LineGraph Utility

Jonas Bergman, jobe0900@student.miun.se

LineGraph Utility

a software module

Jonas Bergman, jobe0900@student.miun.se

LineGraph Utility *for routing*

a software module

Jonas Bergman, jobe0900@student.miun.se

LineGraph Utility

a software module *for* *routing* of public transportation

Jonas Bergman, jobe0900@student.miun.se

1 background

managing flexible *public transportation*

flexible public transportation:

no *timetables*

flexible public transportation:

- no timetables

commission

rides

flexible public transportation:

- no timetables
- commission rides

calculate *routes*

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?

flexible public transportation: **gain**?

less

idle vehicles

flexible public transportation: **gain**?

+ environment

less

idle vehicles

flexible public transportation: **gain**?

+ economy
+ environment

less

idle vehicles

flexible public transportation: **gain**?

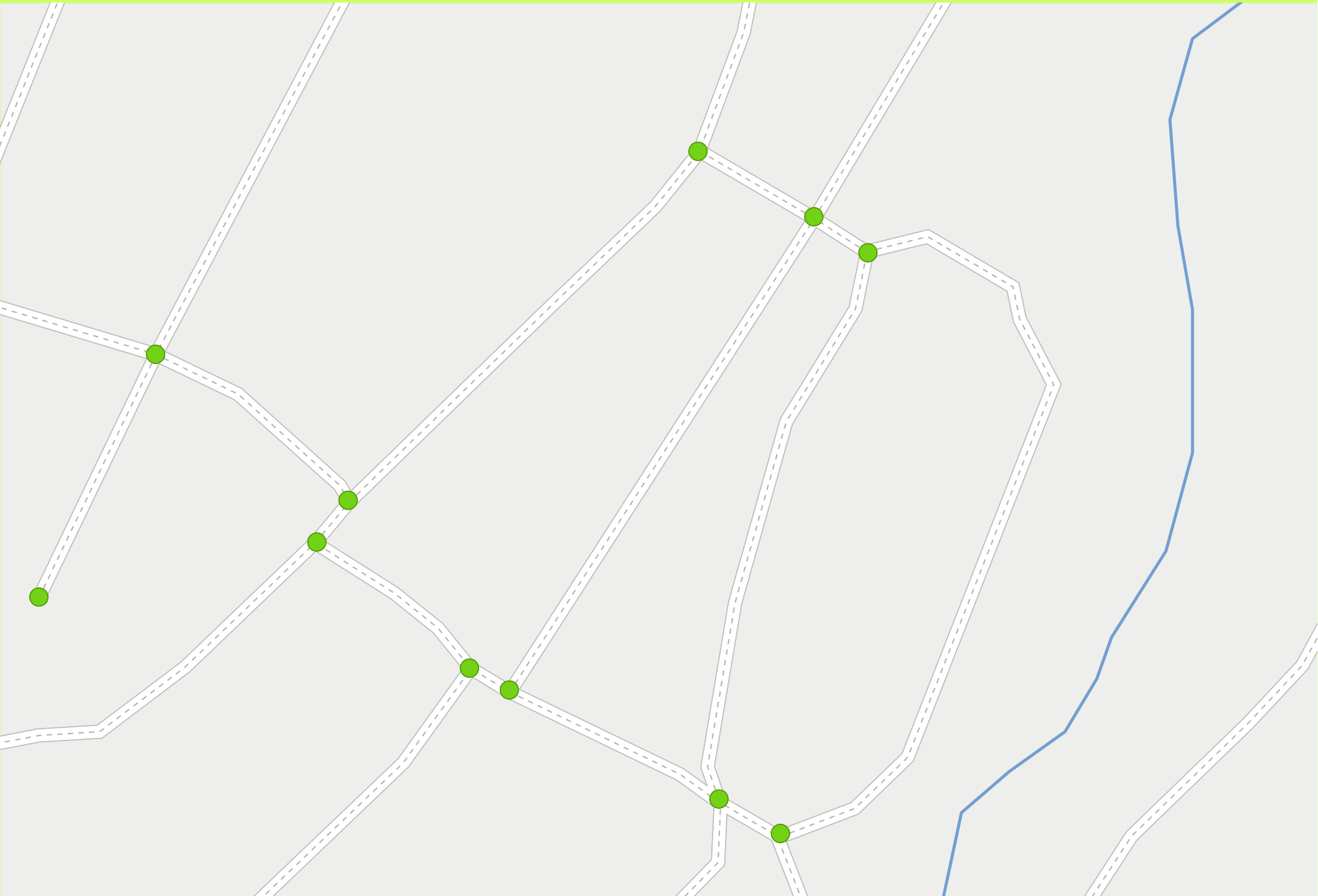
- less idle vehicles

less

waiting

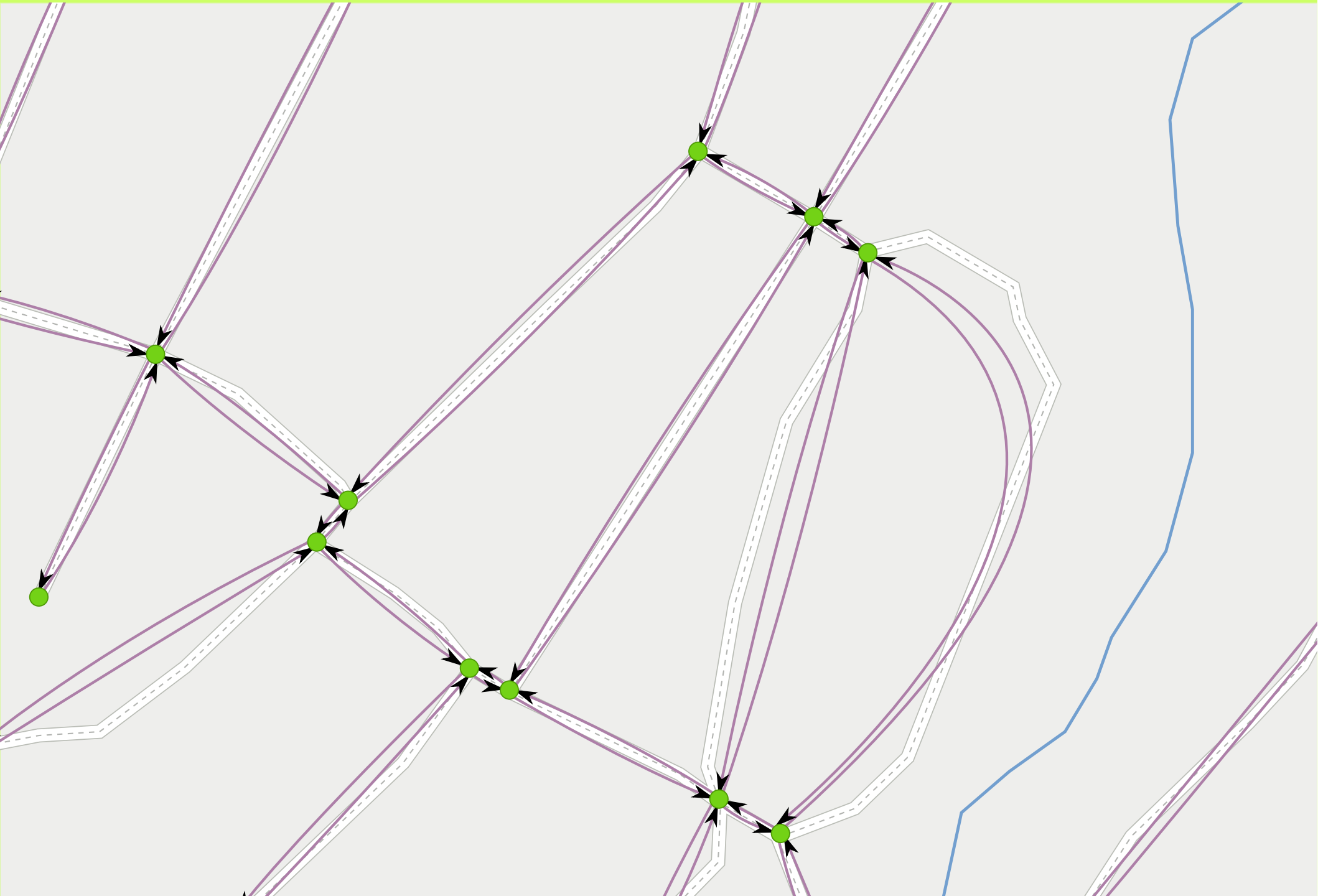
1. background > maps & graphs

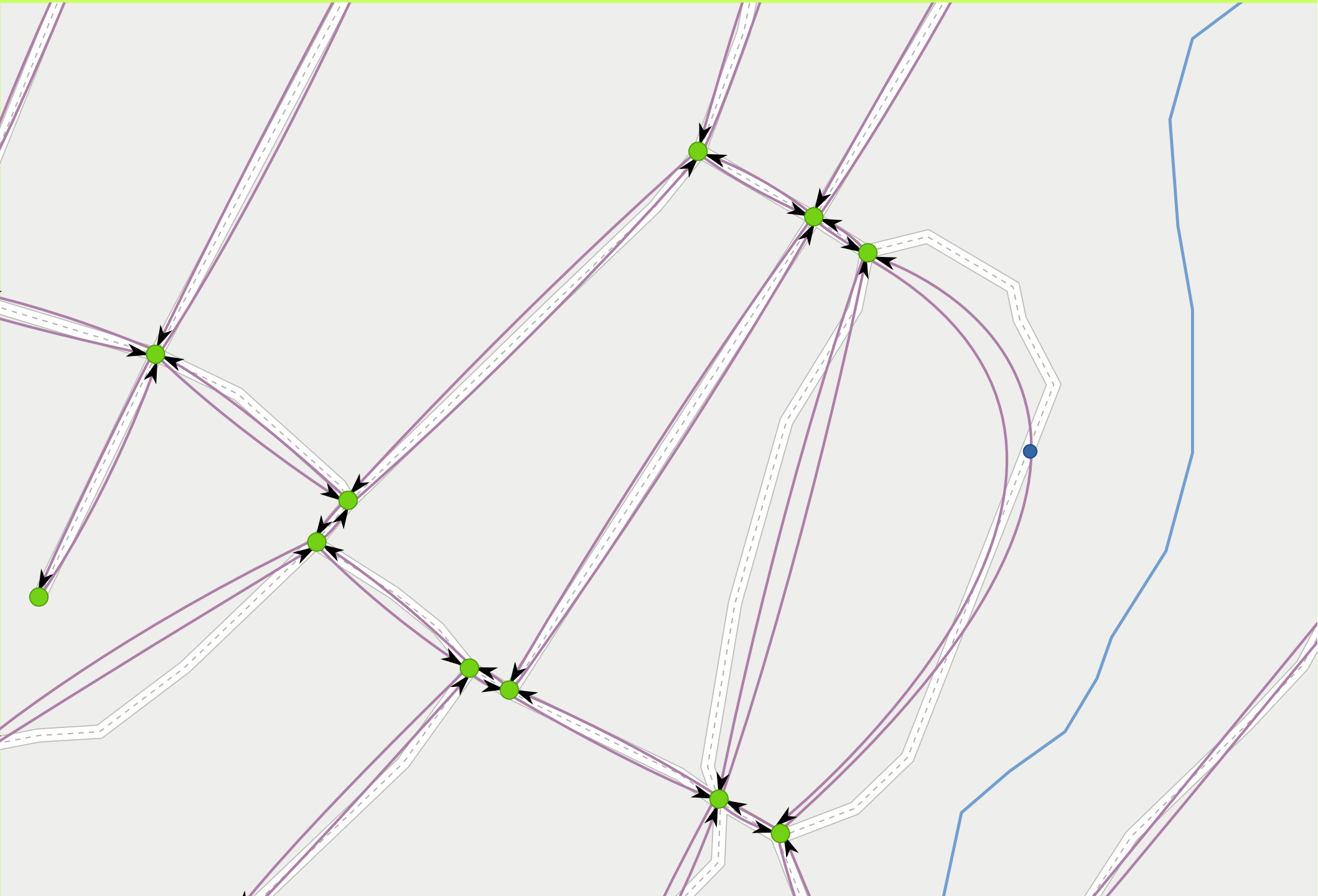
1. background > maps & graphs

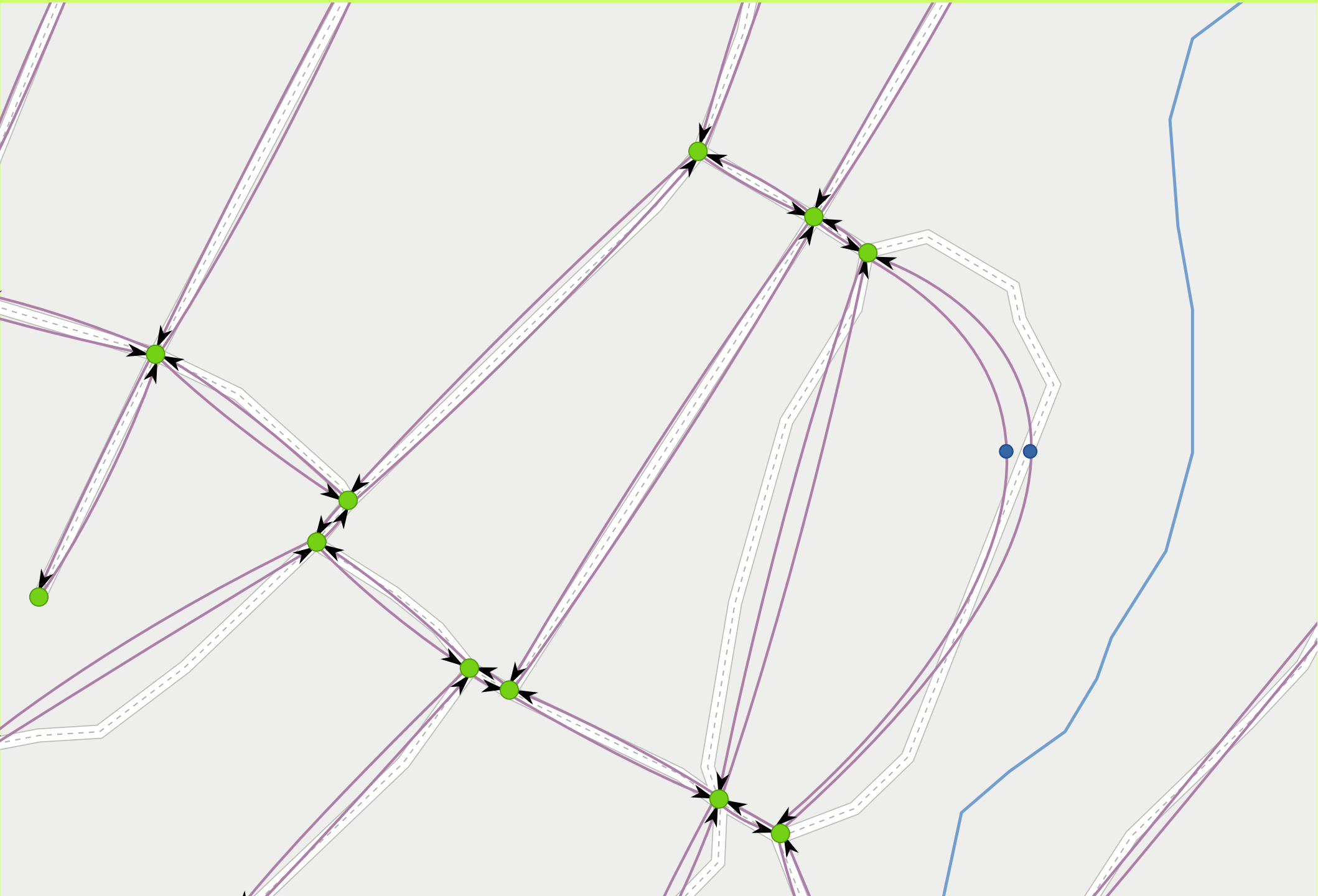


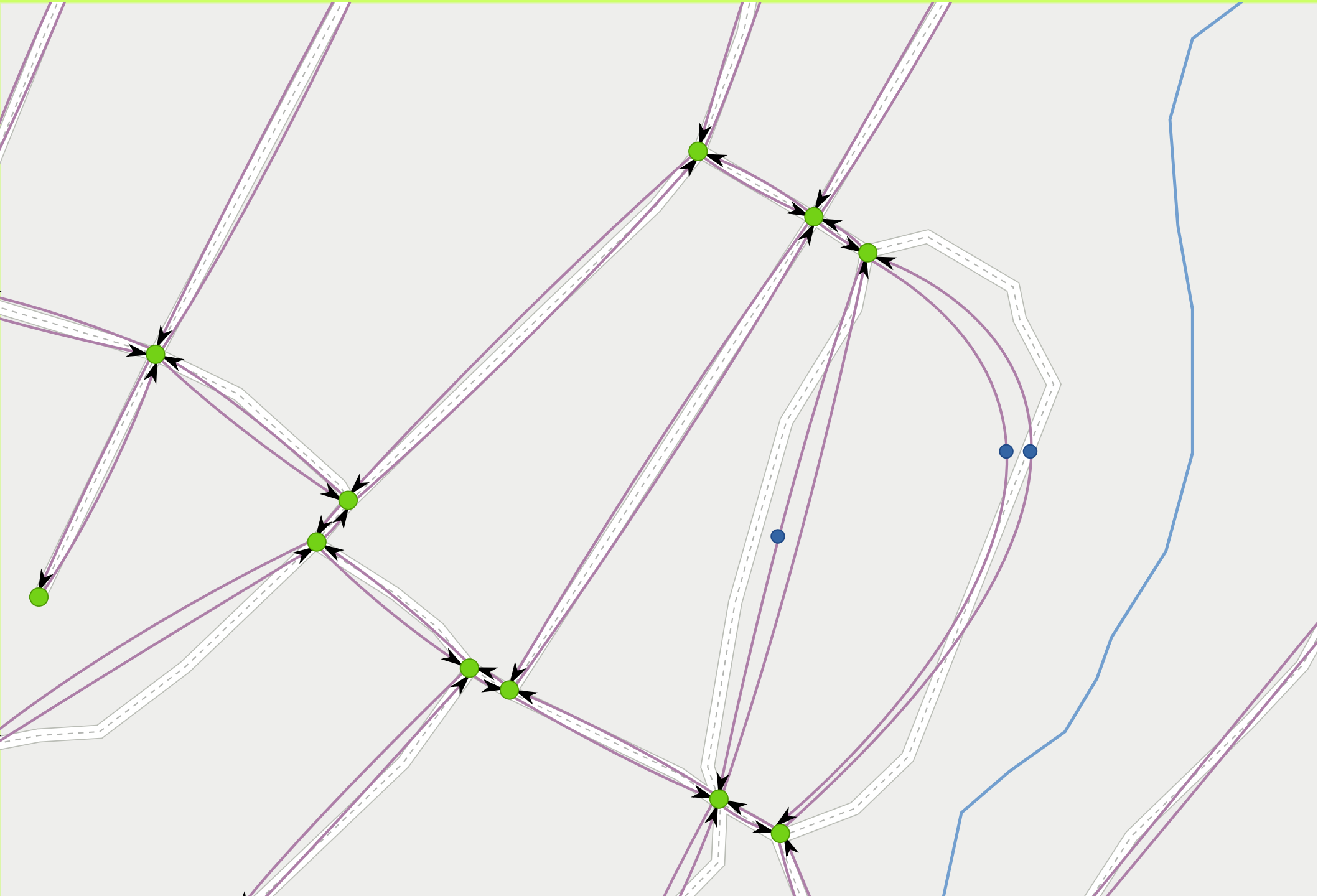
1. background > maps & graphs > topology

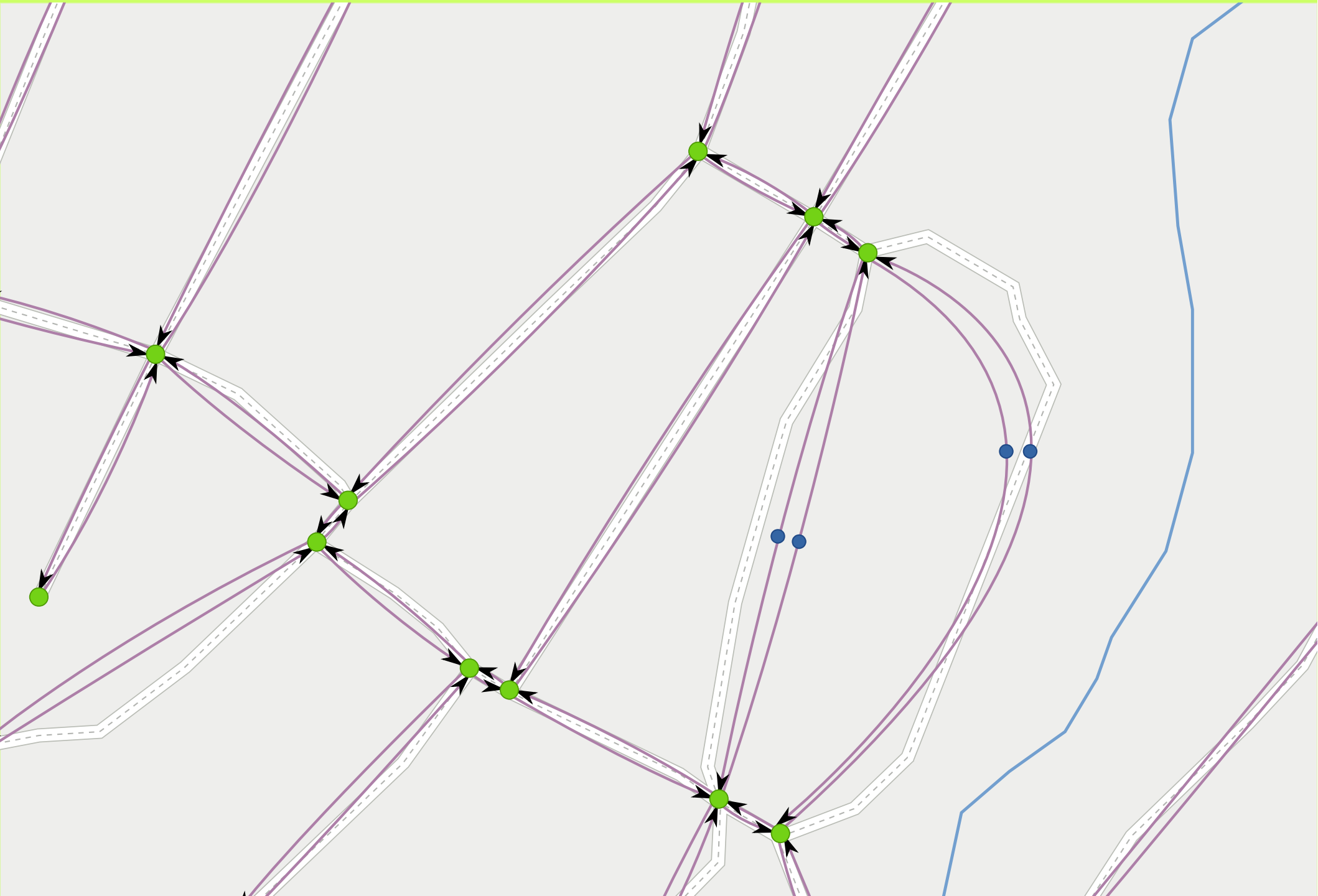

vertex
edge

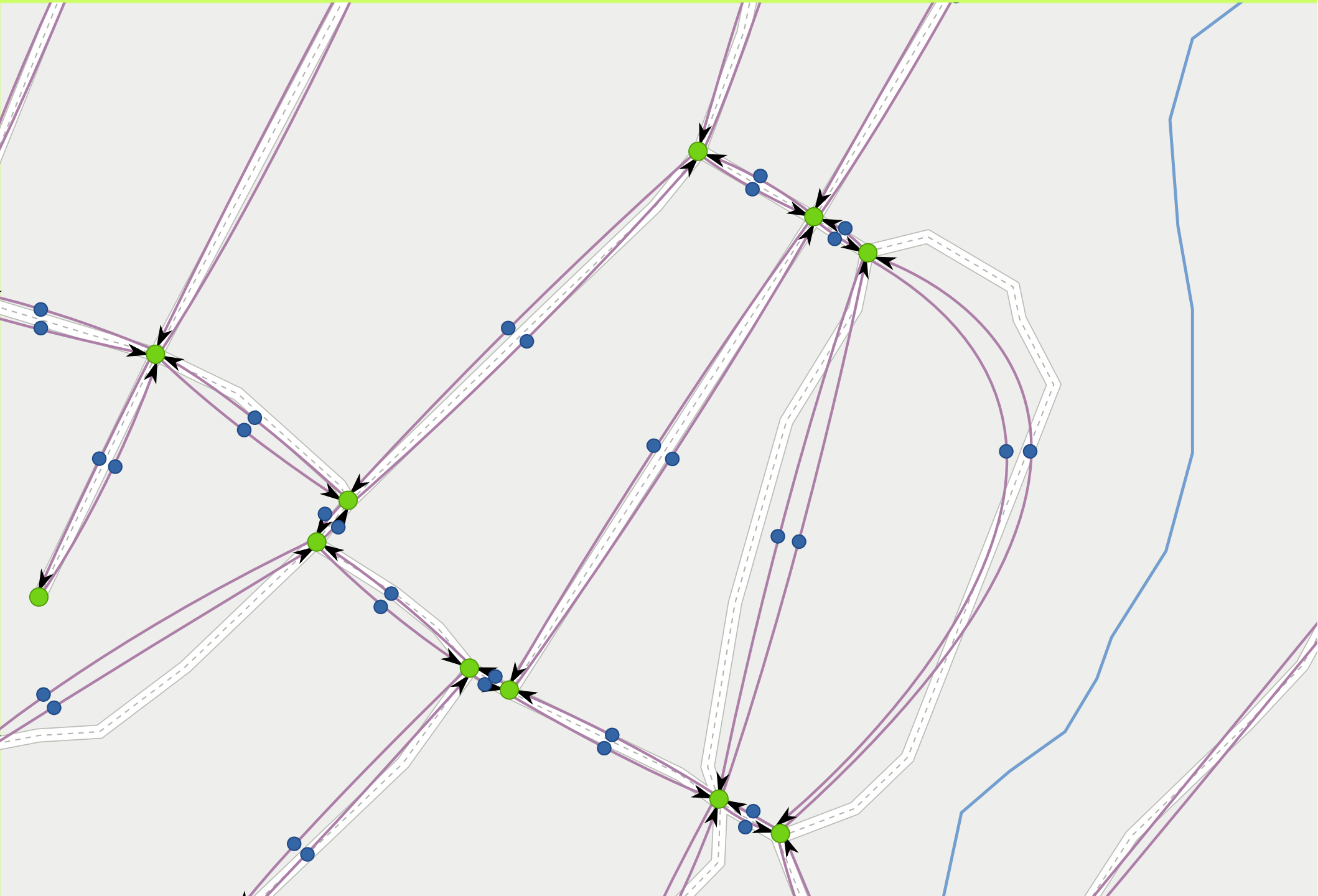


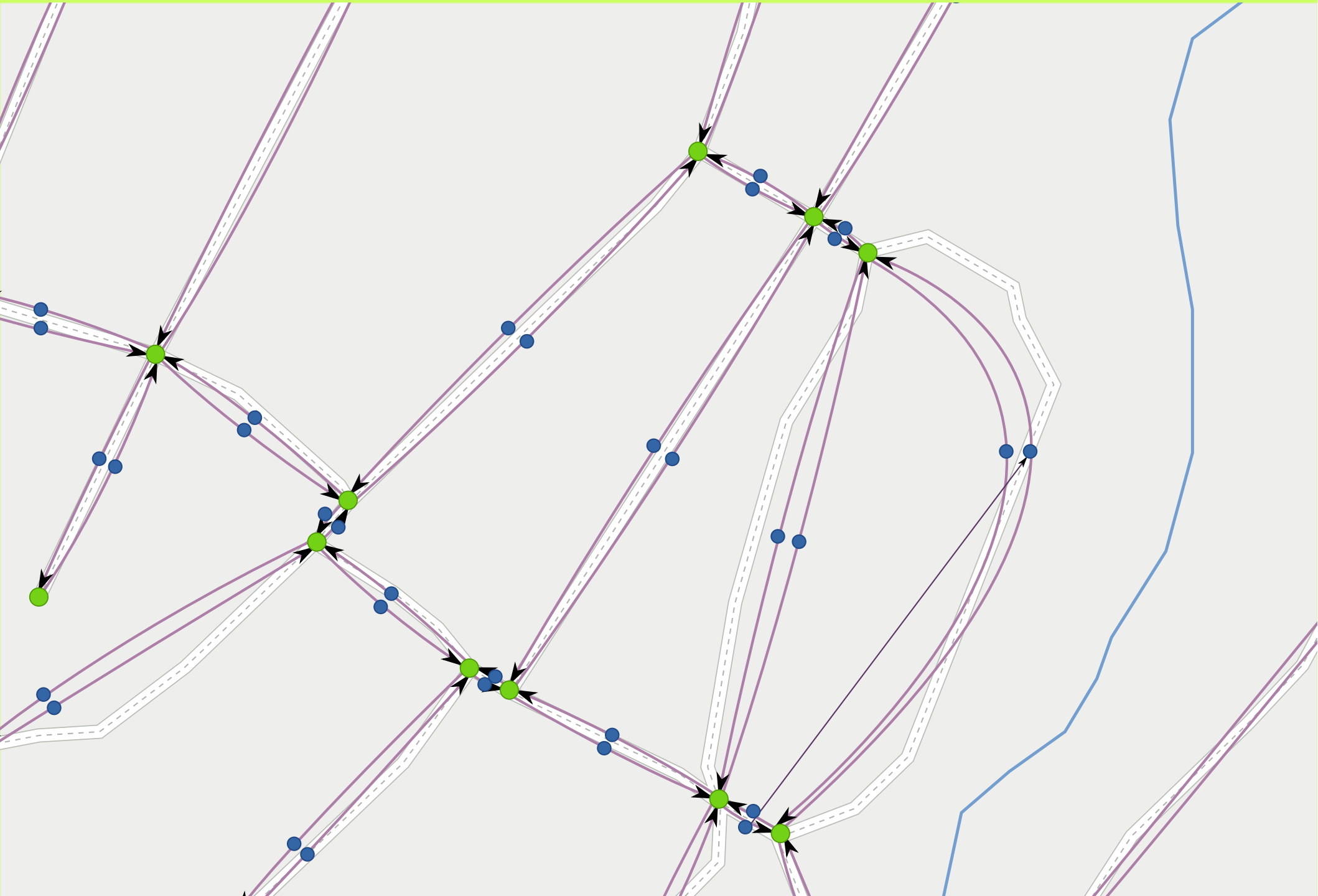


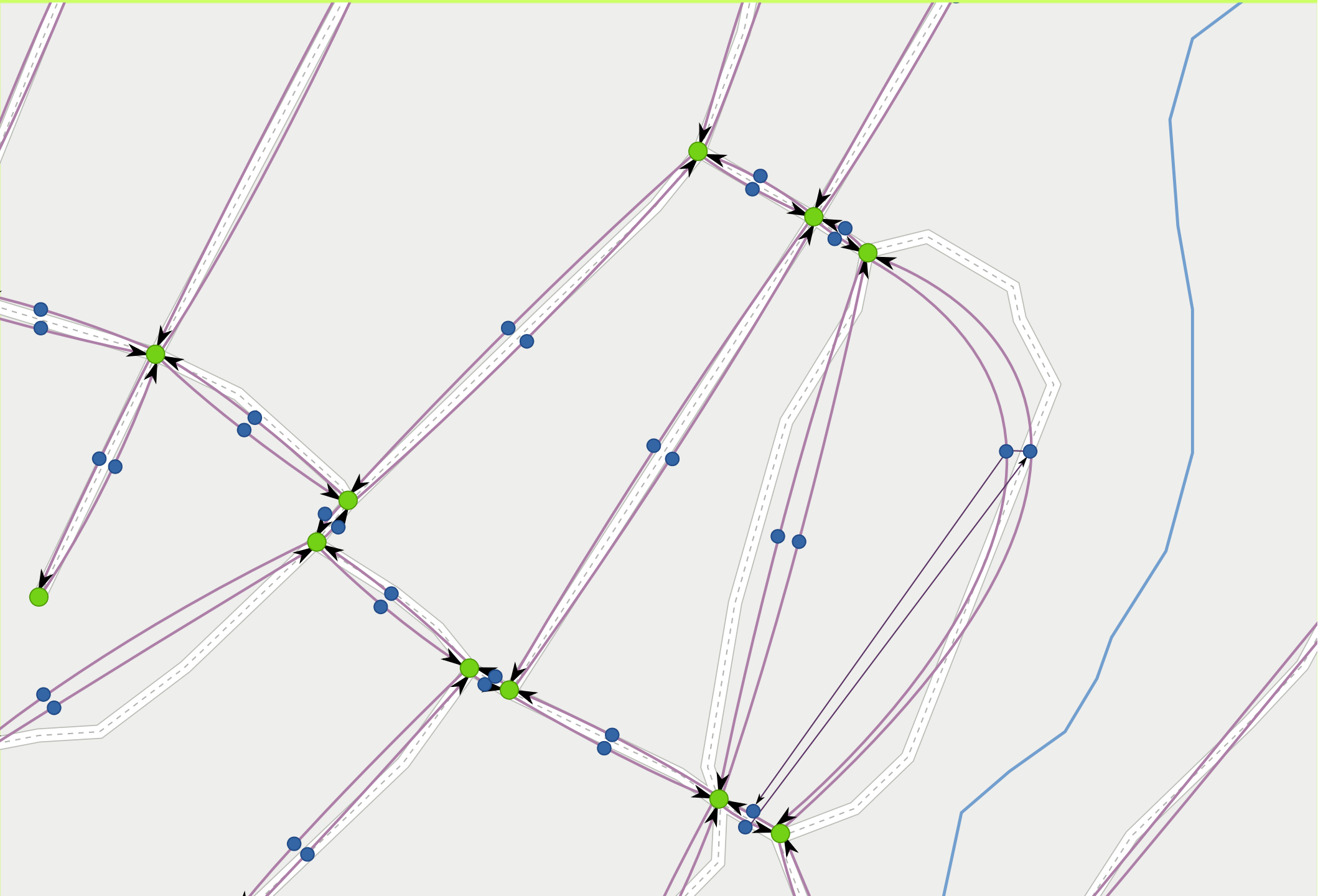


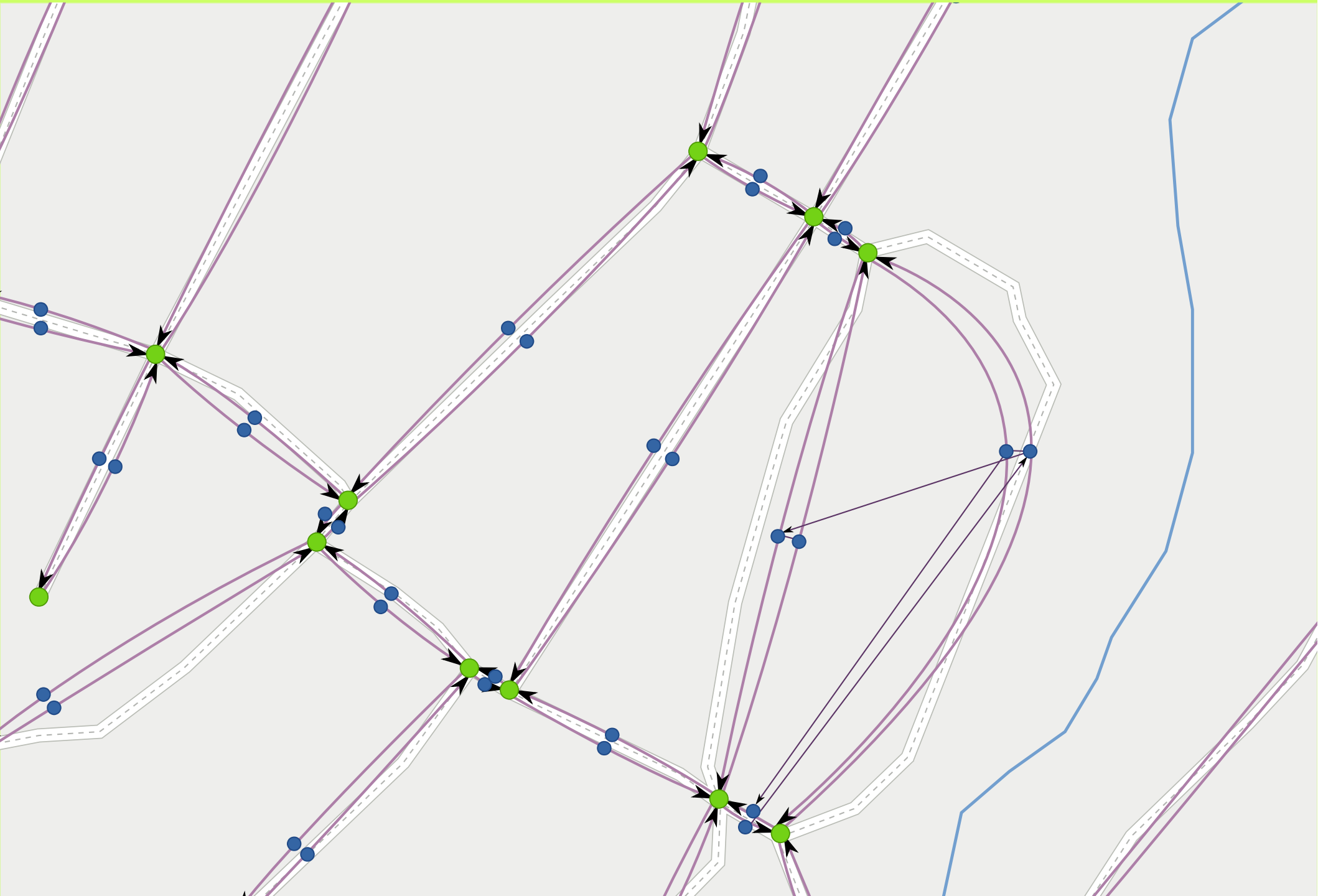


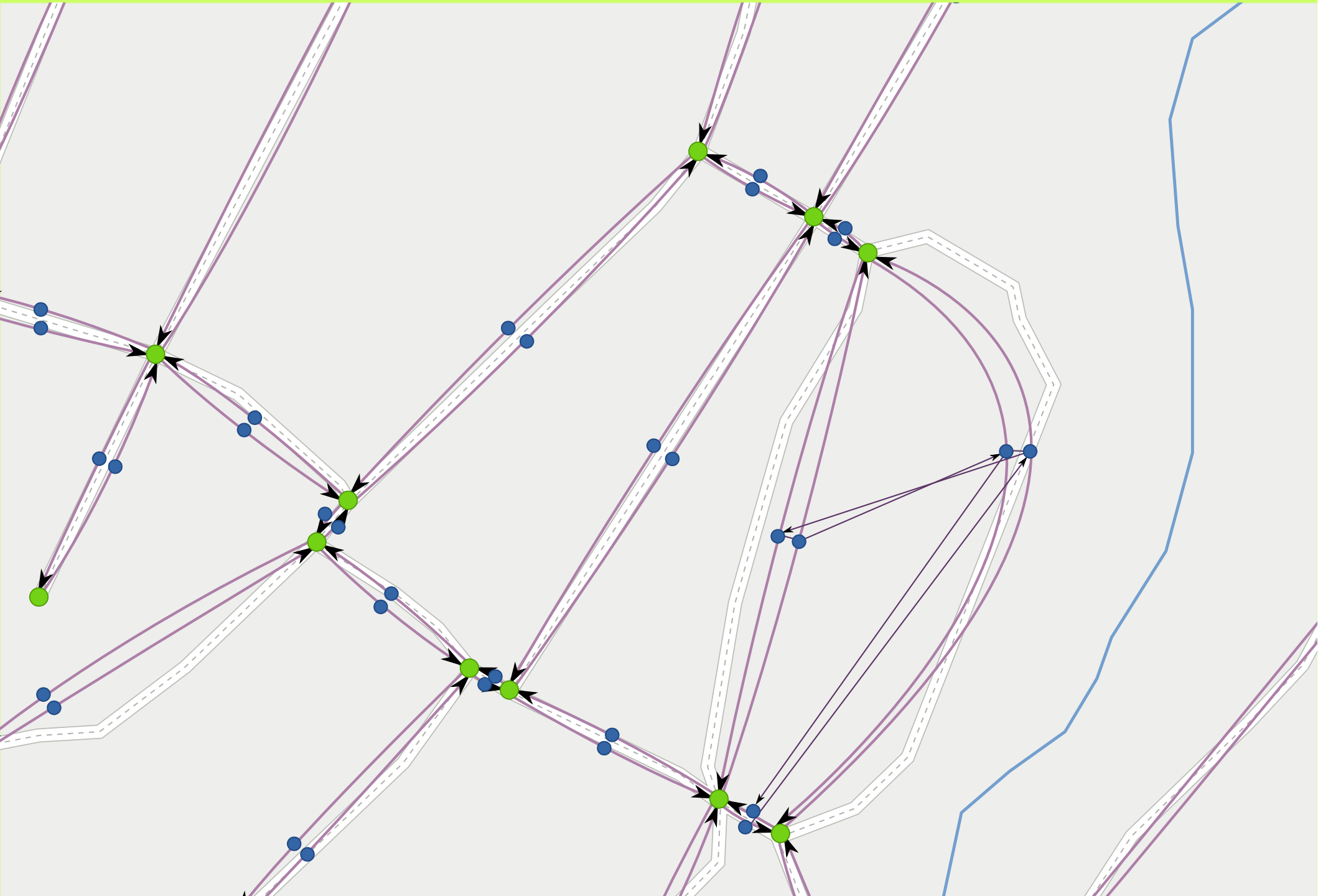


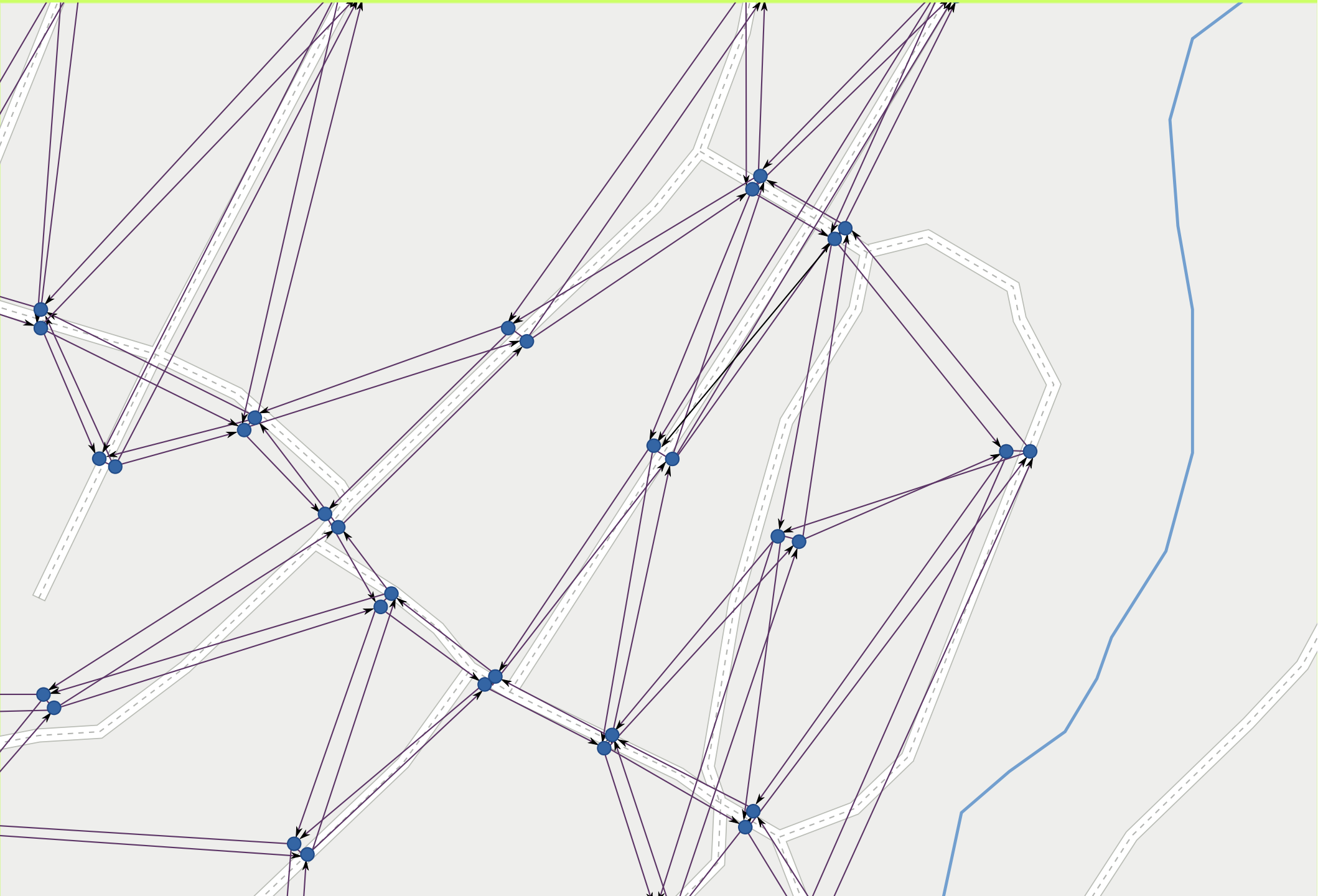












2 *problem* statement

software ***module***

software ***module***

exposing a **function**

2. problem statement

software ***module***

exposing a **function**

returning a

data ***structure***

2. problem statement

software ***module***

exposing a **function**

returning a

data ***structure***

for ***routing***

2. problem statement

software ***module***

exposing a **function**

returning a

data ***structure***

for ***routing***

in *soft* **real-time**

2. problem statement

software ***module***

exposing a **function**

returning a

data ***structure***

for ***routing***

in *soft* **real-time**

sequential operation:

sequential operation:

load ***map data***

sequential operation:

load ***map data***

build topology

sequential operation:

load ***map data***

build topology

apply **restrictions**

sequential operation:

load ***map data***

build topology

apply **restrictions**

build ***directed*** *graph*

sequential operation:

load ***map data***

build topology

apply **restrictions**

build ***directed*** *graph*

return **line graph**

sequential operation:

load ***map data***

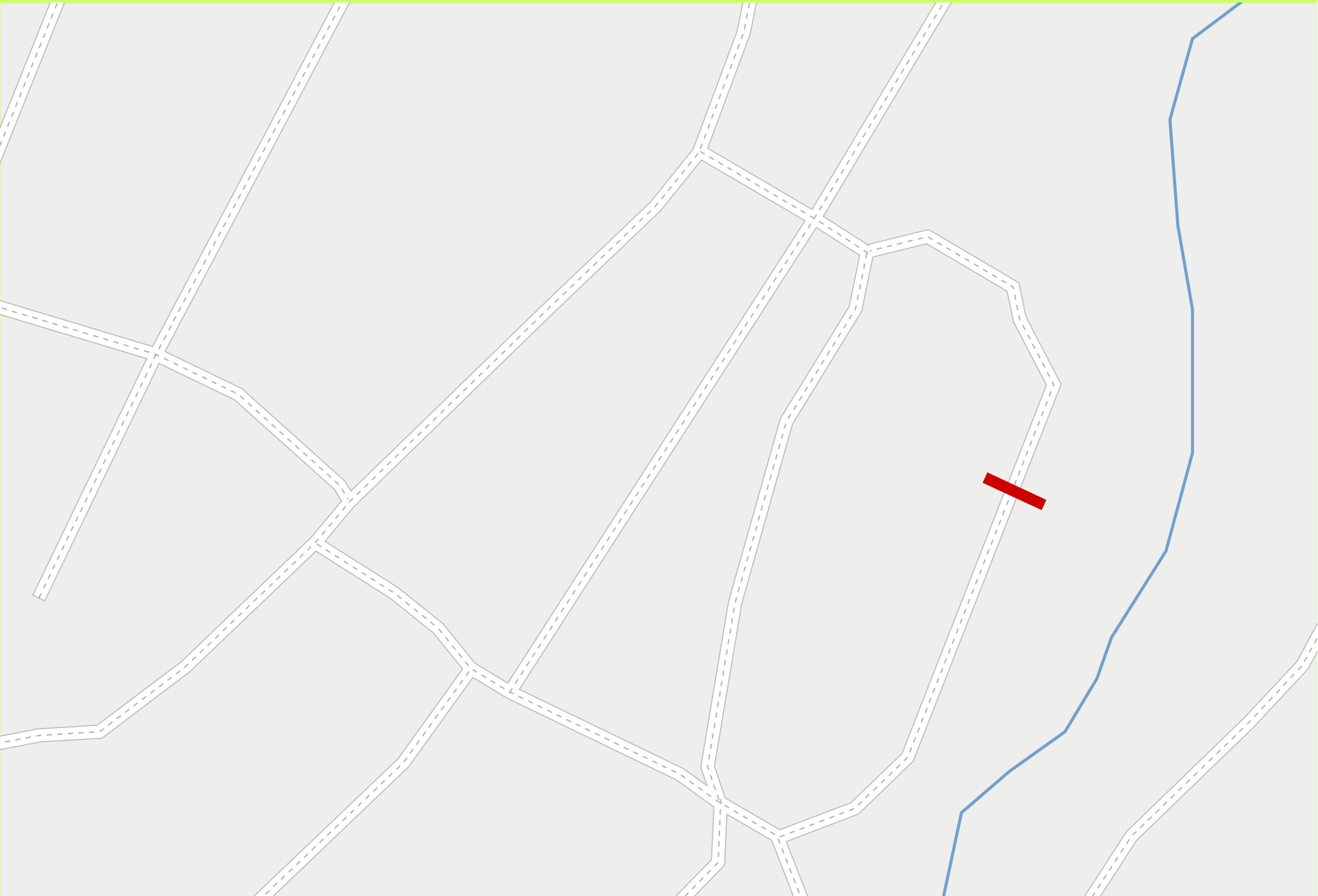
build topology

apply **restrictions**

build ***directed*** *graph*

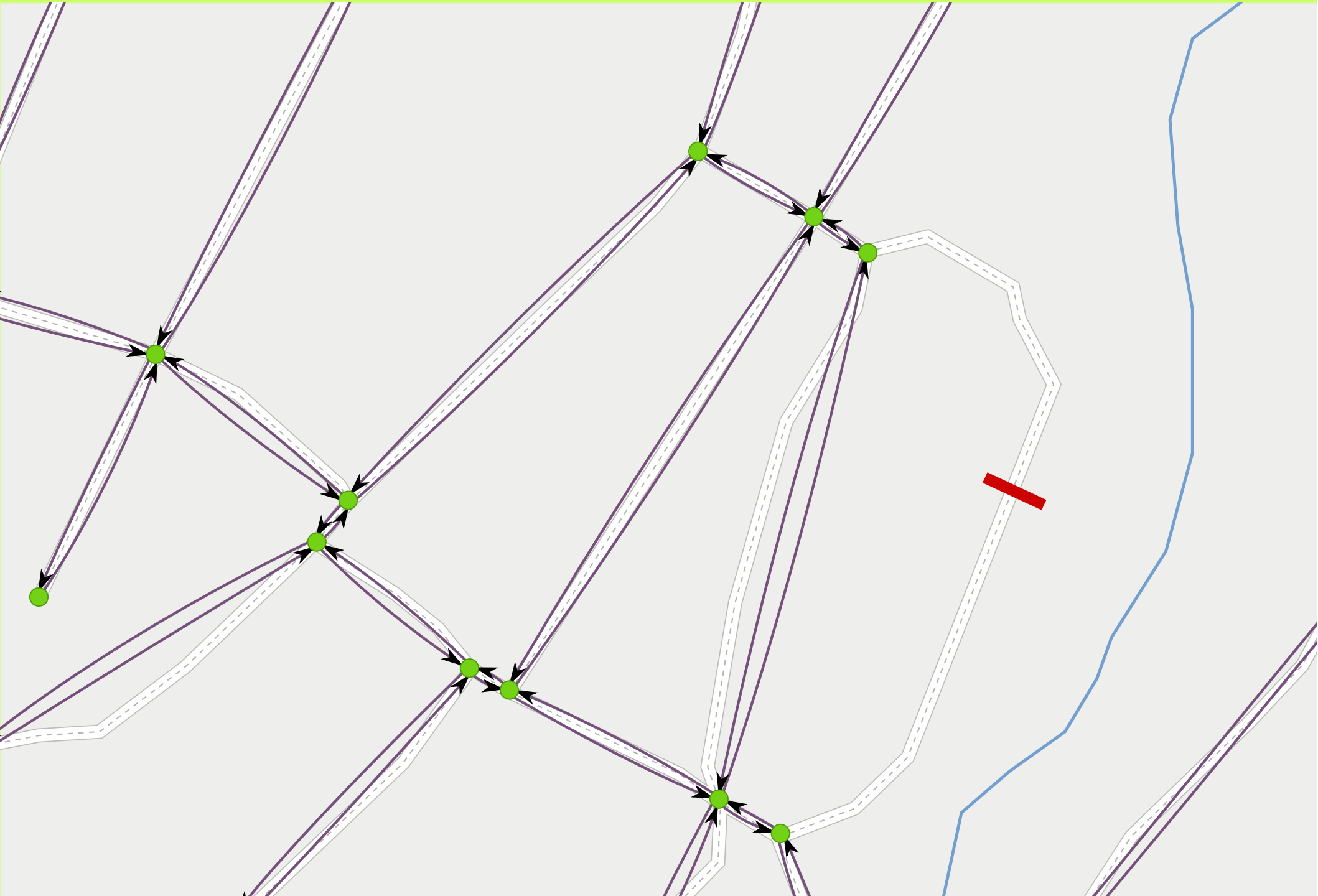
return **line graph**

2. problem statement > applying restrictions > map

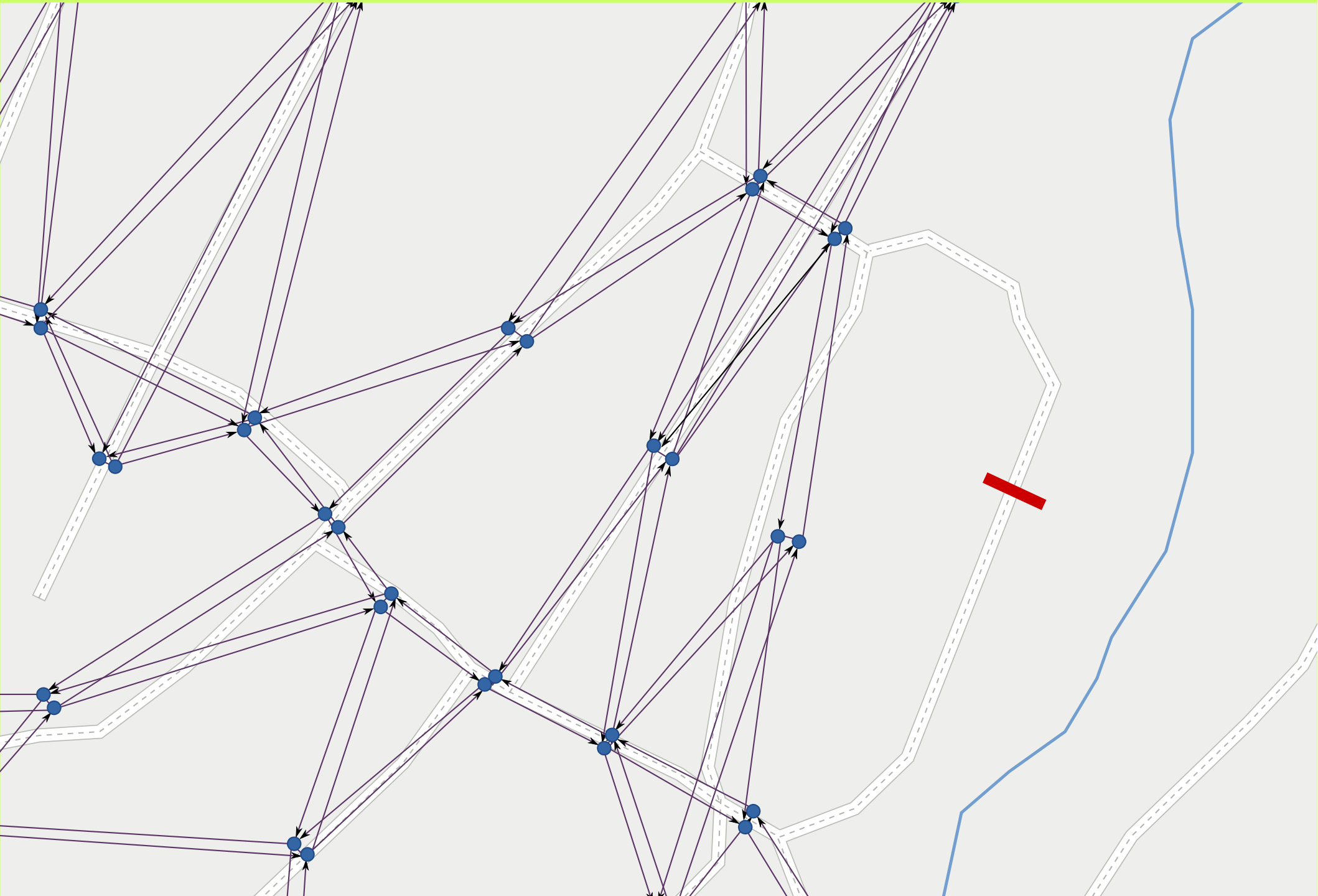


2. problem statement > applying restrictions > topology

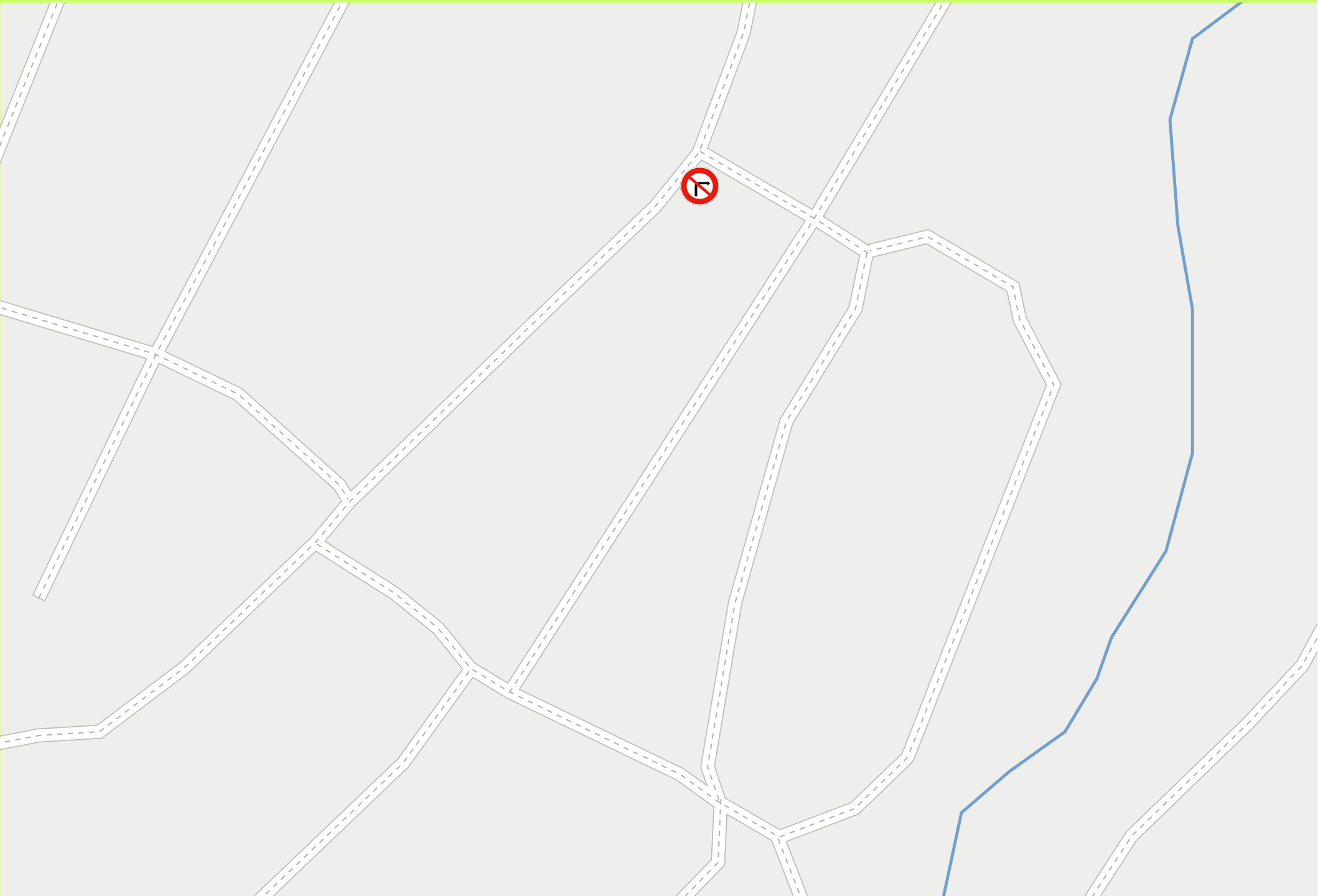
2. problem statement > applying restrictions > directed graph



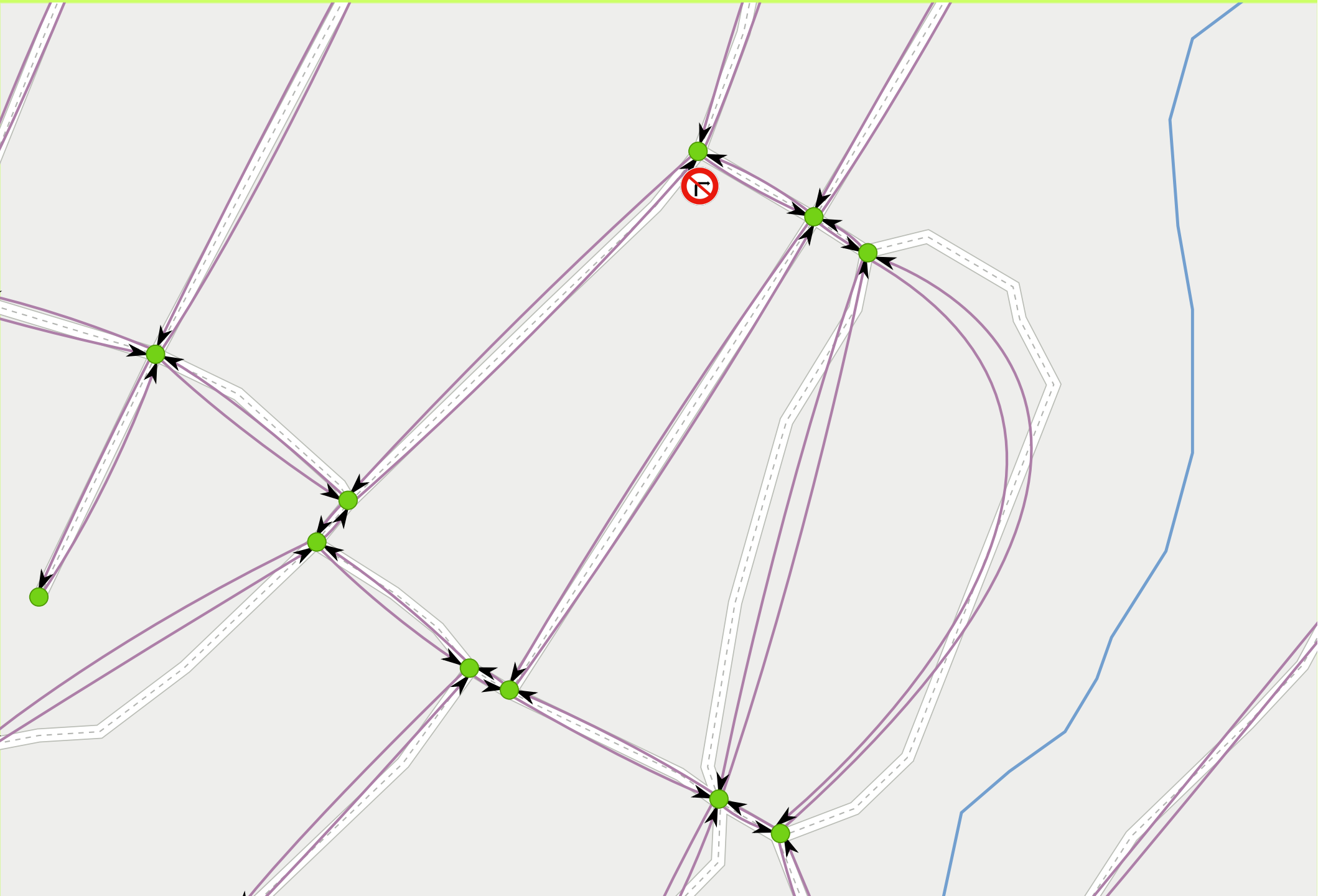
2. problem statement > applying restrictions > line graph



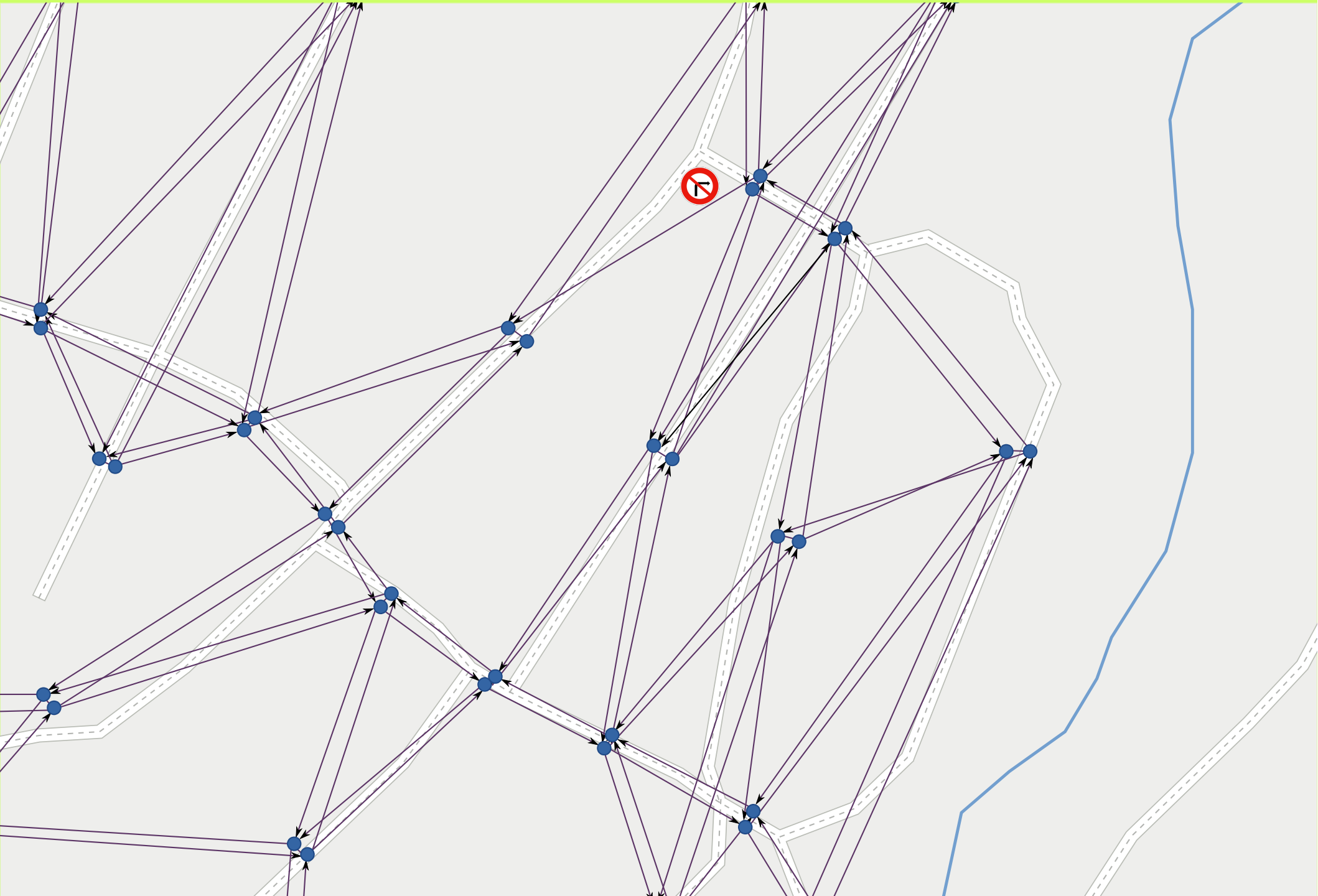
2. problem statement > applying restrictions > map



2. problem statement > applying restrictions > directed graph



2. problem statement > applying restrictions > line graph



sequential operation:

load ***map data***

build topology

preliminary

apply **restrictions**

build ***directed*** *graph*

return **line graph**

sequential operation:

load ***map data***

build topology

on demand

apply **restrictions**

build ***directed*** *graph*

return **line graph**

required tools:

required tools:

C++

required tools:

map *data*

required tools:

map *data*

OpenStreetMap

required tools:

map *data*

OpenStreetMap

PostGIS

required tools:

graph *data structures*

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 driven *development* (BDD)

behavior driven *development* (BDD)

```
Scenario:
```

behavior driven *development* (BDD)

```
Scenario:
    Given:
```

behavior driven *development* (BDD)

```
Scenario:
   Given:
    When:
```

behavior driven *development* (BDD)

```
Scenario:
   Given:
    When:
    Then:
```

behavior driven *development* (BDD)

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

behavior driven *development* (BDD)

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

behavior driven *development* (BDD)

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

behavior driven *development* (BDD)

```
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

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;
}
```

osm2pgsql

source: **OpenStreetMap**

load:

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

store: **PostGIS**

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);"
```

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

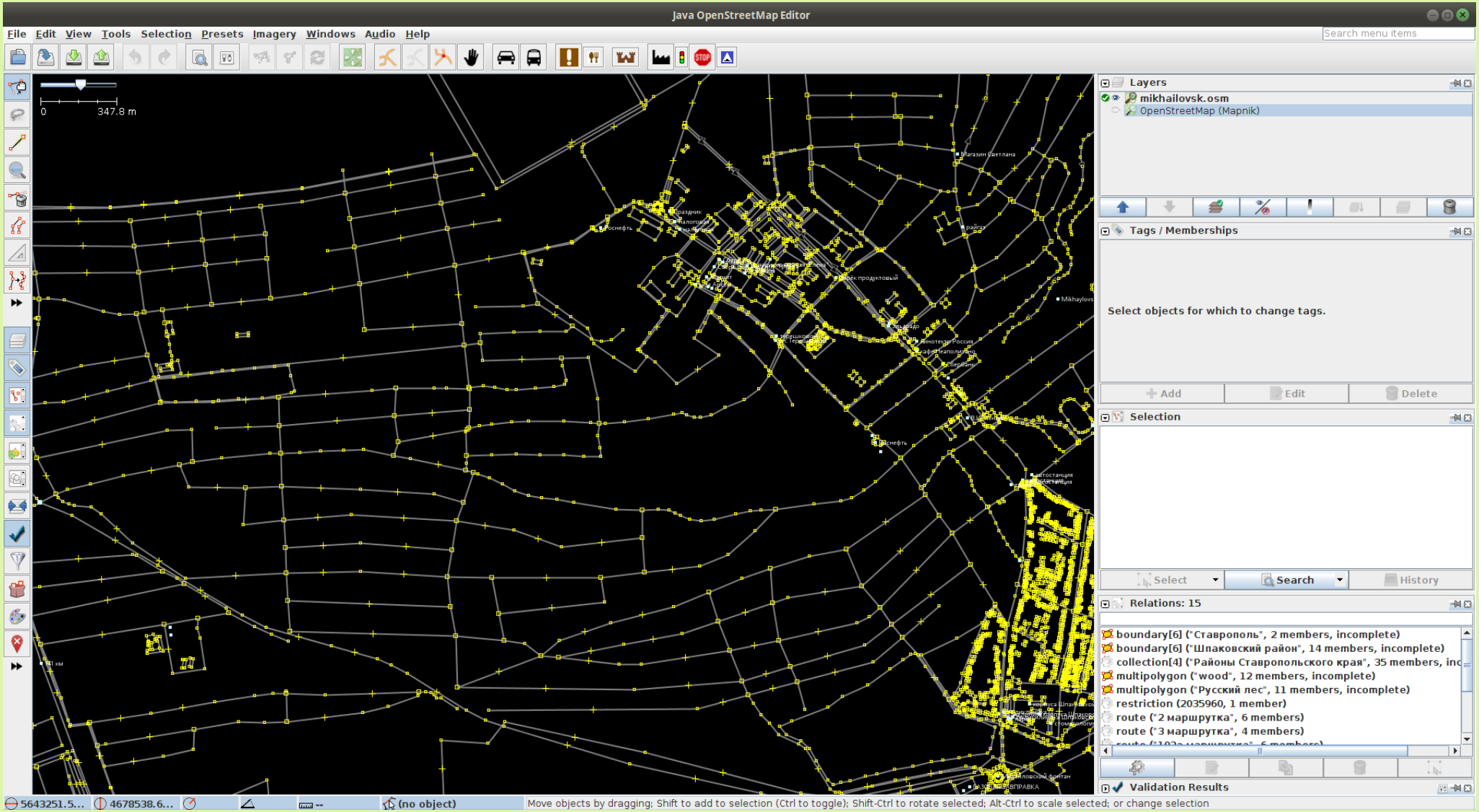
```
struct City {
  string      name;
  int         population;
  vector<int> zipcodes;
};

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;
```

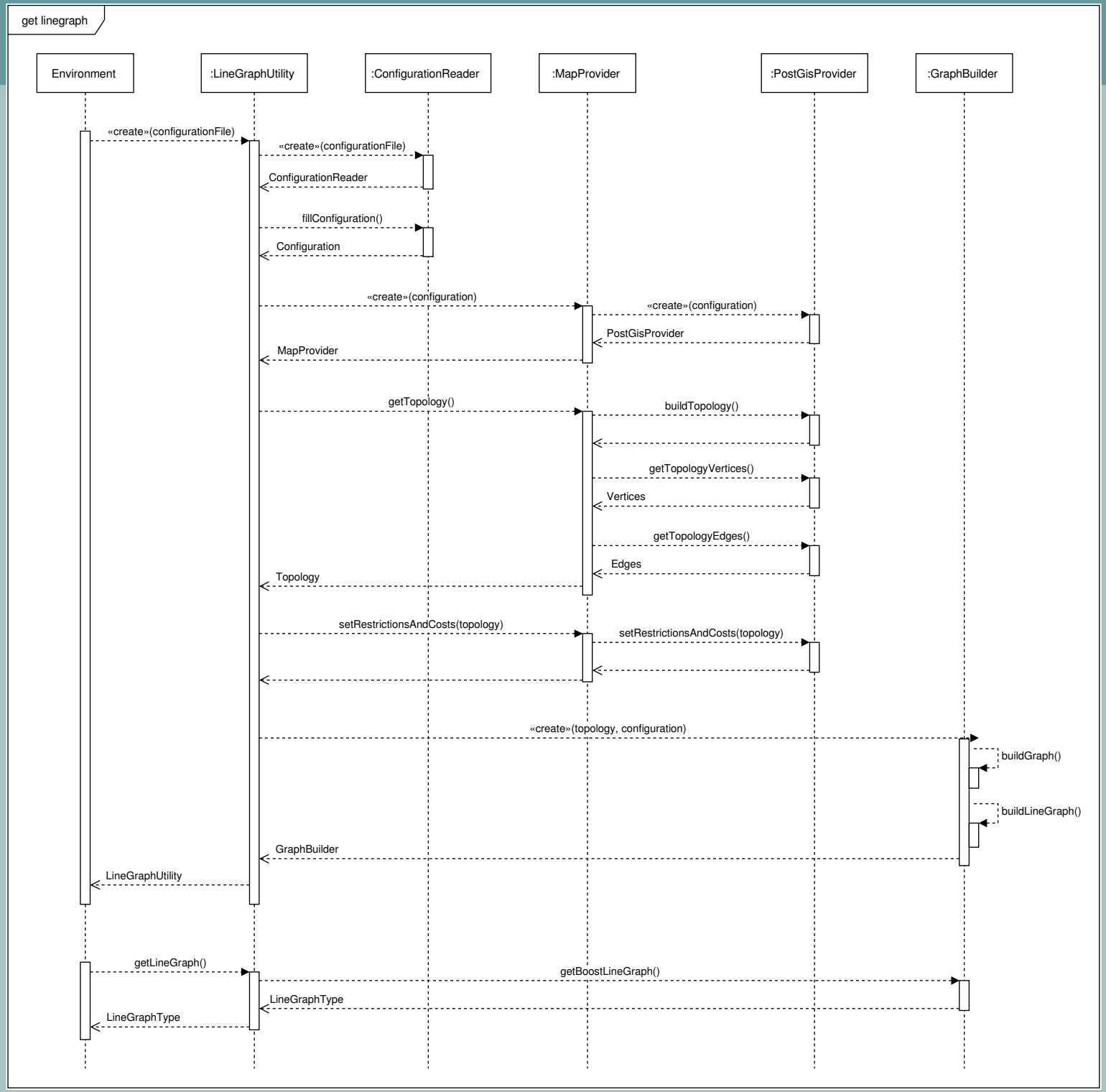
3. method > tools > edit map data

JOSM

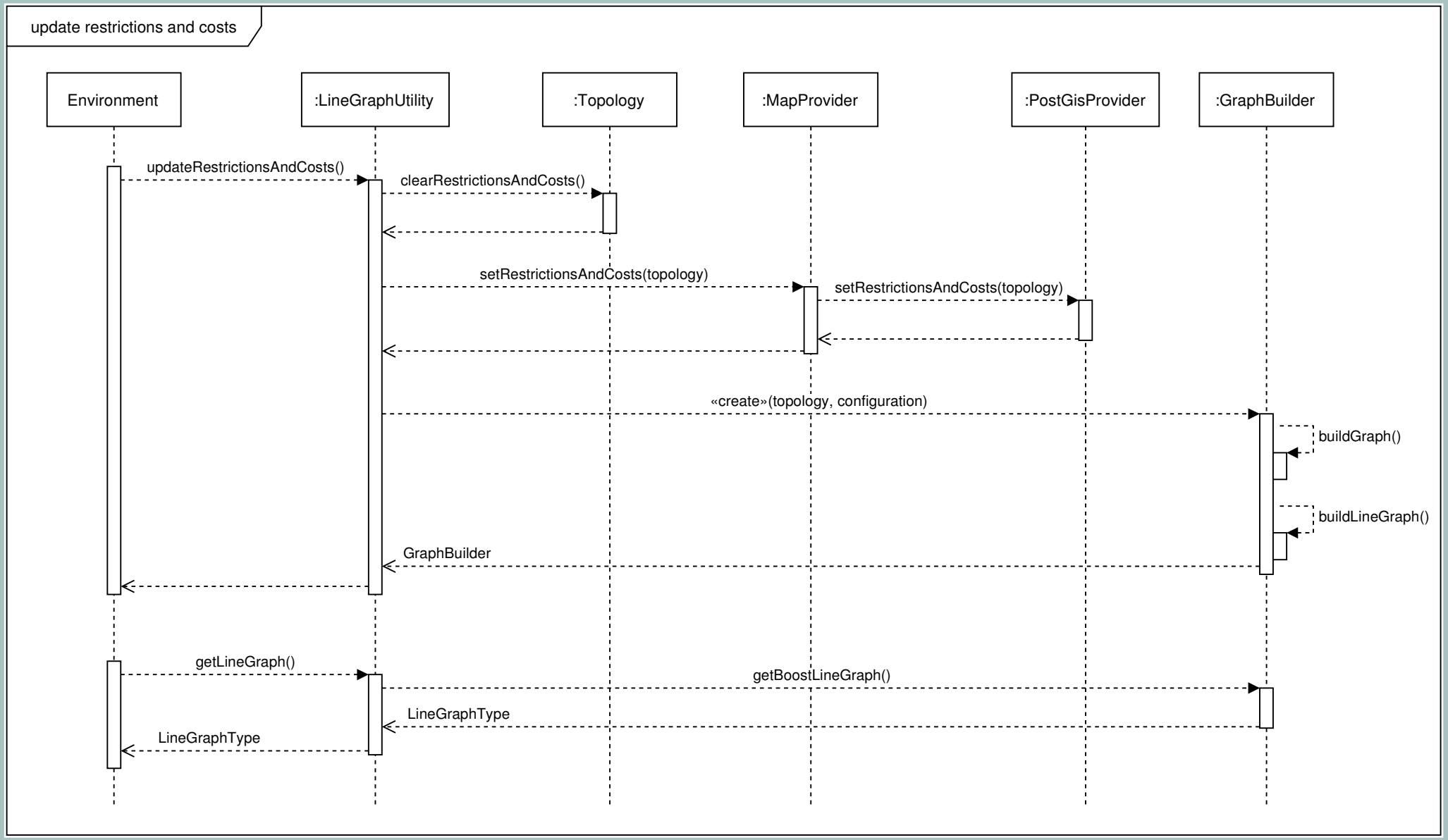


4 results

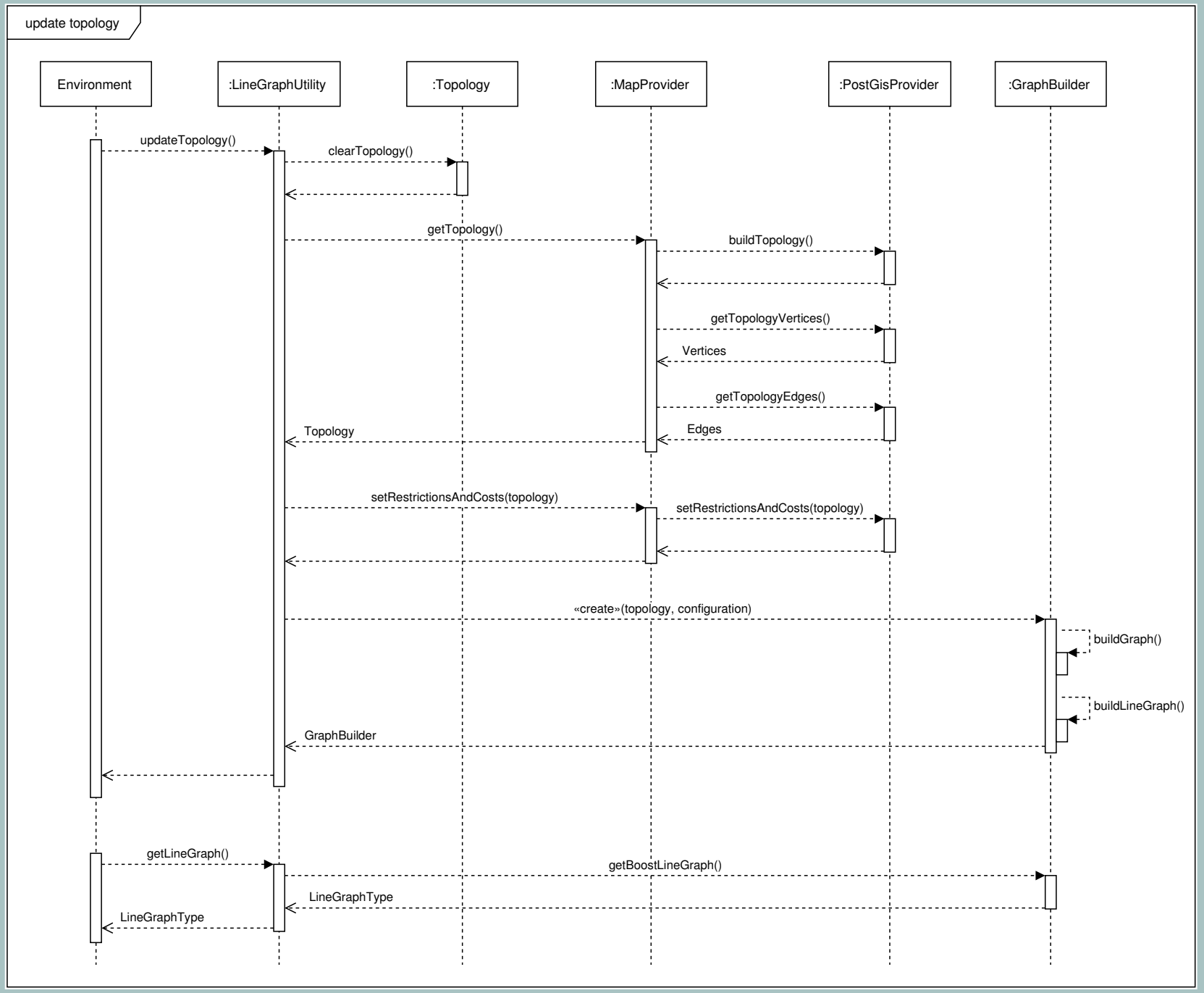
4. results > design



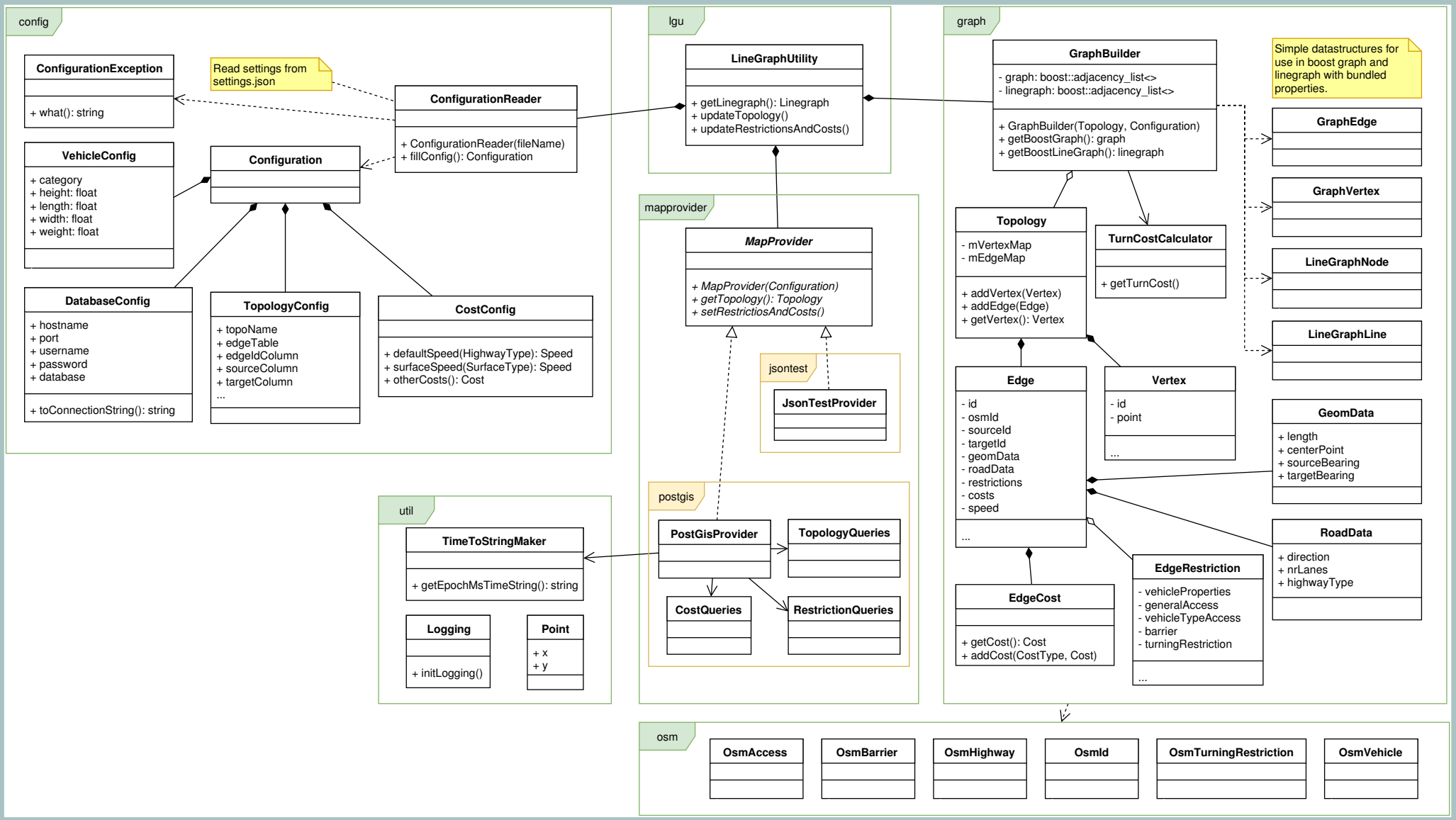
4. results > design



4. results > design



4. results > design



tests

specification **fulfillment**

tests

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

tests

4. results

conditional
restrictions

```
motor_vehicle=no
motor_vehicle:conditional=yes @ (18:30-07:30)
psv=yes
```

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

time of day, inclination ...

tests

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

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

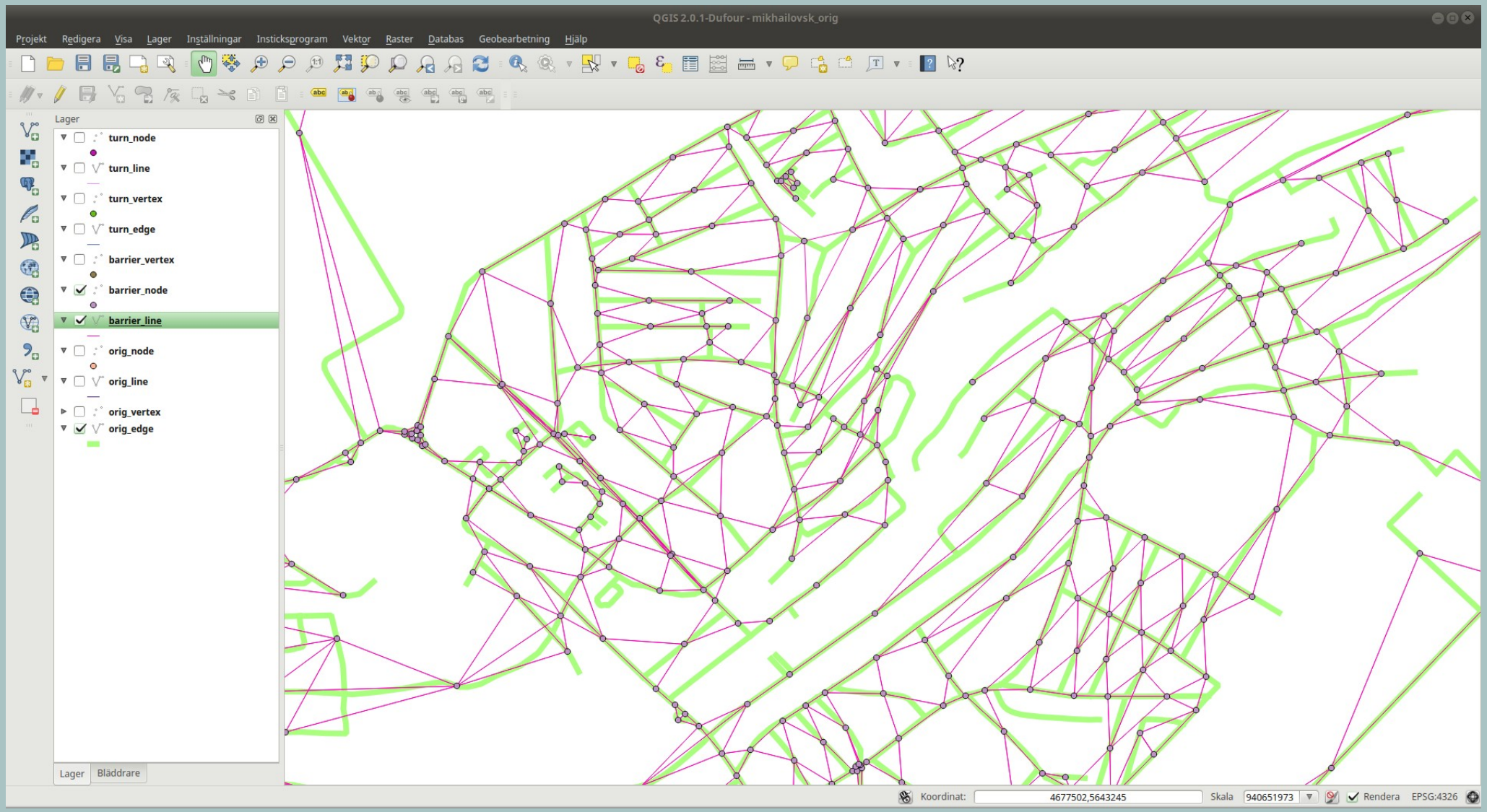
time of day, inclination ...

tests

performance

visual examination

4. results



QGIS

visual examination

4. results

specification **fulfillment**

except *restrictions*

most edge, no conditional ...

conditions

time of day, inclination ...

tests

performance

visual examination

5 conclusions

working

working but *incomplete*

working but ***incomplete***

due to *time*

working but *incomplete*

due to *time*

late *specification*

working but *incomplete*

due to *time*

late *specification*

complex *restrictions*

6 *future* work

re-model

restrictions

re-model restrictions

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

LineGraph Utility

Jonas Bergman, jobe0900@student.miun.se