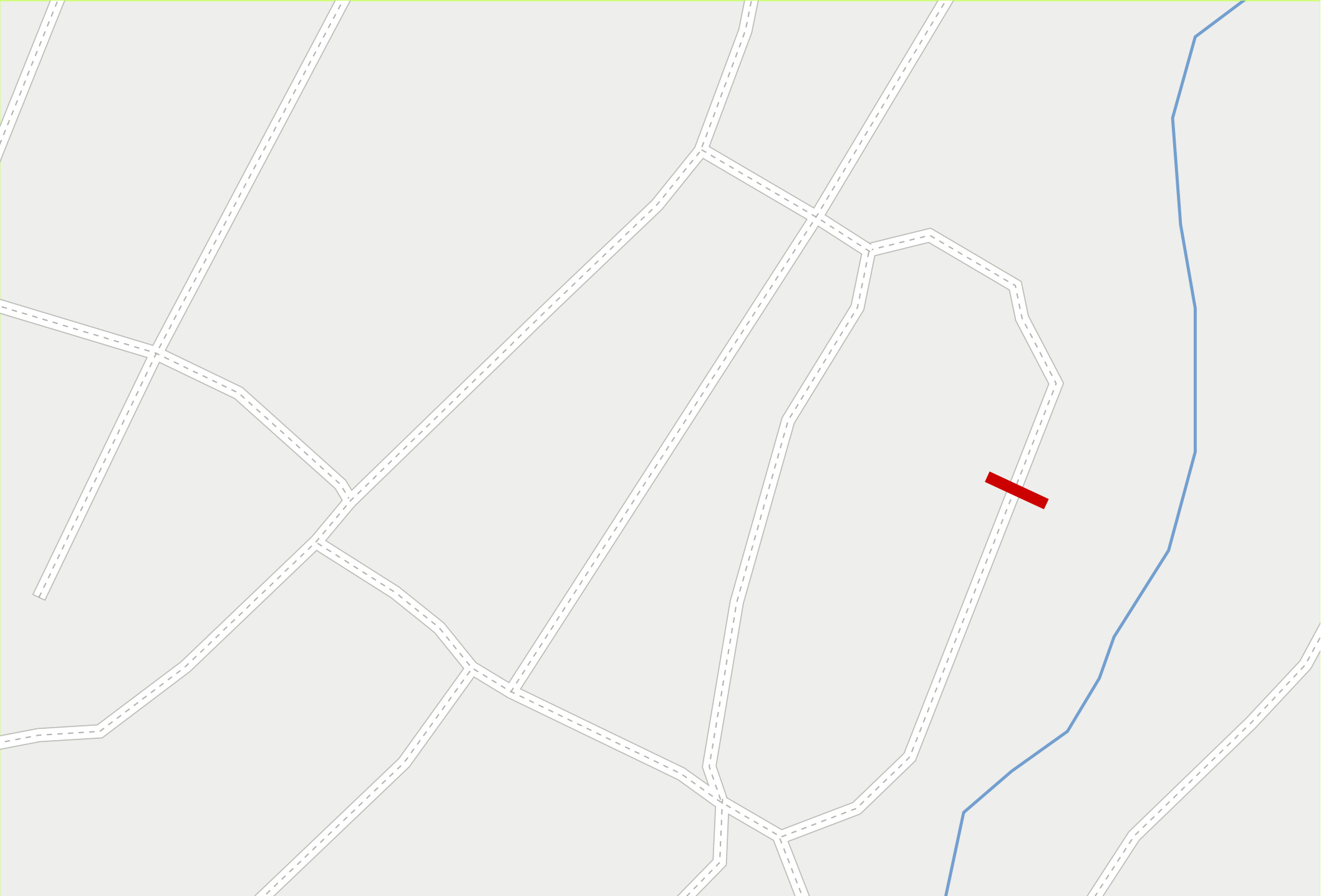
load *map data*
build topology

apply **restrictions**

build *directed* graph

return **line graph**

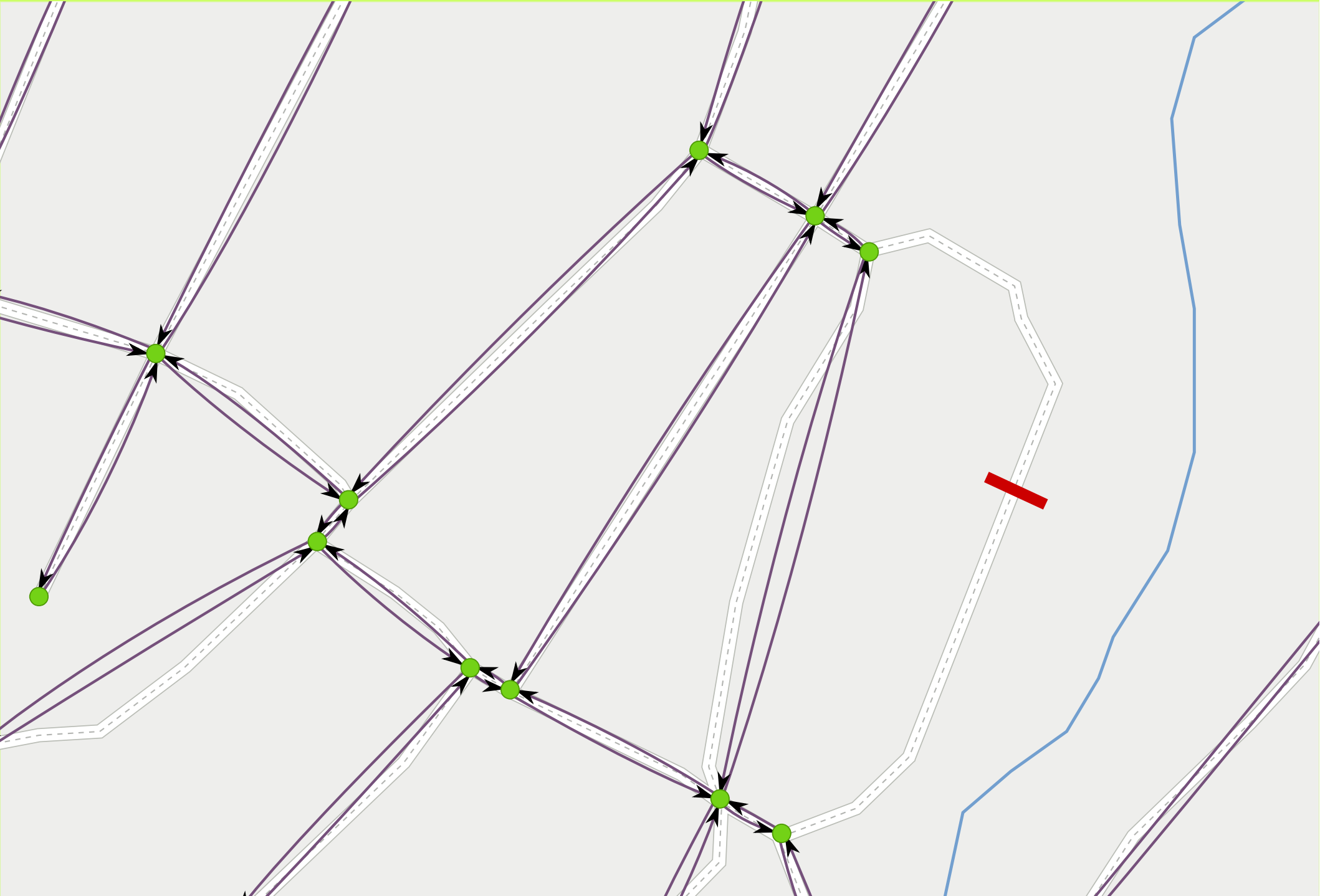
2. problem statement > applying restrictions > map



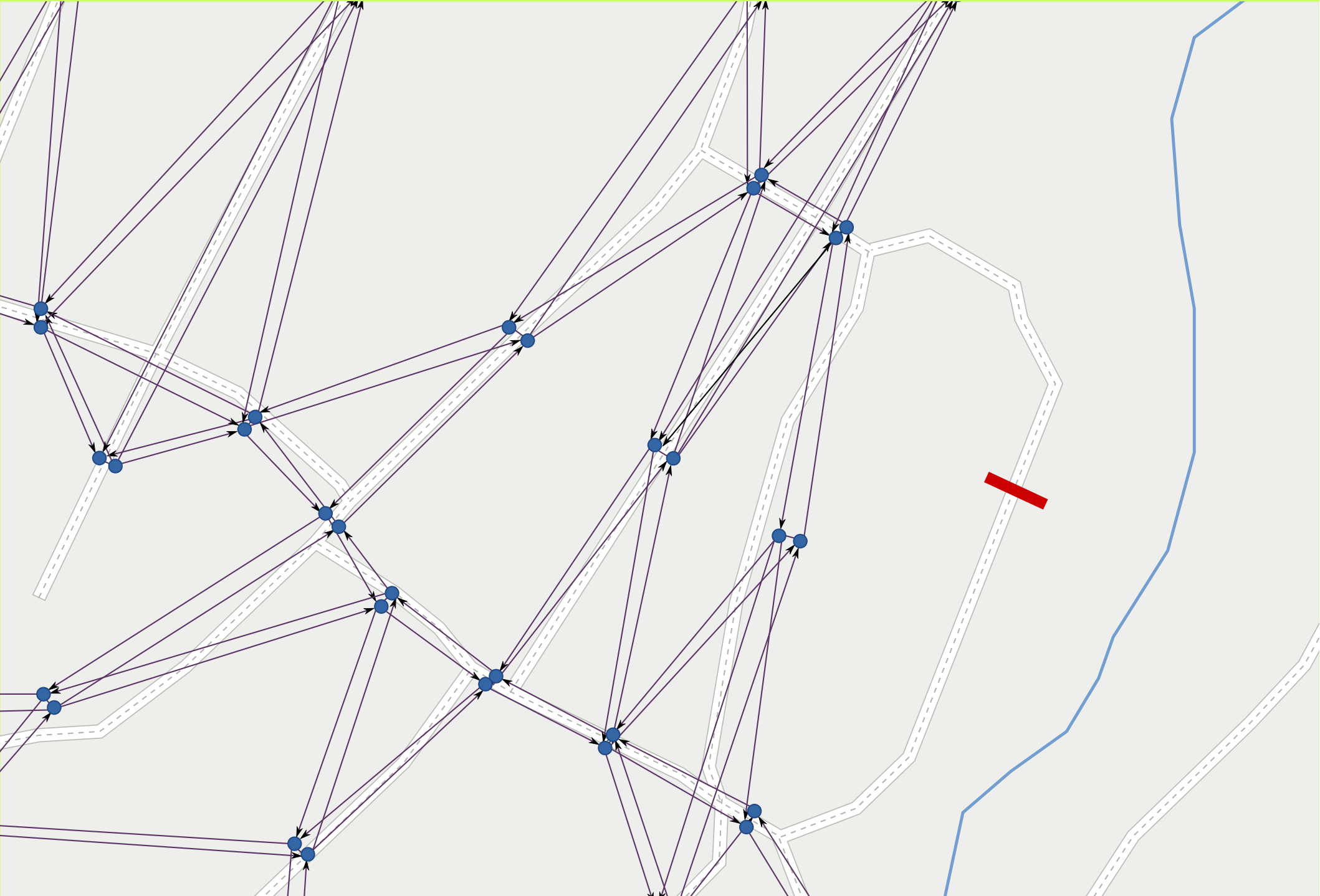
2. problem statement > applying restrictions > topology



2. problem statement > applying restrictions > directed graph



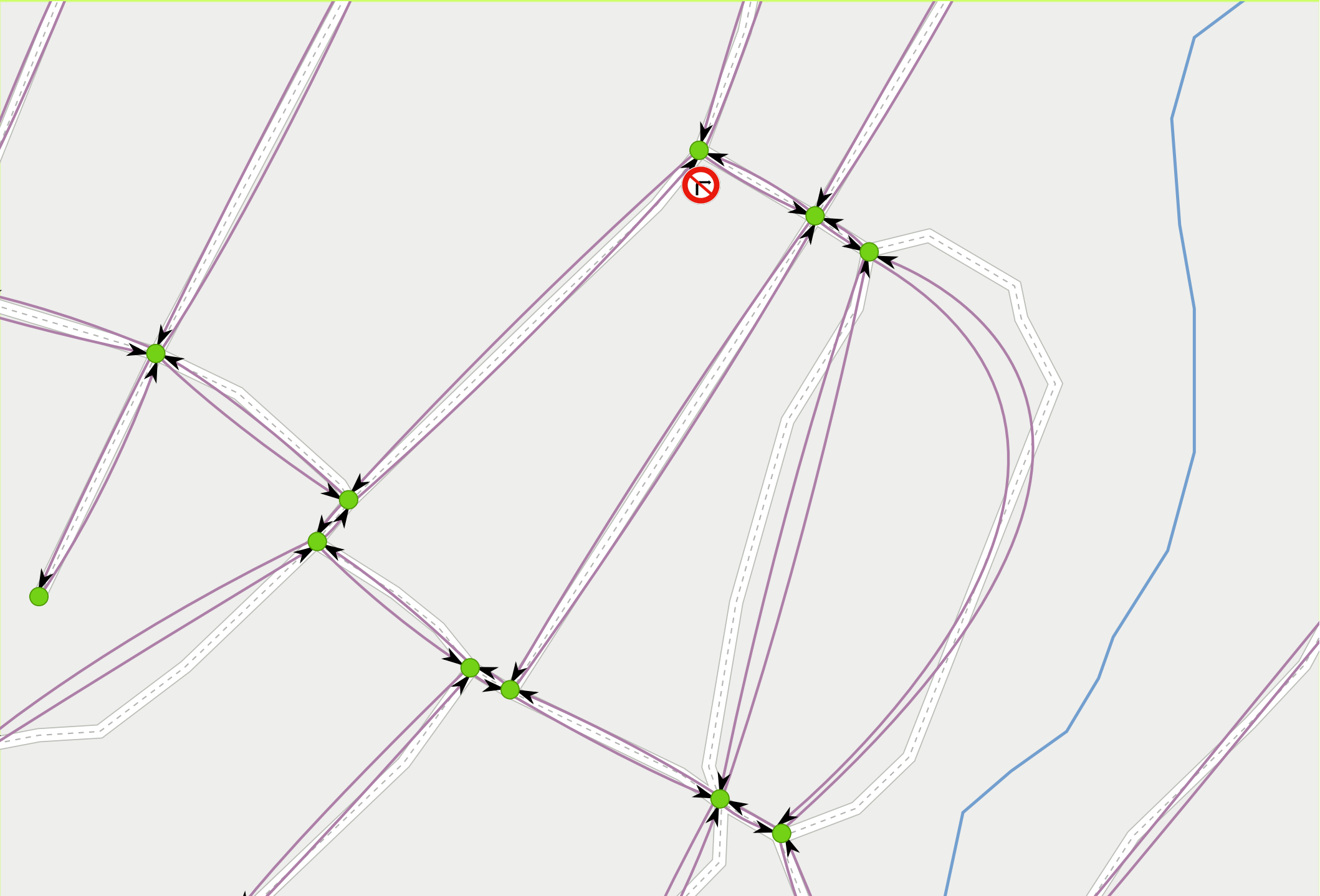
2. problem statement > applying restrictions > line graph



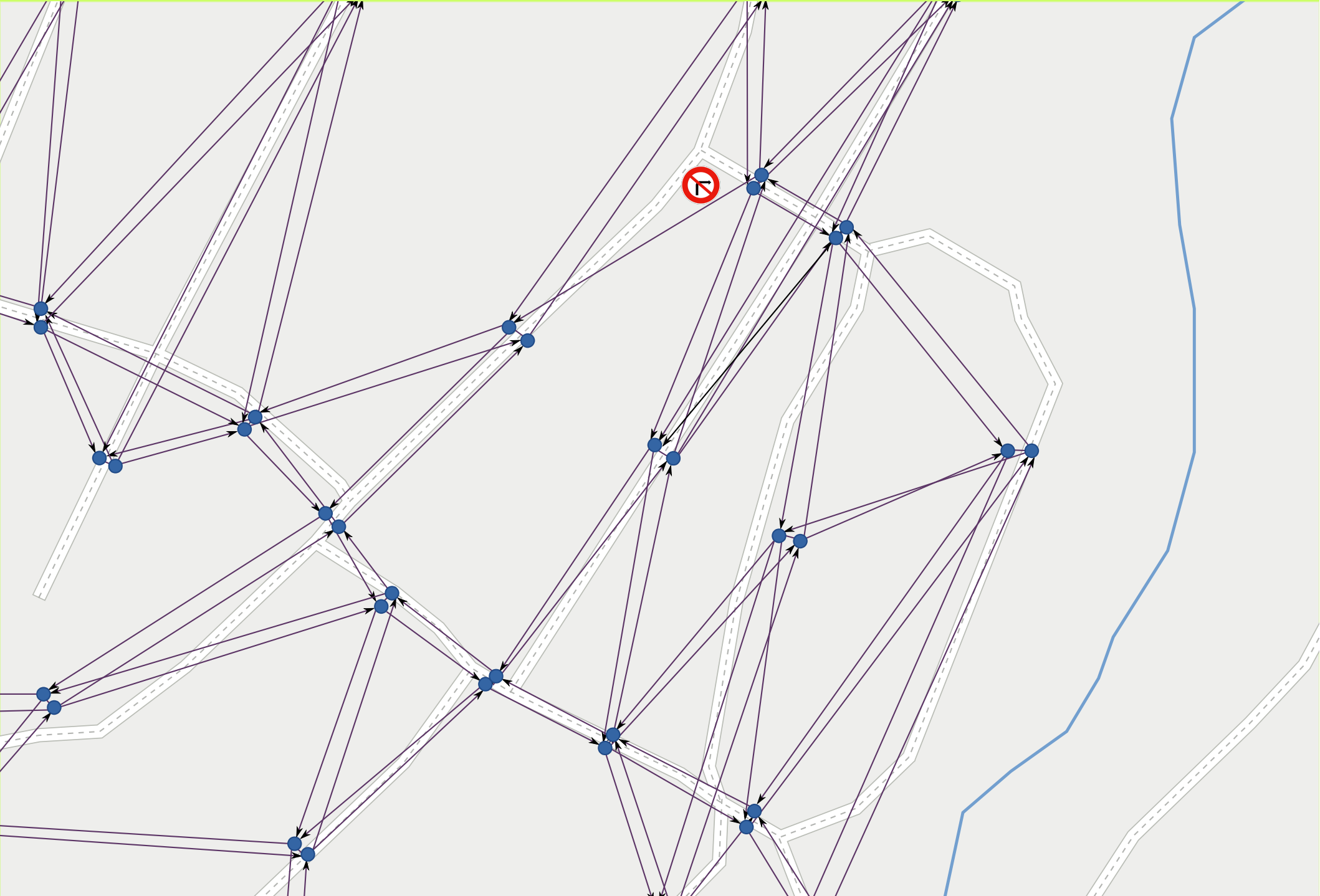
2. problem statement > applying restrictions > map



2. problem statement > applying restrictions > directed graph



2. problem statement > applying restrictions > line graph



2. problem statement

sequential operation:

load *map data*
build topology

preliminary

apply **restrictions**

build ***directed*** graph

return **line graph**

2. problem statement

sequential operation:

load *map data*
build topology

apply **restrictions**

build *directed* graph

return **line graph**

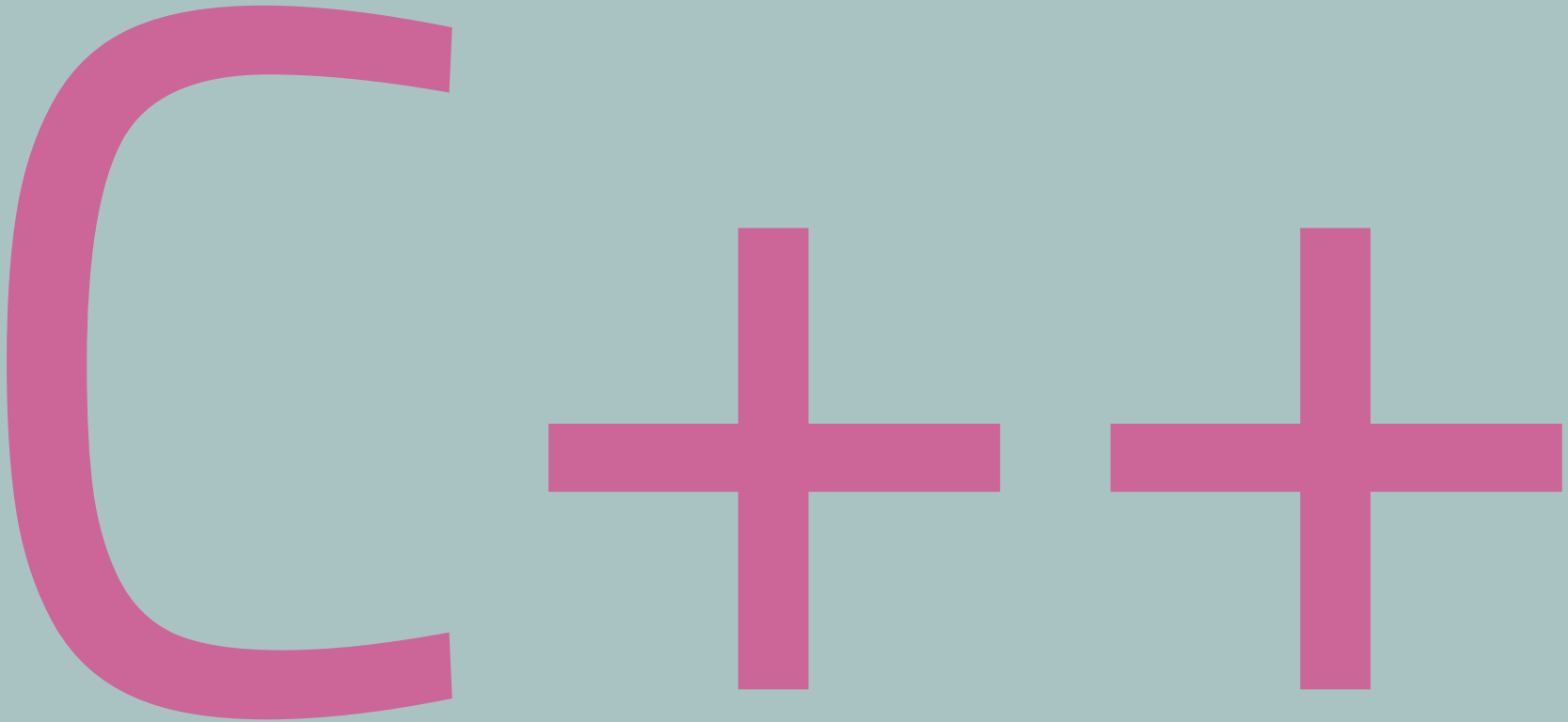
on demand

2. problem statement

required tools:

2. problem statement

required tools:



2. problem statement

required tools:

map *data*

2. problem statement

required tools:

map *data*

OpenStreetMap

2. problem statement

required tools:

map *data*

OpenStreetMap

PostGIS

2. problem statement

required tools:

graph *data structures*

2. problem statement

required tools:

graph *data structures*

Boost
graph library



3

method

requirement:

development

requirement:

behavior

driven development

(BDD)

requirement:

behavior

or

test

driven development

(BDD/TDD)

behavior (BDD) driven *development*

behavior (BDD) driven *development*

Scenario:

behavior (BDD) driven *development*

Scenario:

Given:

behavior (BDD) driven *development*

Scenario:

Given:

When:

behavior (BDD) driven *development*

Scenario:

Given:

When:

Then:

behavior (BDD) driven *development*

Scenario: Vectors can be sized and resized
Given:
When:
Then:

behavior (BDD) *driven development*

Scenario: Vectors can be sized and resized
Given: A vector with some items
When:
Then:

behavior (BDD) *driven development*

Scenario: Vectors can be sized and resized
 Given: A vector with some items
 When: The size is increased
 Then:

behavior (BDD) *driven development*

Scenario: Vectors can be sized and resized
 Given: A vector with some items
 When: The size is increased
 Then: The size and capacity change

tools

3. method > tools > BDD

Catch

```
#define CATCH_CONFIG_MAIN
#include "catch.hpp"
#include <vector>

SCENARIO ("Vectors can be sized and resized", "[vector]") {
    GIVEN ("A vector with some items") {
        std::vector<int> v(5);

        REQUIRE (v.size() == 5);
        REQUIRE (v.capacity() >= 5);

        WHEN ("The size is increased") {
            v.resize(10);

            THEN ("The size and capacity change") {
                REQUIRE (v.size() == 10);
                REQUIRE (v.capacity() >= 10);
            }
        }
    }
}
```

Boost Property Tree

```
#include <string>
#include <iostream>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>

void readJsonFile(const std::string& filename) {
    boost::property_tree::ptree pt;
    boost::property_tree::read_json(filename, pt);
    std::string host = pt.get<std::string>("host");
    int port = pt.get<int>("port");
    std::cout << "Host: " << host << ", port: " << port << std::endl;
}
```


3. method > tools > load map data

osm2pgsql

source:

OpenStreetMap



load:

```
$ osm2pgsql -U user -d map_db -s -k mapdata.osm
```



store:

PostGIS

3. method > tools > load map data

osm2pgsql + postgis_topology

source:

OpenStreetMap

load:

```
$ osm2pgsql -U user -d map_db -s -k mapdata.osm
```

store:

PostGIS

build topology:

```
$ psql -U user -d map_db  
-c "SELECT topology.CreateTopology('roads_topo', 900913);"
```

3. method > tools > work with DB

libpqxx

```
#include <pqxx/pqxx>
//...
pqxx::connection conn(
    "dbname=testdb"
    "user=tester"
    "password=tester"
    "hostaddr=127.0.0.1"
    "port=5432");
```

link:

```
$ g++ mytest.cpp -lpqxx -lpq -o mytest
```

Boost Graph Library

property lists

```
typedef boost::adjacency_list<
    boost::listS, boost::vecS, boost::bidirectionalS,

    // Vertex properties
    boost::property< boost::vertex_name_t, std::string,
    boost::property< population_t, int,
    boost::property< zipcodes_t, std::vector<int> > > >,

    // Edge properties
    boost::property< boost::edge_name_t, std::string,
    boost::property< boost::edge_weight_t, double,
    boost::property< edge_speed_limit_t, int,
    boost::property< edge_lanes_t, int,
    boost::property< edge_divided, bool> > > > > >
    Map;
```

Boost Graph Library

bundled properties

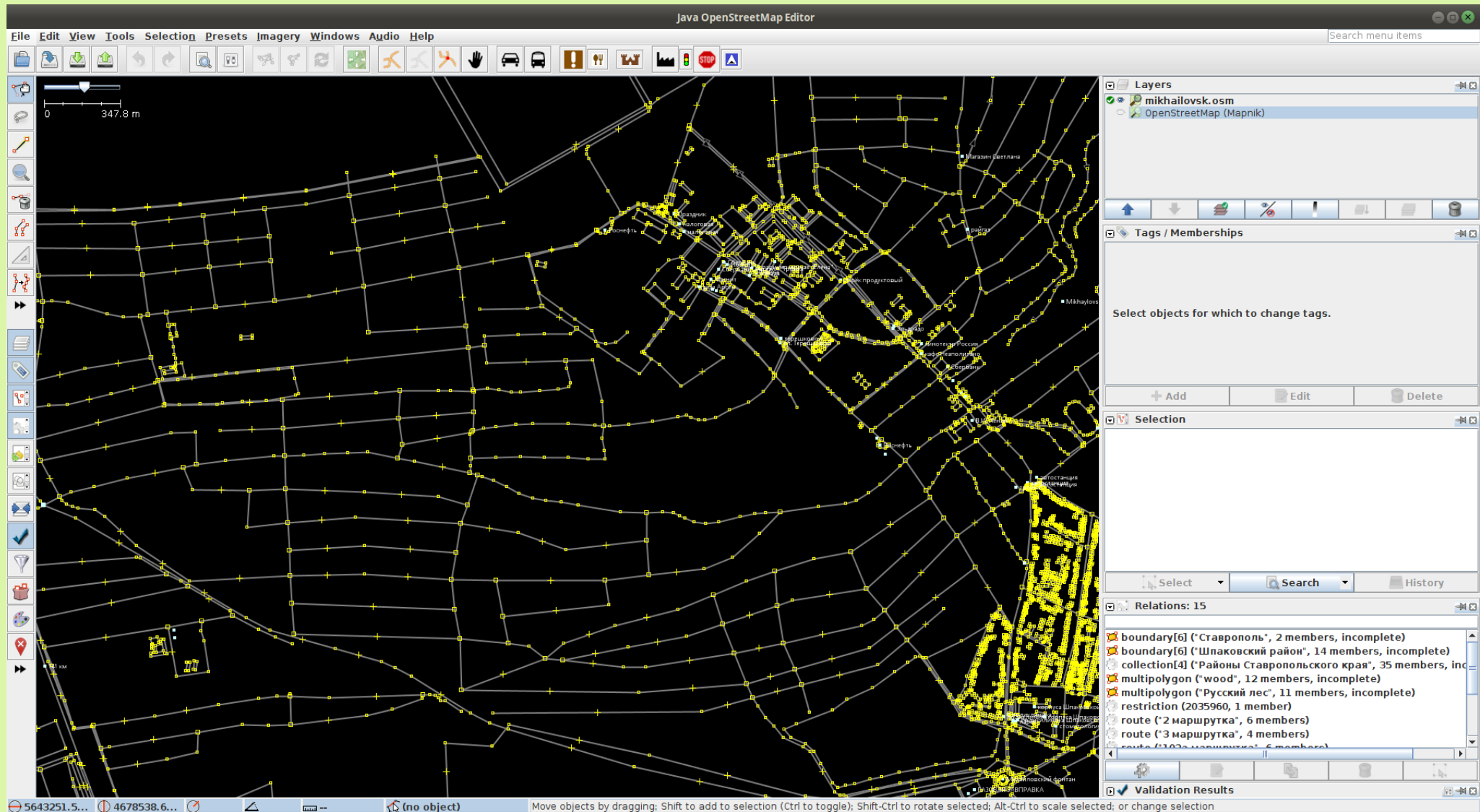
```
struct City {
    string      name;
    int         population;
    vector<int> zipcodes;
};

struct Highway {
    string name;
    double miles;
    int    speed_limit;
    int    lanes;
    bool   divided;
};

typedef boost::adjacency_list<
    boost::listS, boost::vecS, boost::bidirectionalS,
    City, Highway>
    Map;
```

3. method > tools > edit map data

JOSM

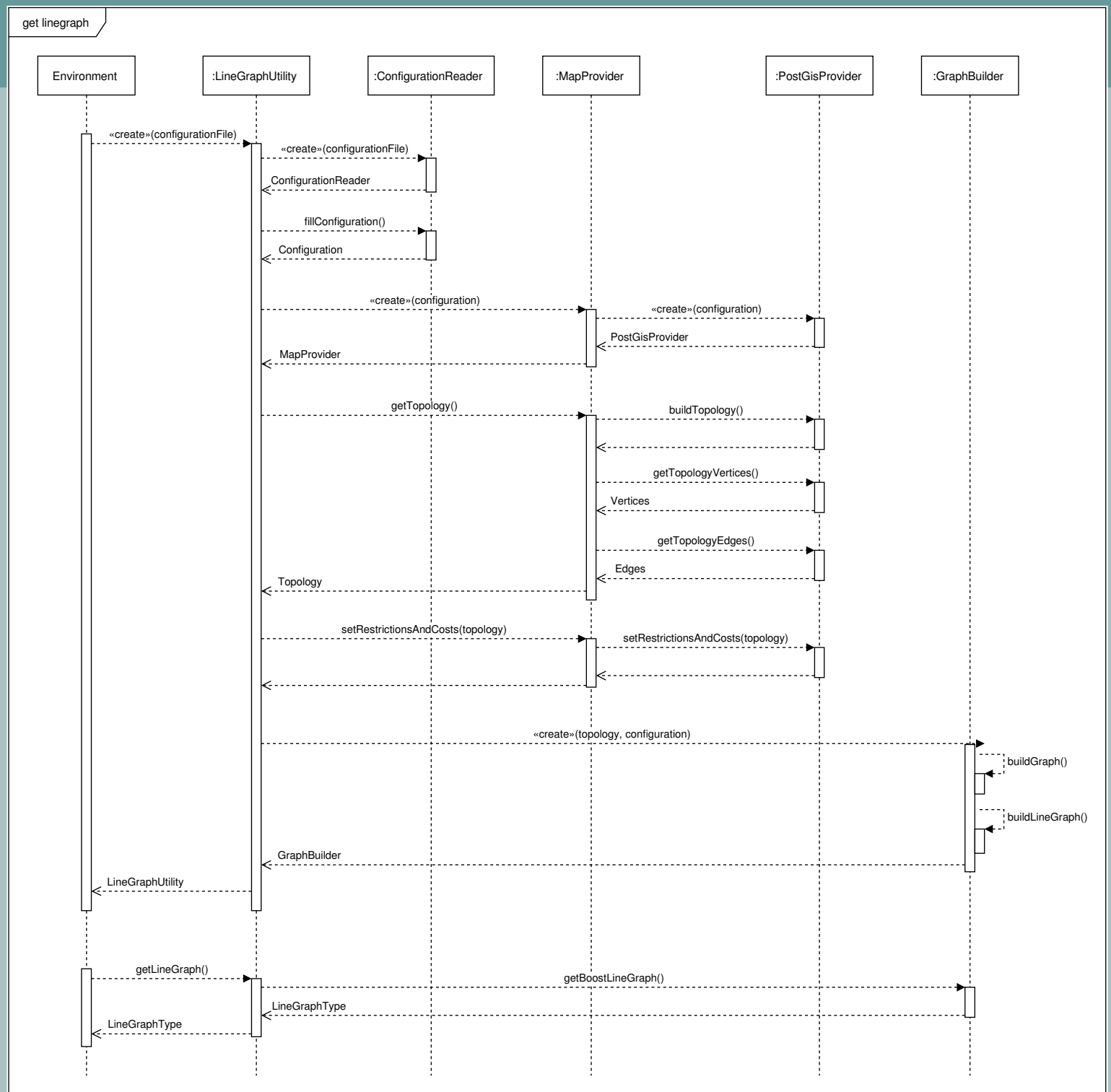




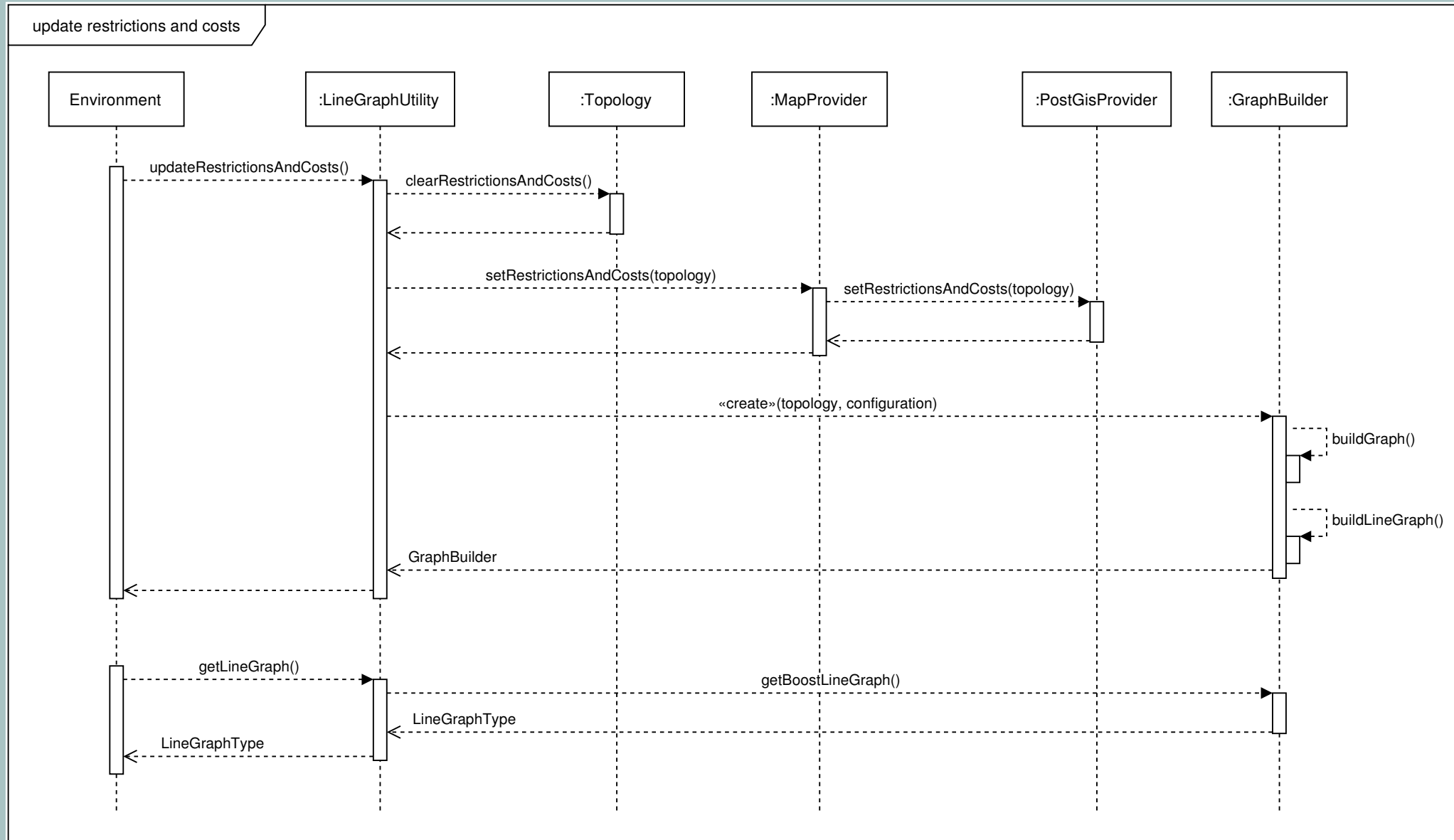
4

results

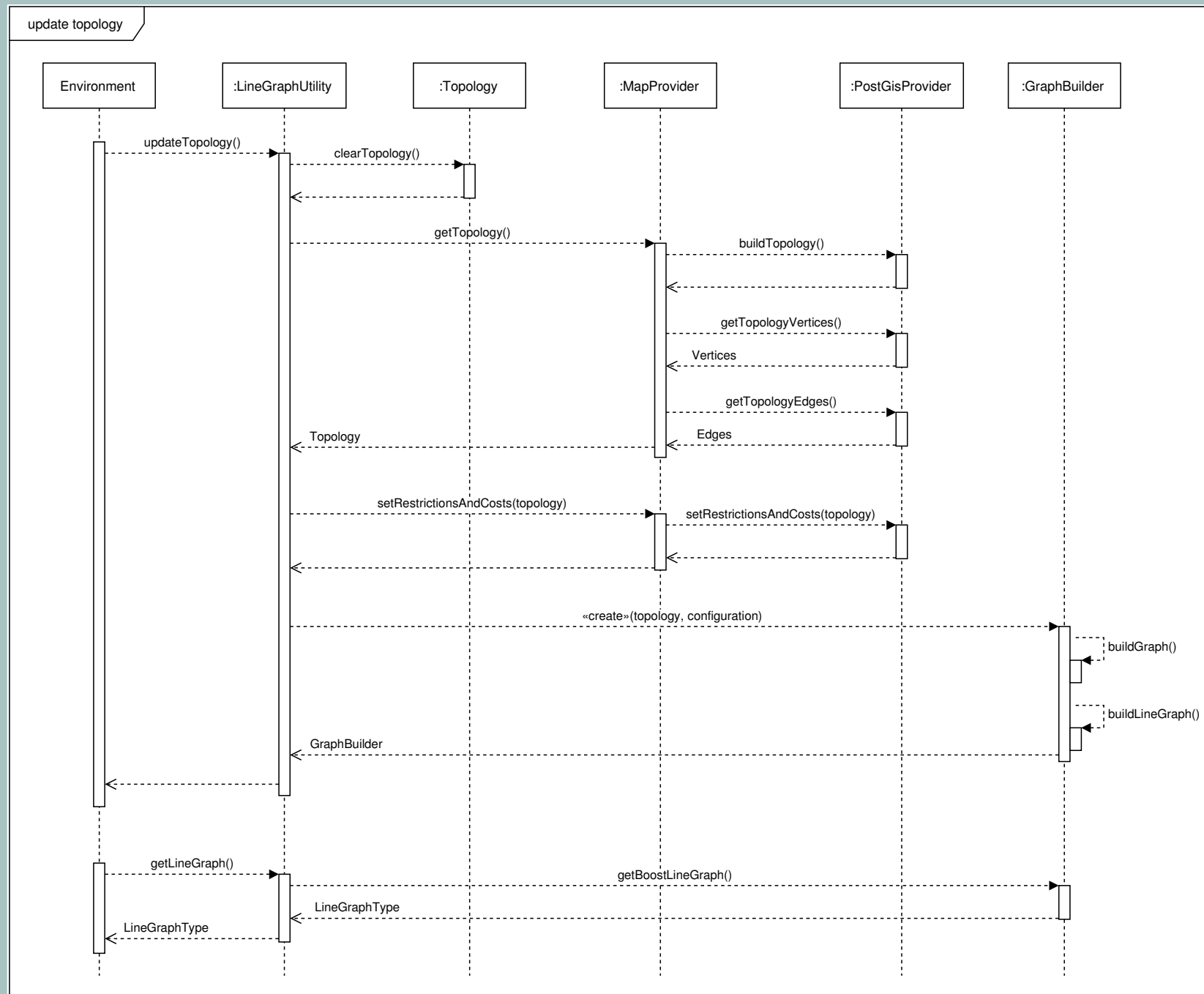
4. results > design



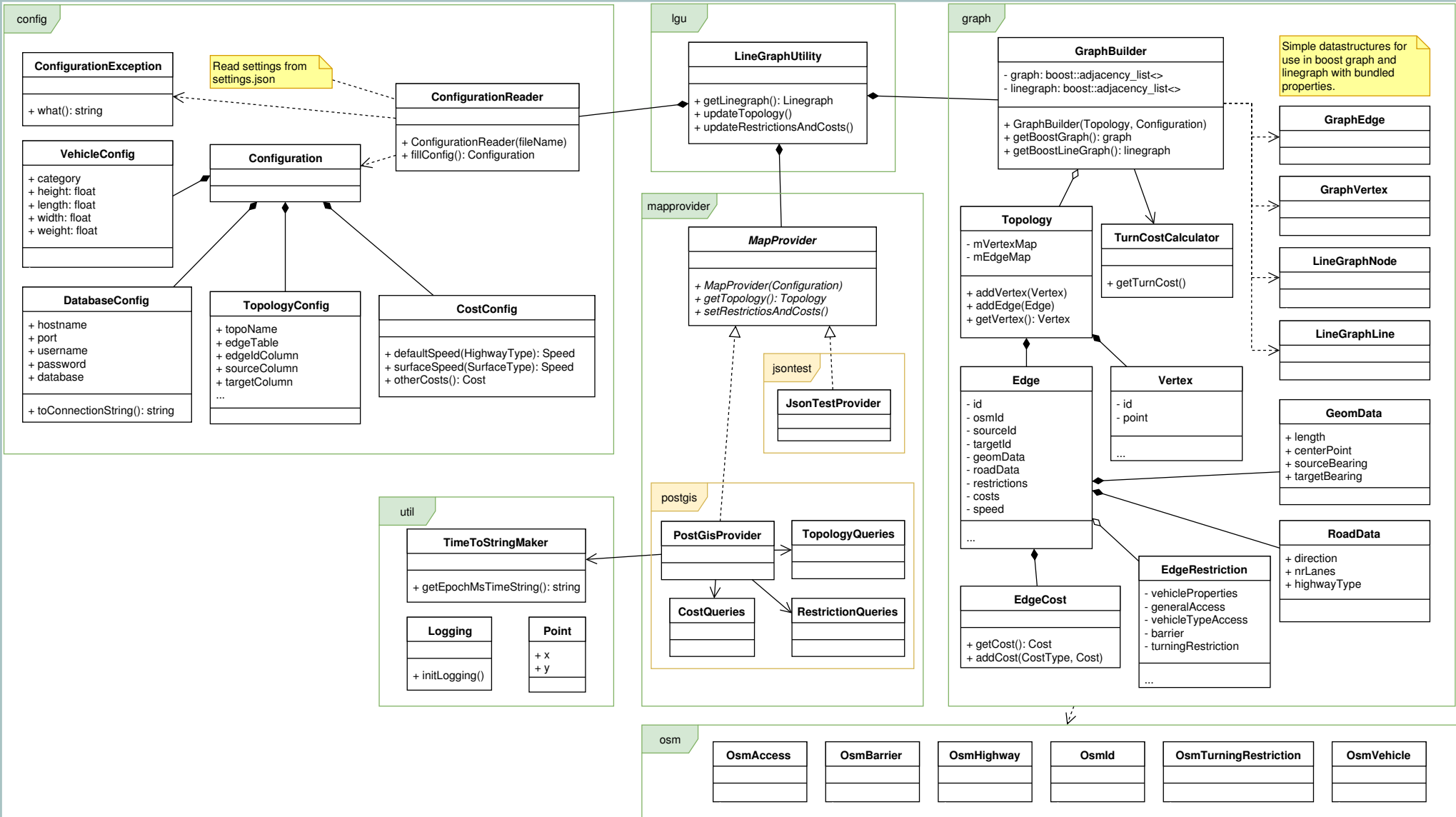
4. results > design



4. results > design



4. results > design



tests

specification fulfillment

tests

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

tests

4. results

Photo (cropped): Achadwick. ©CC-SA 2.0
http://wiki.openstreetmap.org/wiki/File:UK_motor_restriction_sign_with_exceptions.jpg



conditional
restrictions

motor_vehicle=no

motor_vehicle:conditional=yes @ (18:30-07:30)

psv=yes

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

time of day, inclination ...

tests

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

time of day, inclination ...

tests

performance

4. results

test graph sizes

	Graph		Line graph	
	vertices	edges	nodes	lines
Mikhailovsk	654	1618	1618	4758
Partille	1645	2265	2265	5577

performance

4. results

test graph sizes

	Graph		Line graph	
	vertices	edges	nodes	lines
Mikhailovsk	654	1618	1618	4758
Partille	1645	2265	2265	5577

time to get line graph
average of 100 rounds

Topology		get LineGraph (s)
Mikhailovsk	pre-built	0.143171
	on demand	4.936007
Partille	pre-built	0.182152
	on demand	10.557756

performance

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

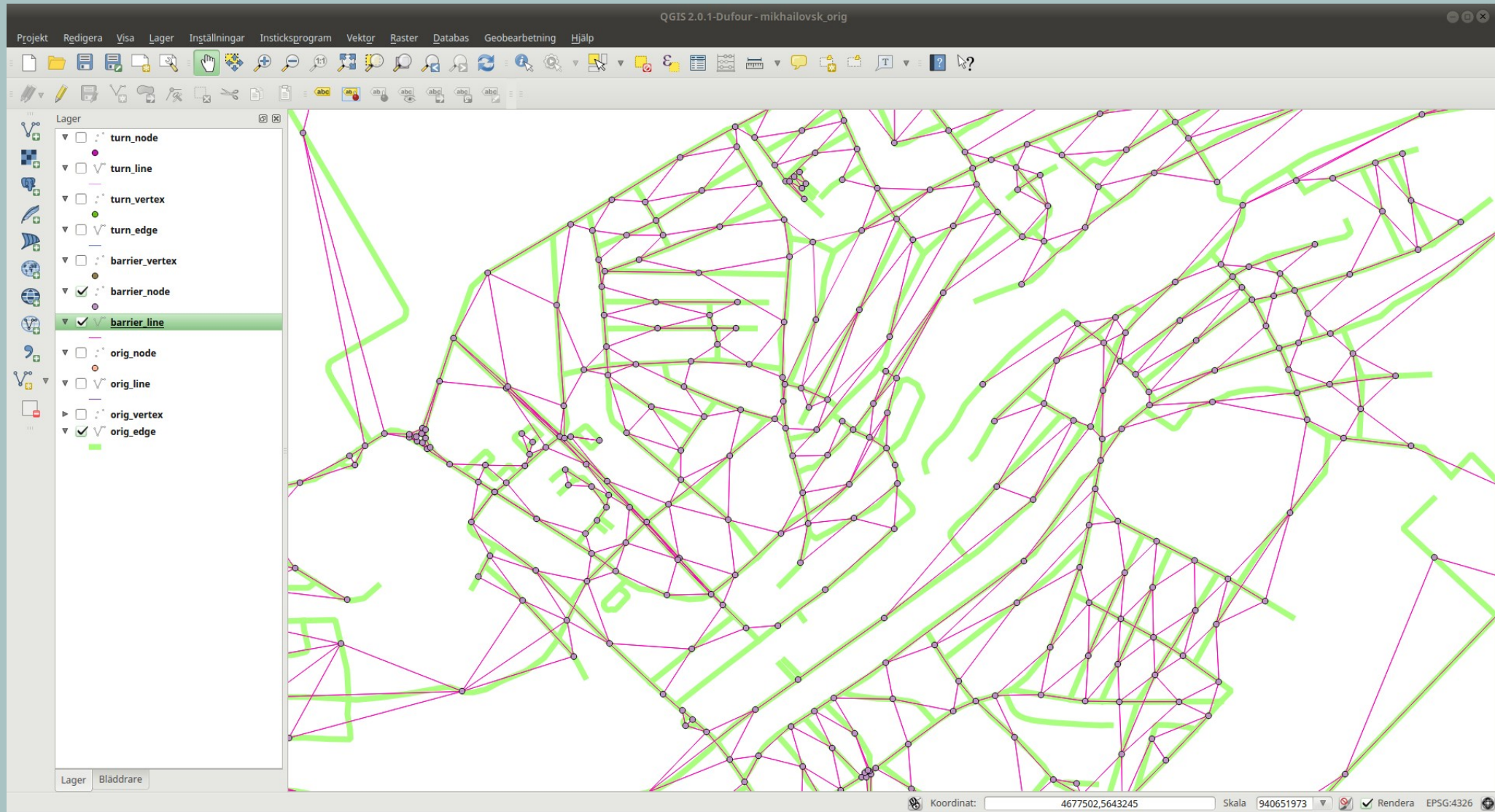
time of day, inclination ...

tests

performance

visual examination

4. results



visual examination

QGIS

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

time of day, inclination ...

tests

performance

visual examination



5

conclusions

working

5. conclusions

working
but
incomplete

5. conclusions

working
but
incomplete
due to *time*

5. conclusions

working
but
incomplete

due to *time*

late *specification*

5. conclusions

working
but
incomplete

due to *time*

late *specification*

complex *restrictions*



6

future
work

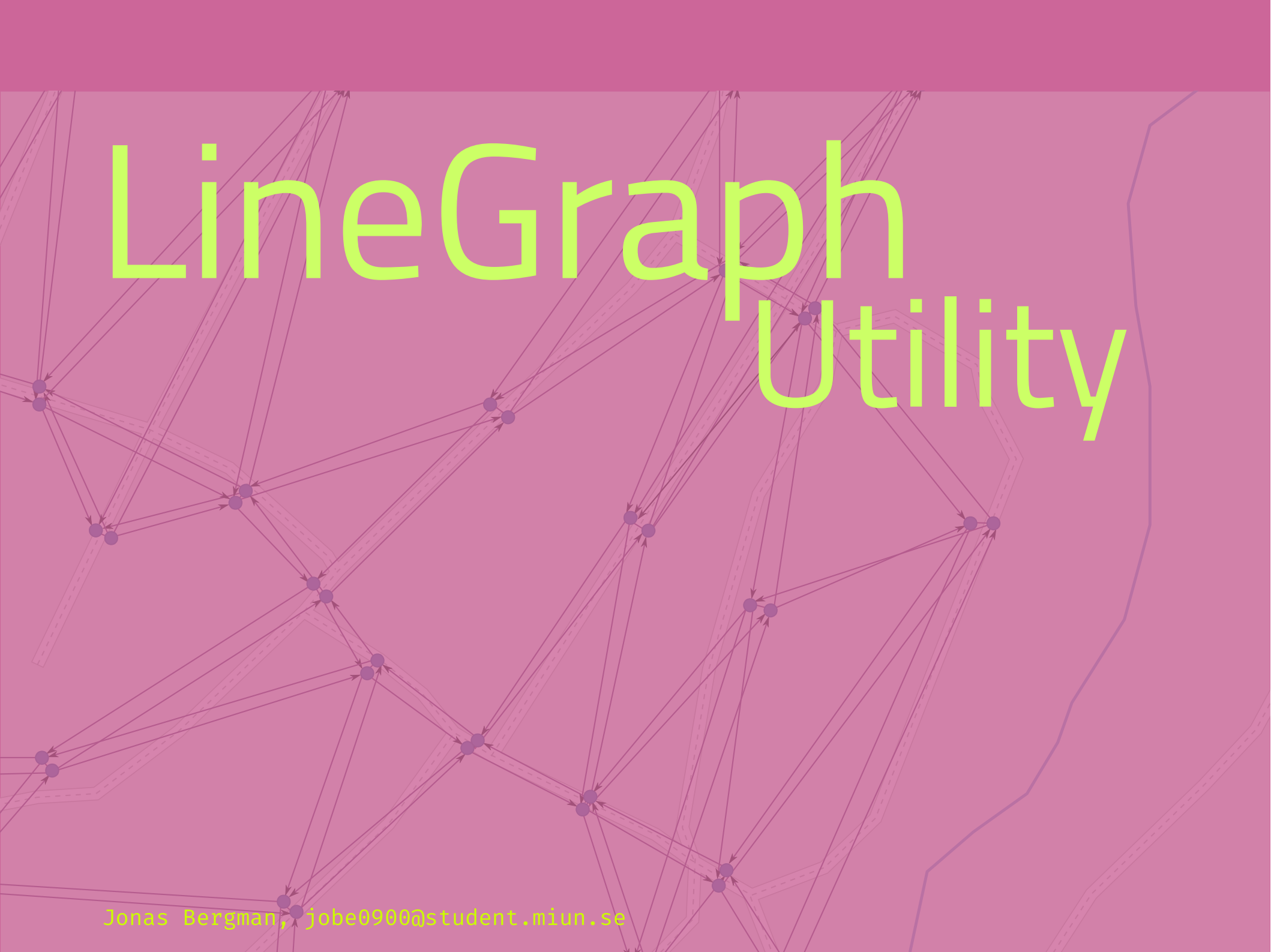
re-model
restrictions

6. future work

re-model restrictions

```
<restriction-type>[:<transportation mode>][:<direction>]:conditional  
  = <restriction-value> @ <condition>[;<restriction-value> @ <condition>]
```

LineGraph Utility

The background of the slide features a light purple map of a road network. Overlaid on this map is a complex graph structure. The graph consists of numerous small, dark purple circular nodes. These nodes are interconnected by a dense web of thin, dark purple lines representing edges. Some of these edges are directed, as indicated by small arrowheads. The graph appears to be a utility or network layer, possibly representing a transit system or a data network, that is being mapped onto a geographical area.