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**?

+ economy
+ environment

less

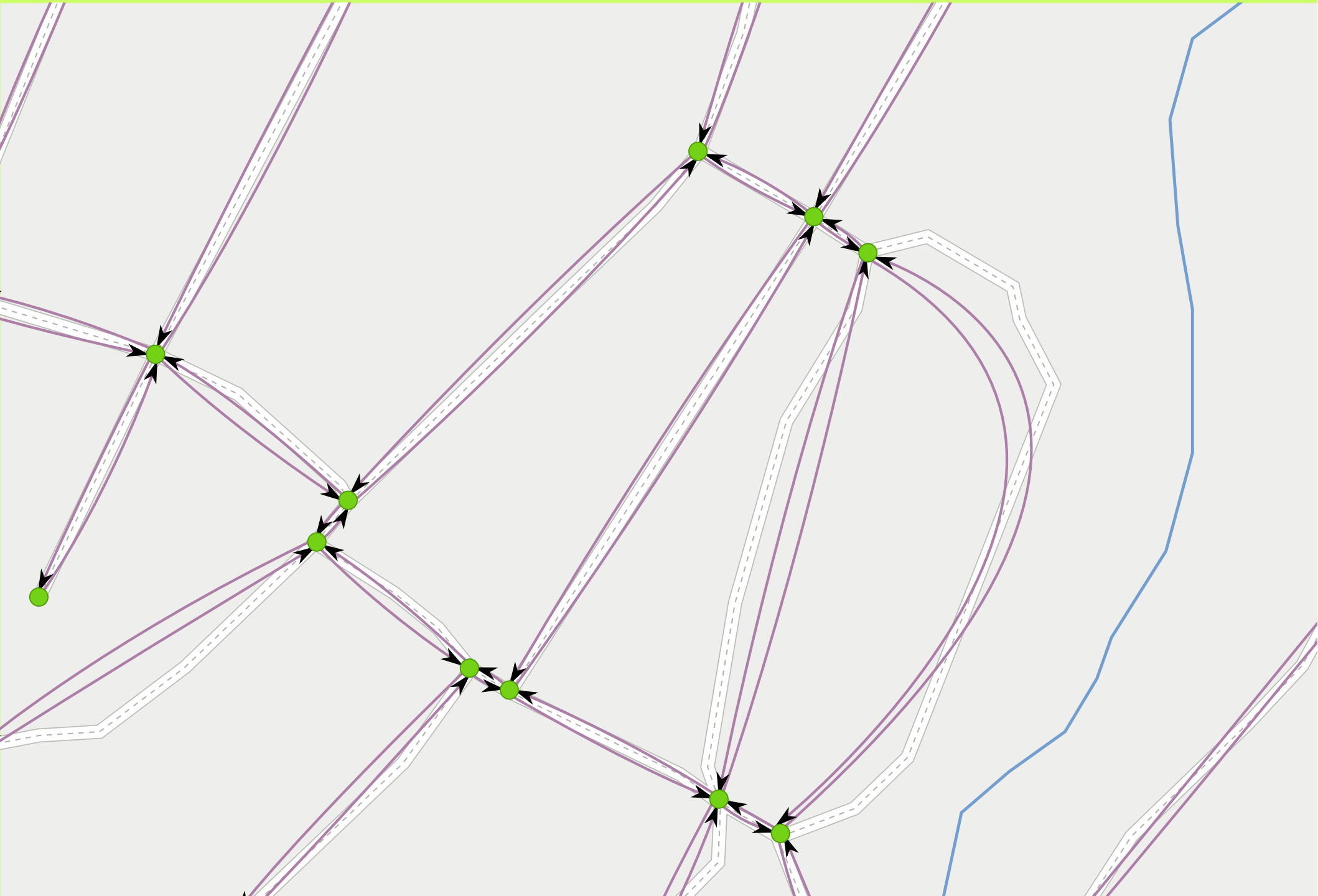
idle vehicles

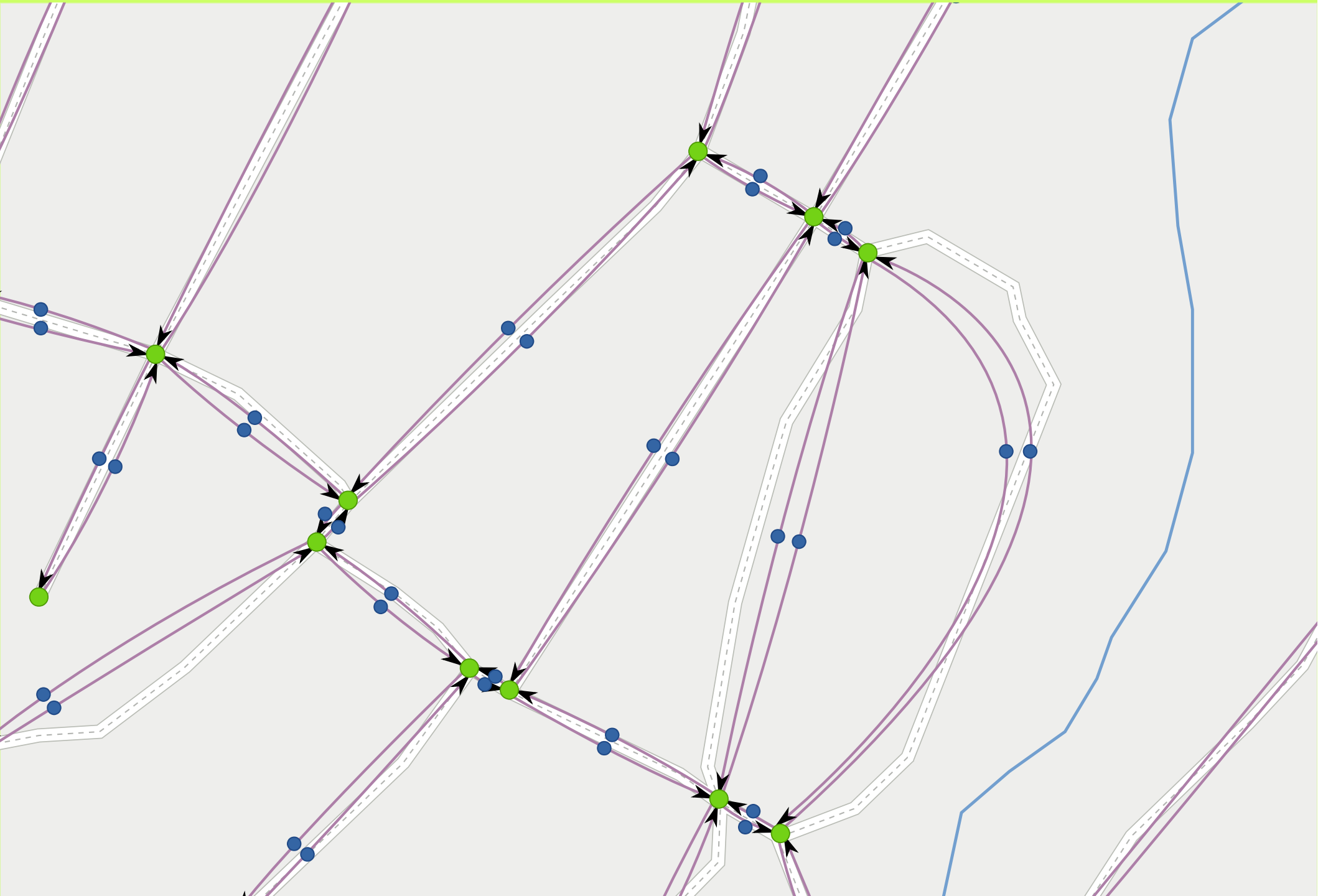
flexible public transportation: **gain**?

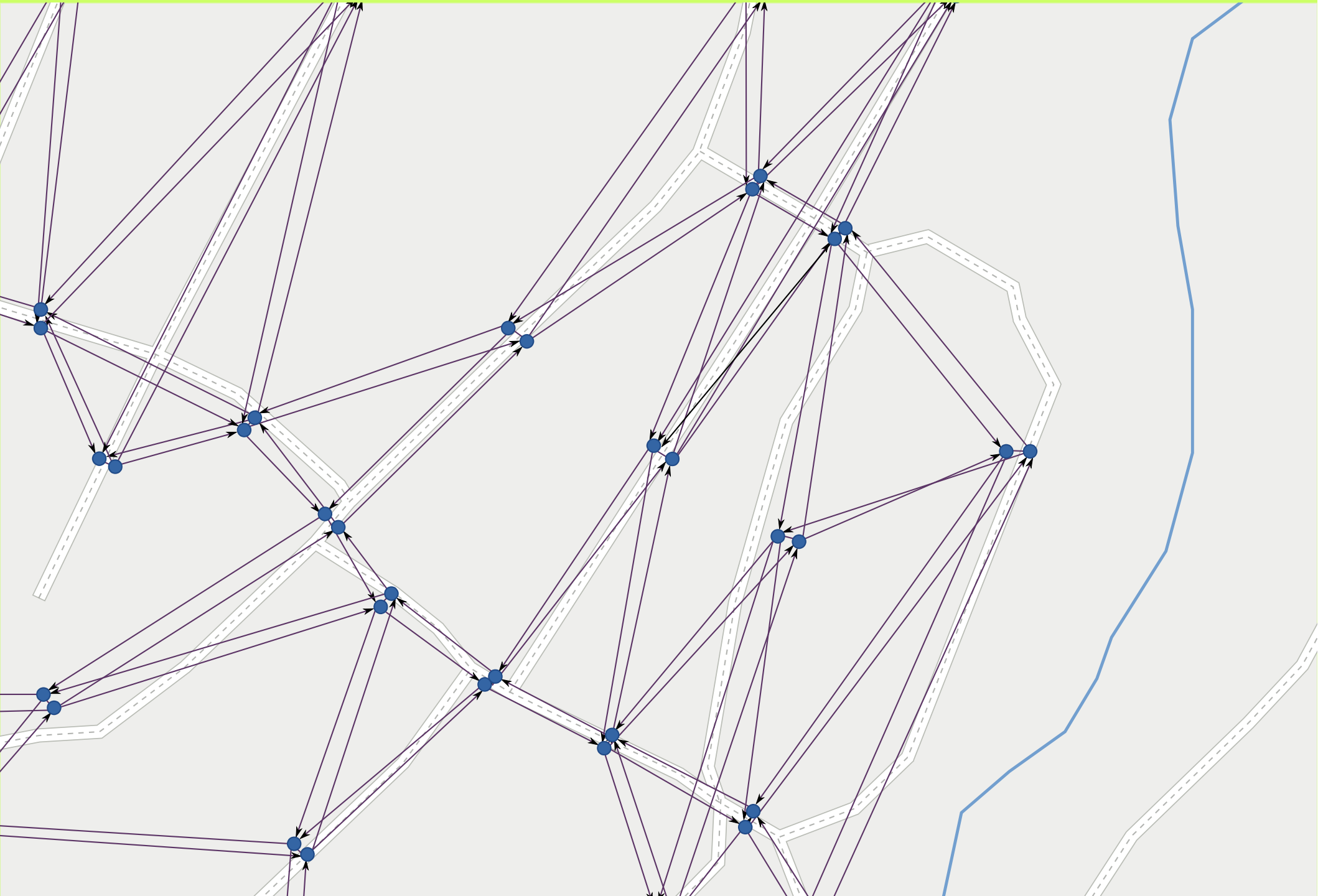
- less idle vehicles

less *waiting*

1. background > maps & graphs







2 problem statement

software ***module***

exposing a **function**

returning a

data ***structure***

for ***routing***

in *soft* **real-time**

software ***module***
exposing a **function**
returning a
data ***structure***
for ***routing***
in *soft* **real-time**

sequential operation:

load ***map data***

build topology

apply **restrictions**

build ***directed*** *graph*

return **line graph**

sequential operation:

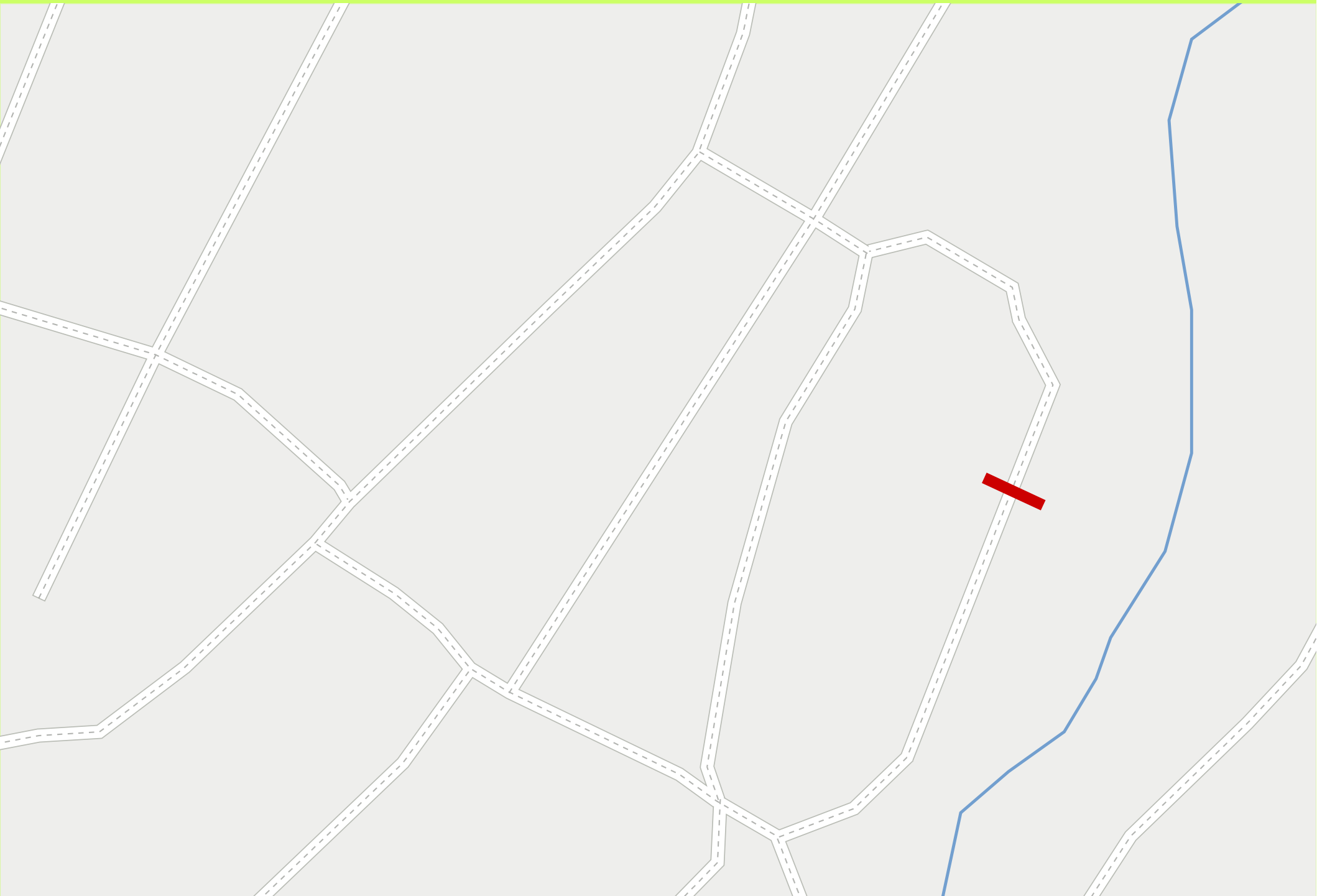
load ***map data***

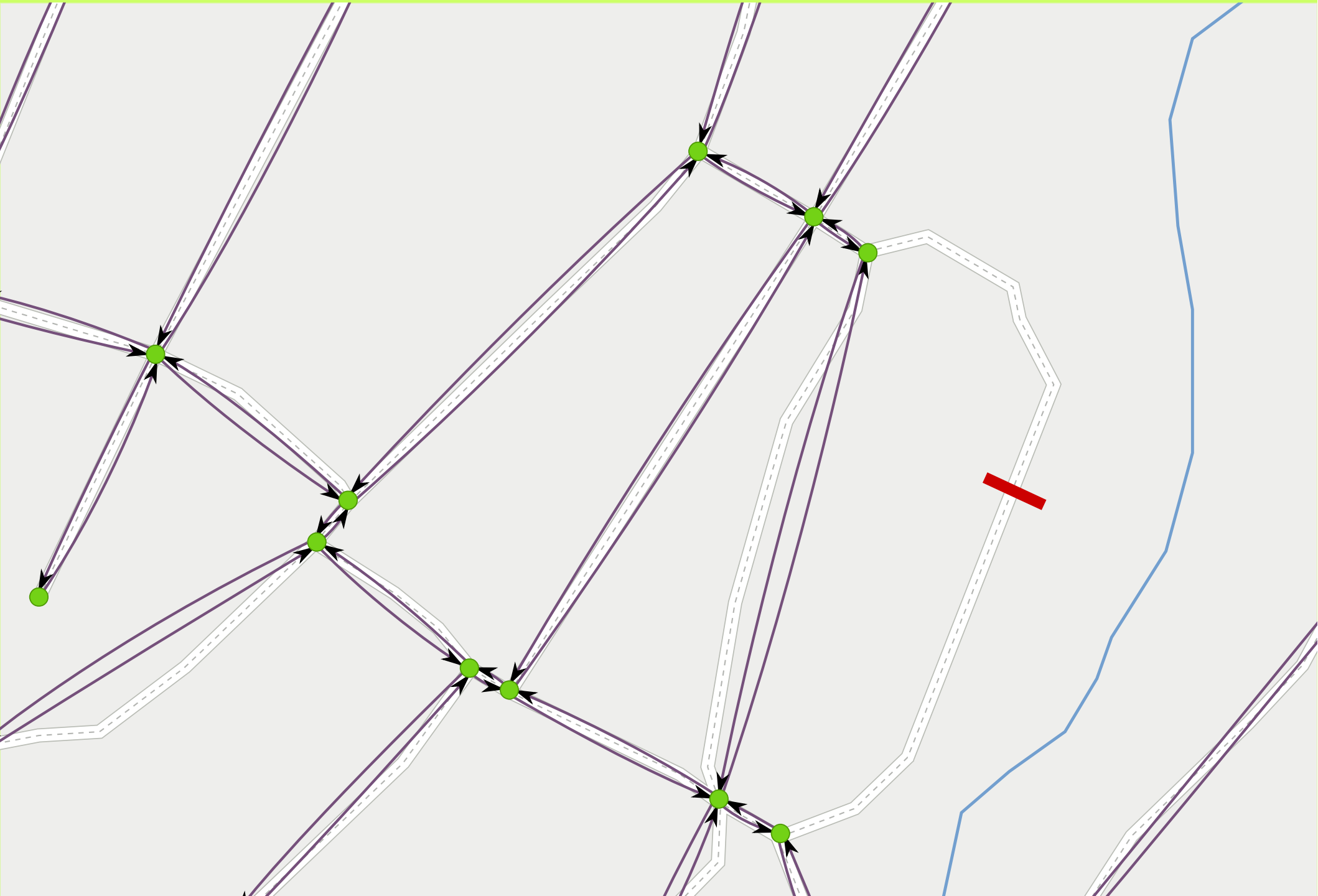
build topology

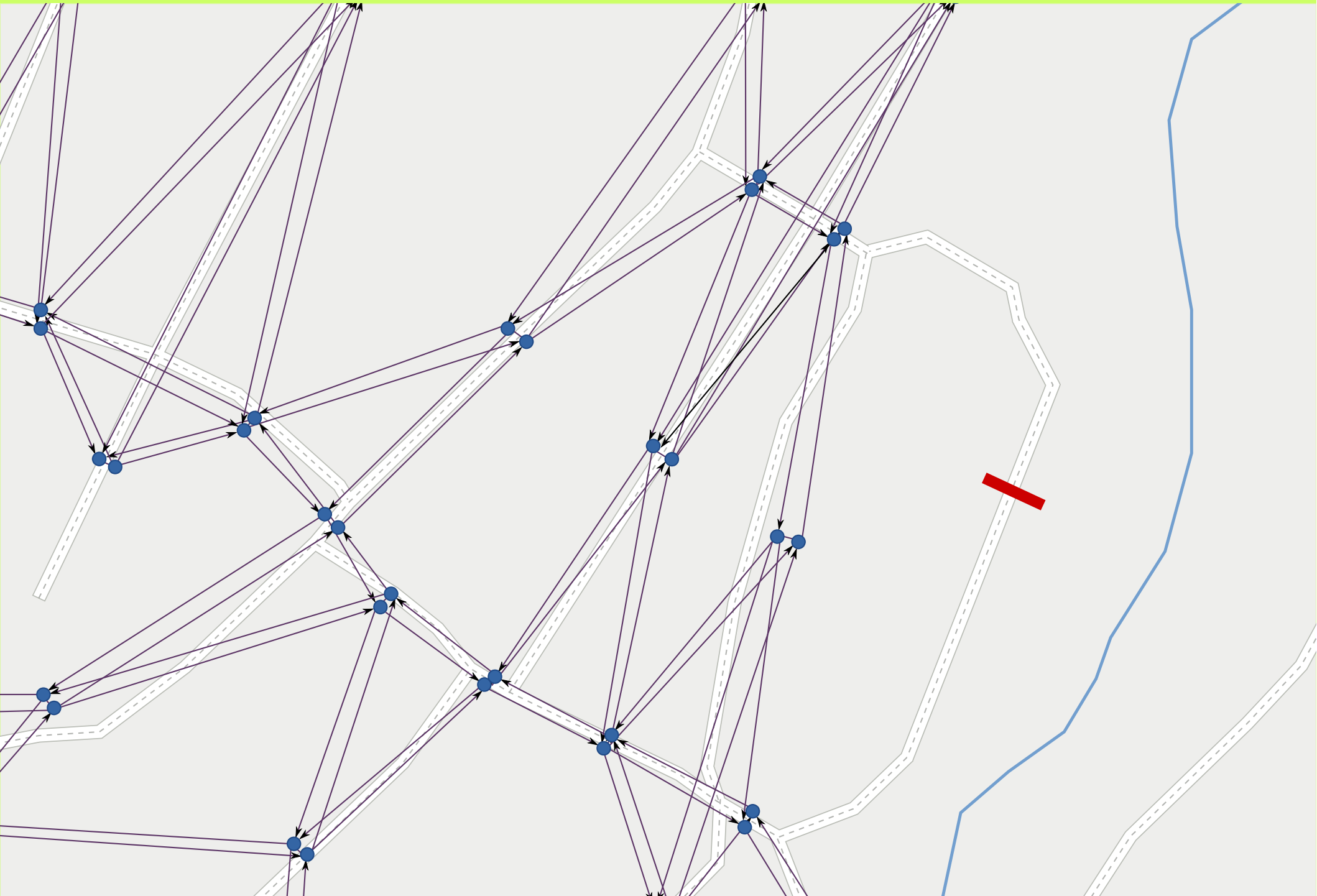
apply **restrictions**

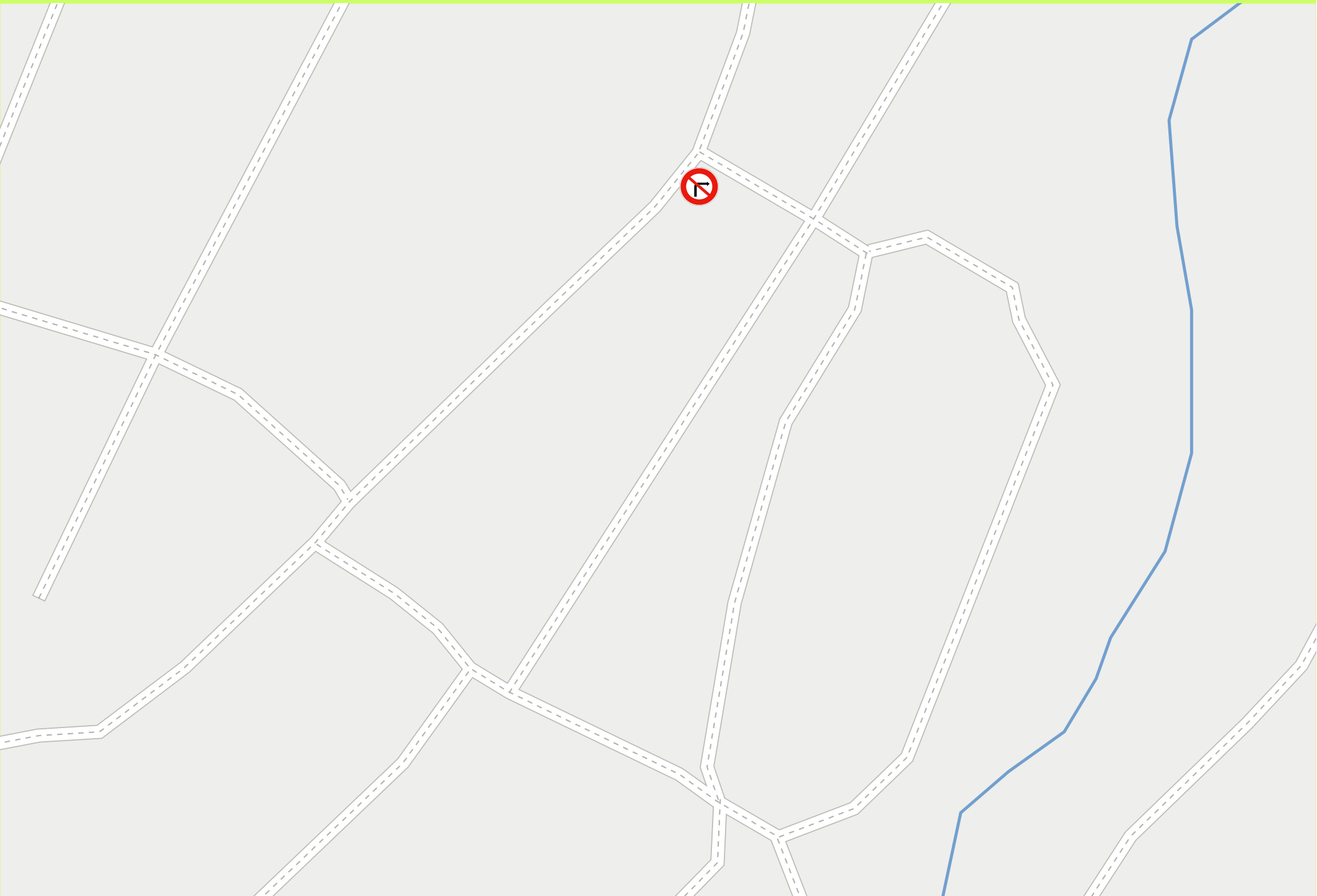
build ***directed*** *graph*

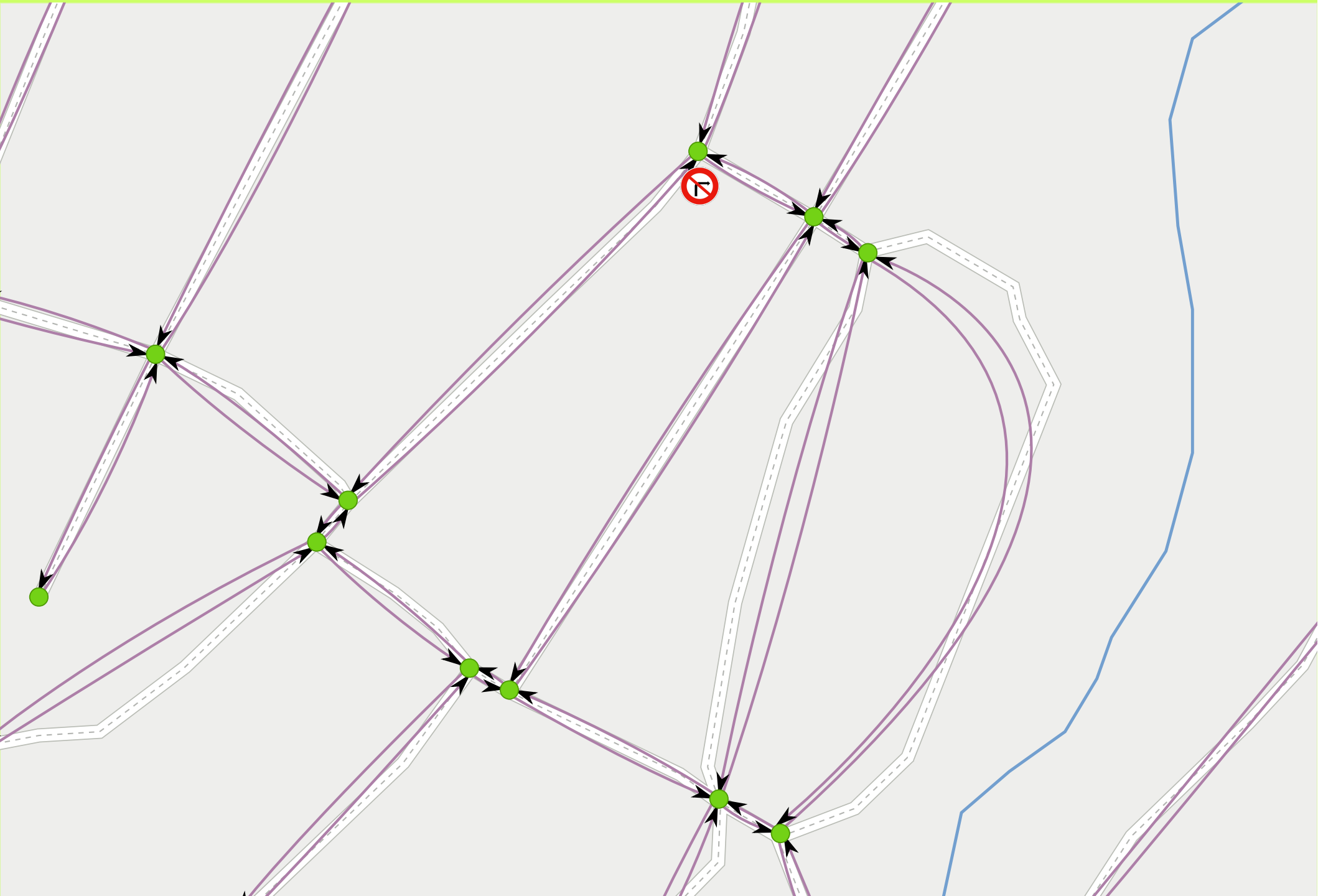
return **line graph**



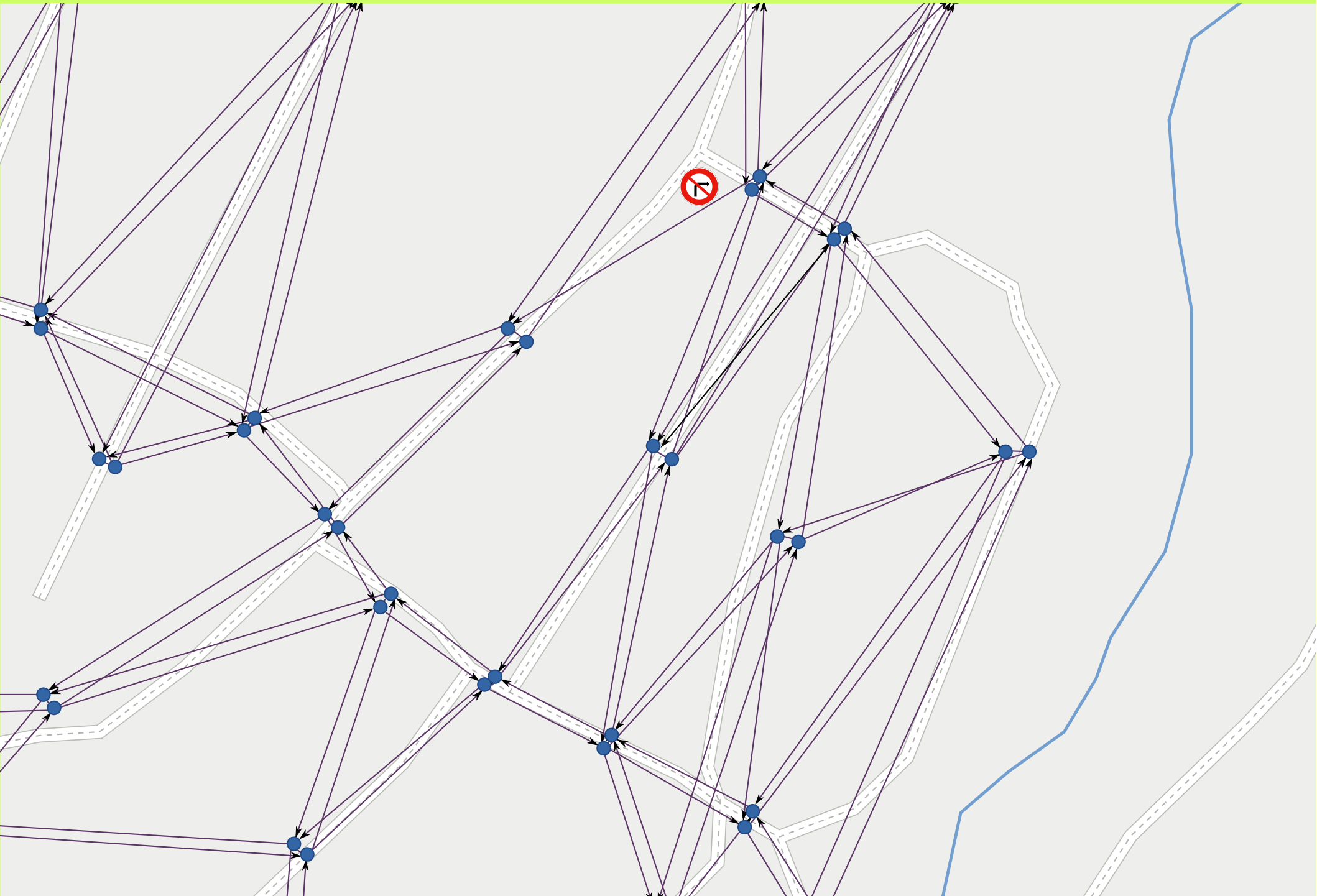








2. problem statement > applying restrictions > line graph



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:

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:

behavior

or ***test***

driven *development*

(BDD/TDD)

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

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

```cpp
#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

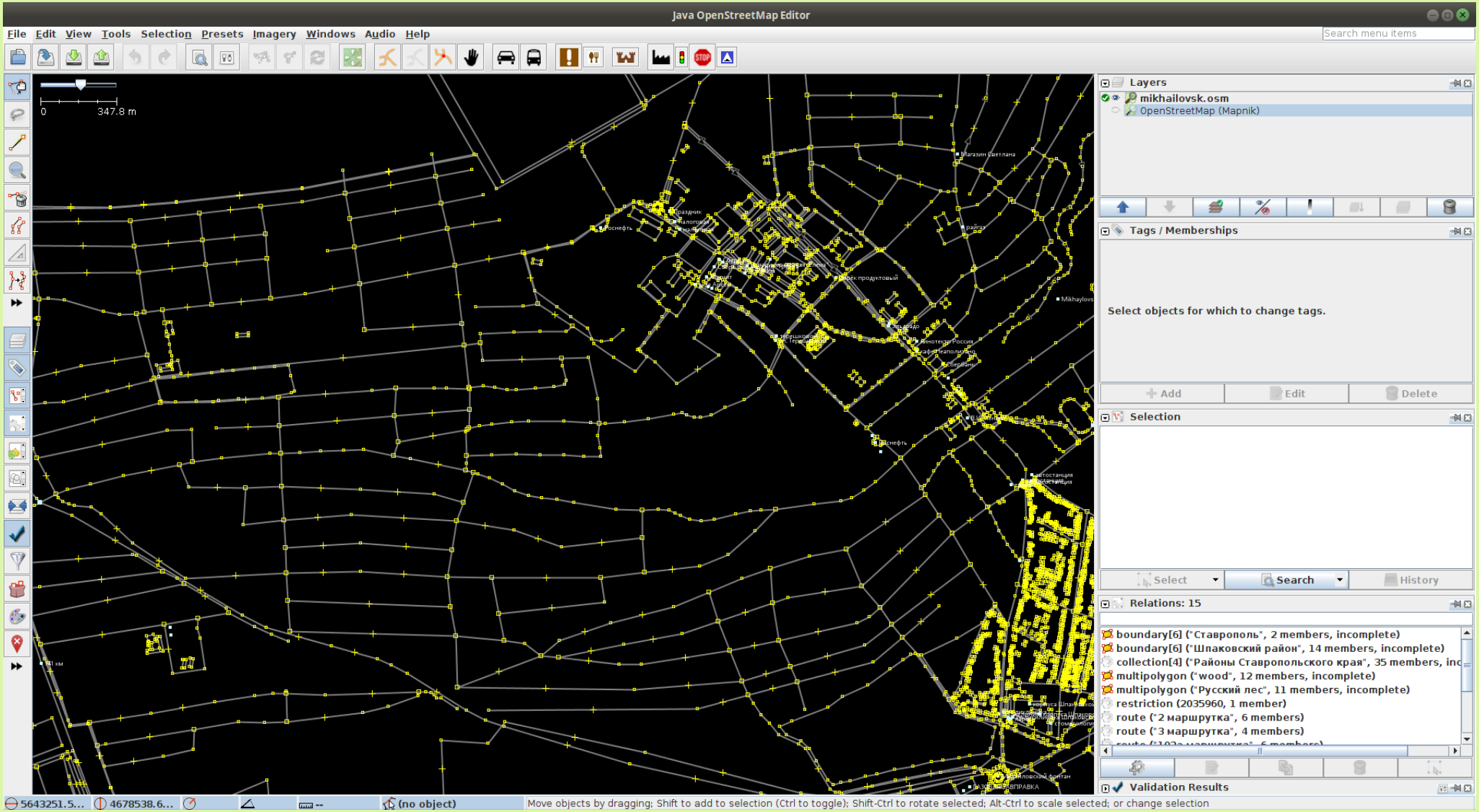
```
struct City {
  string      name;
  int         population;
  vector<int> zipcodes;
};

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

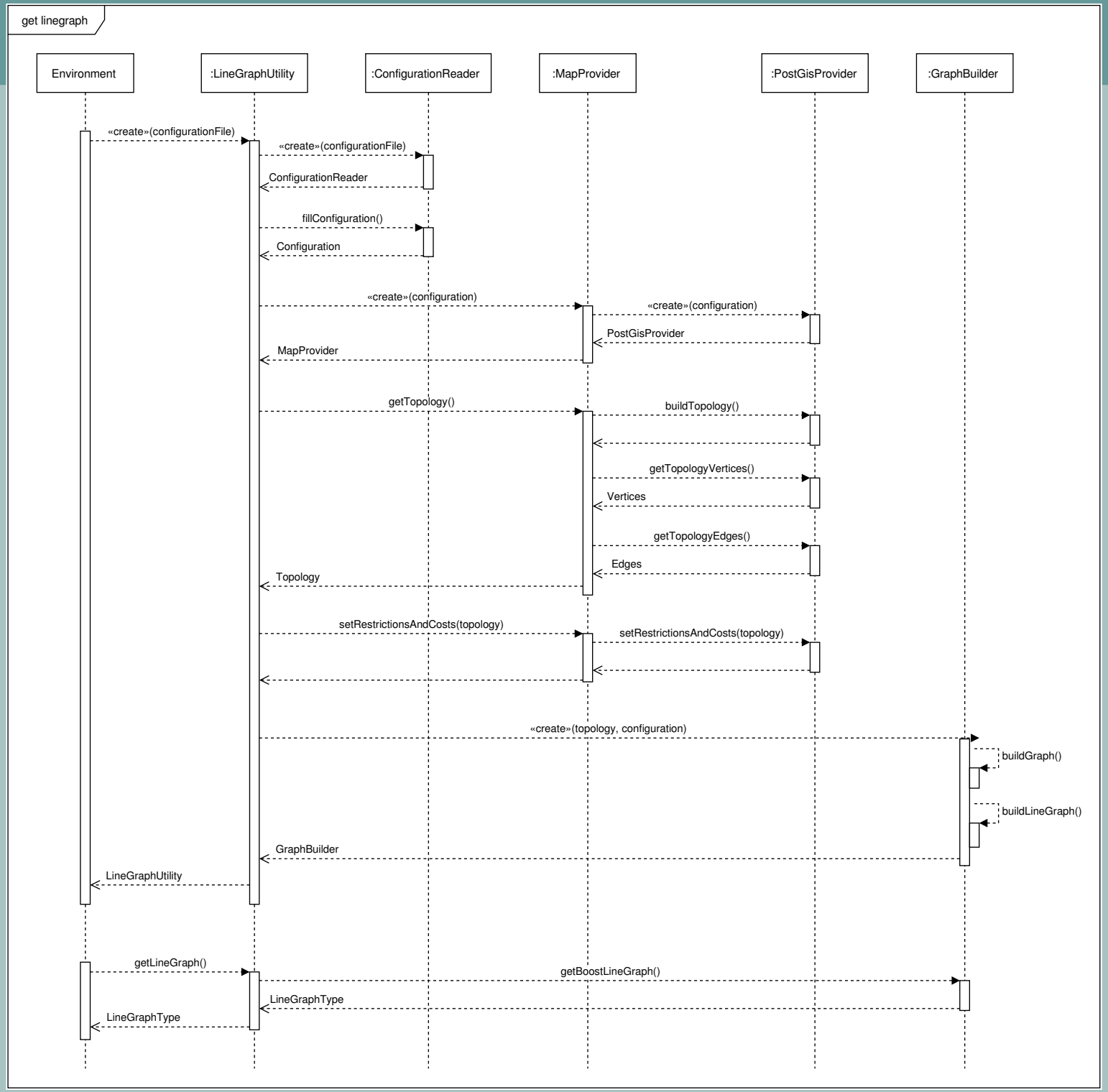
JOSM



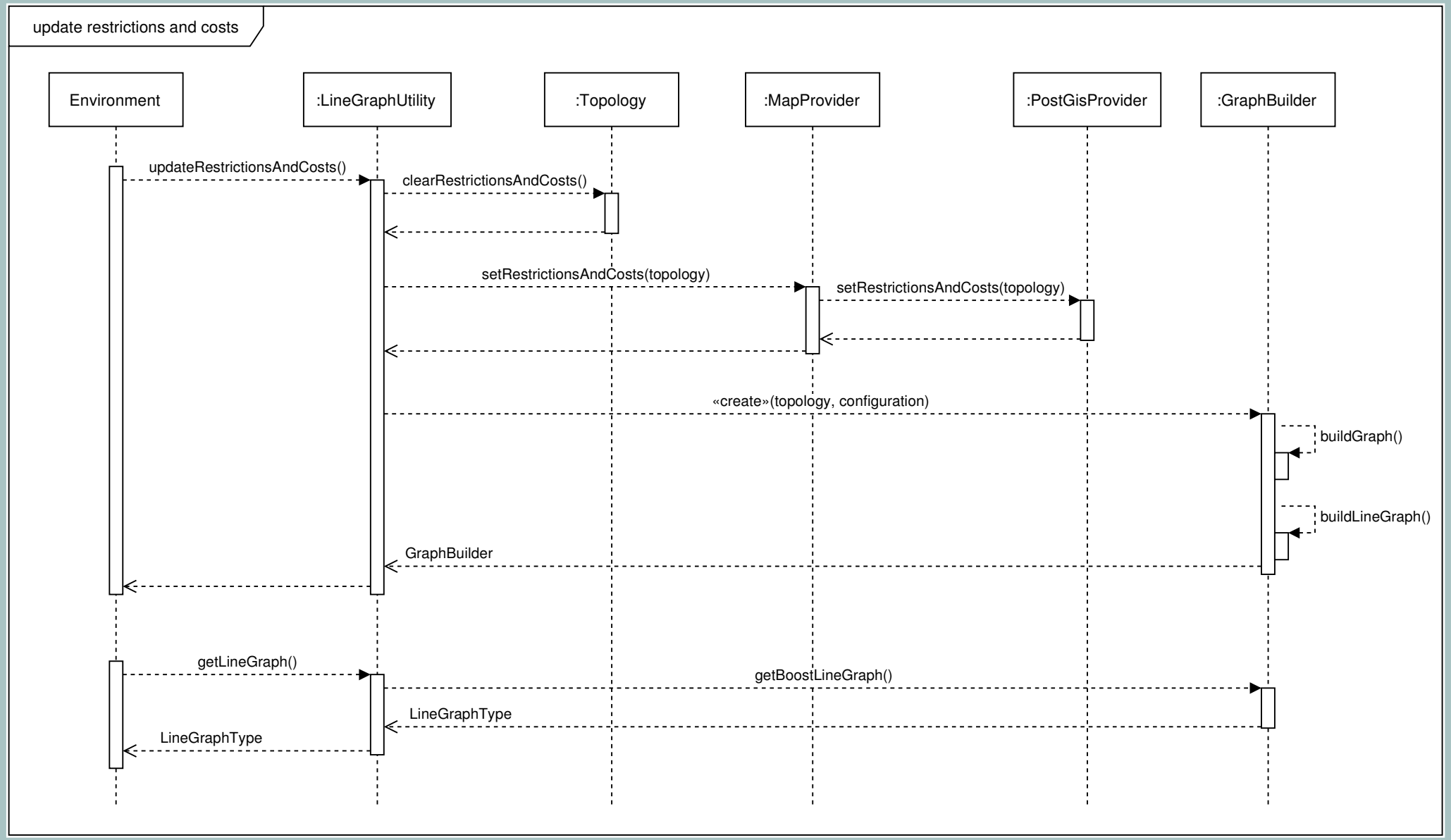

4
results

4. results >

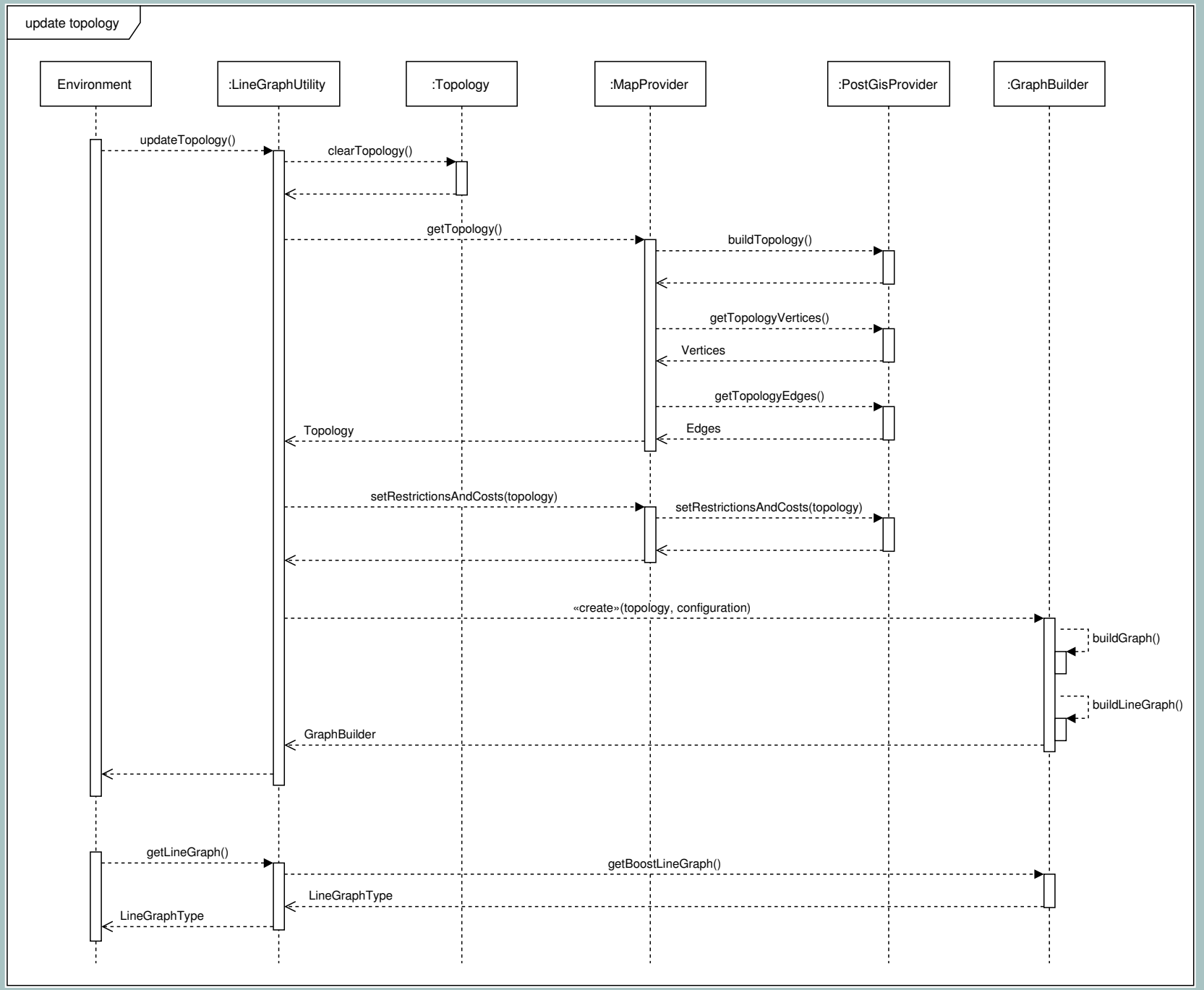
design



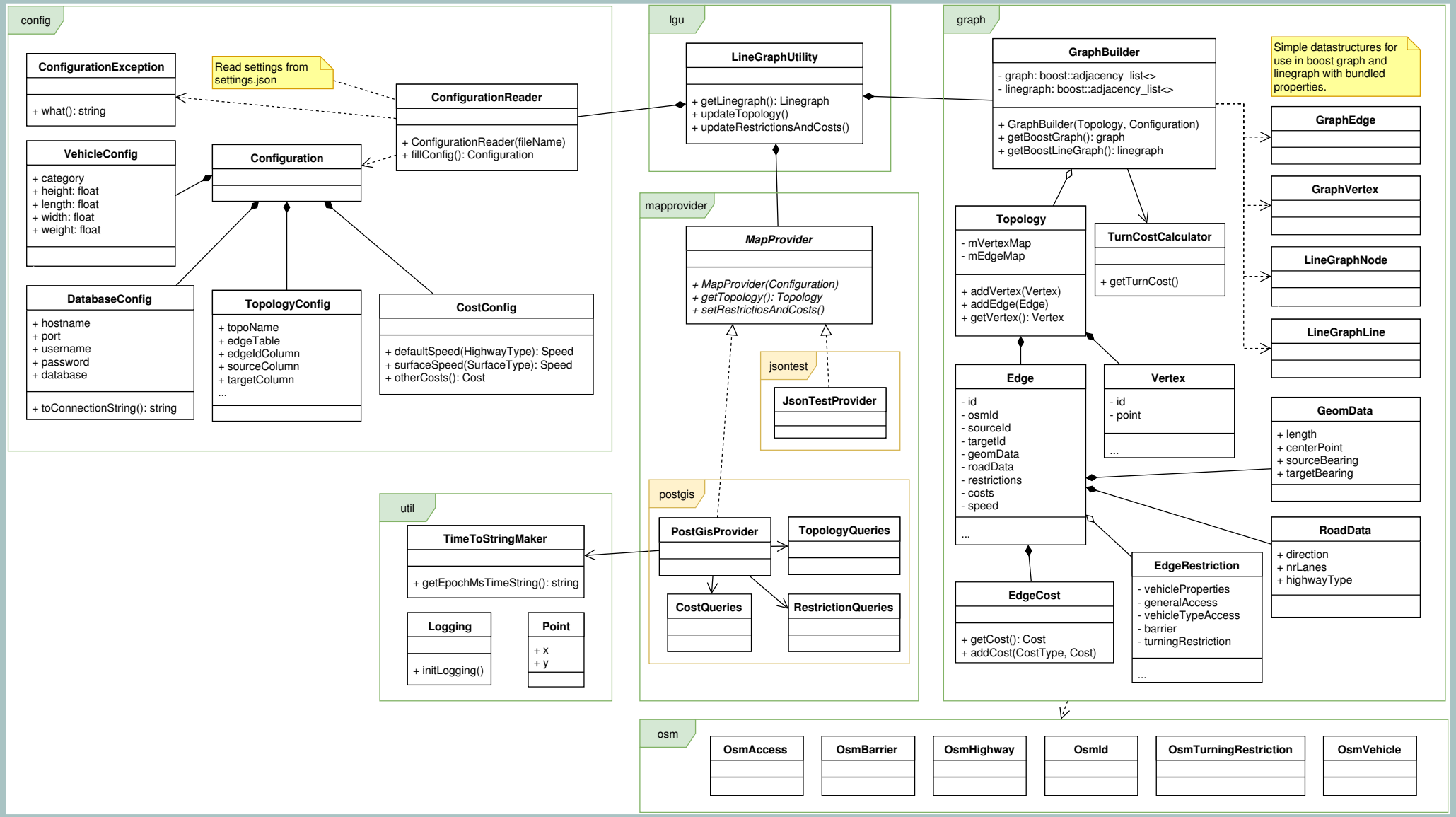
4. results > design



4. results > design



4. results > design



tests

specification **fulfillment**

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
```

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

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

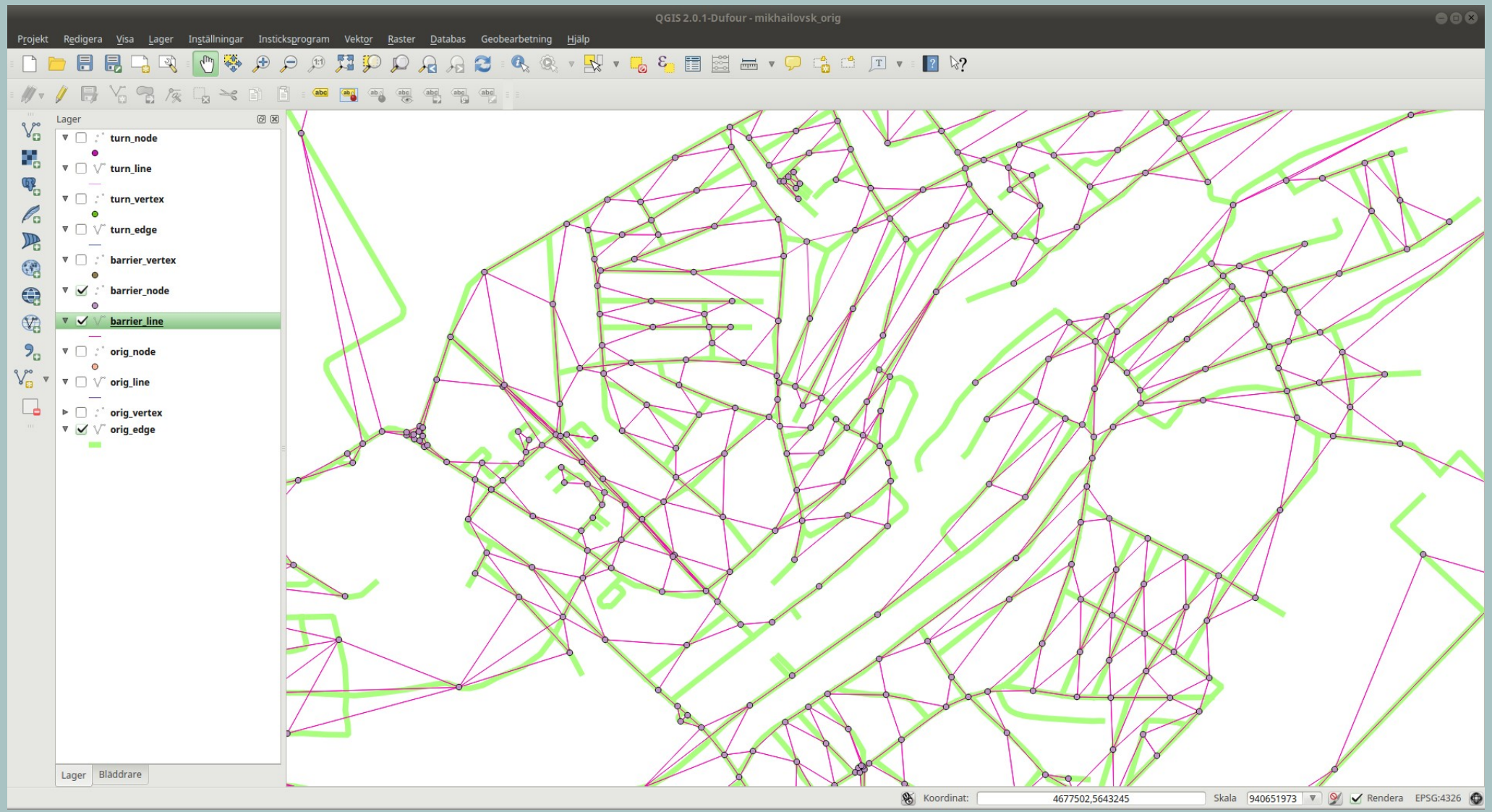
time of day, inclination ...

tests

performance

visual examination

4. results



QGIS

visual examination

5 conclusions

working but ***incomplete***

due to *time*

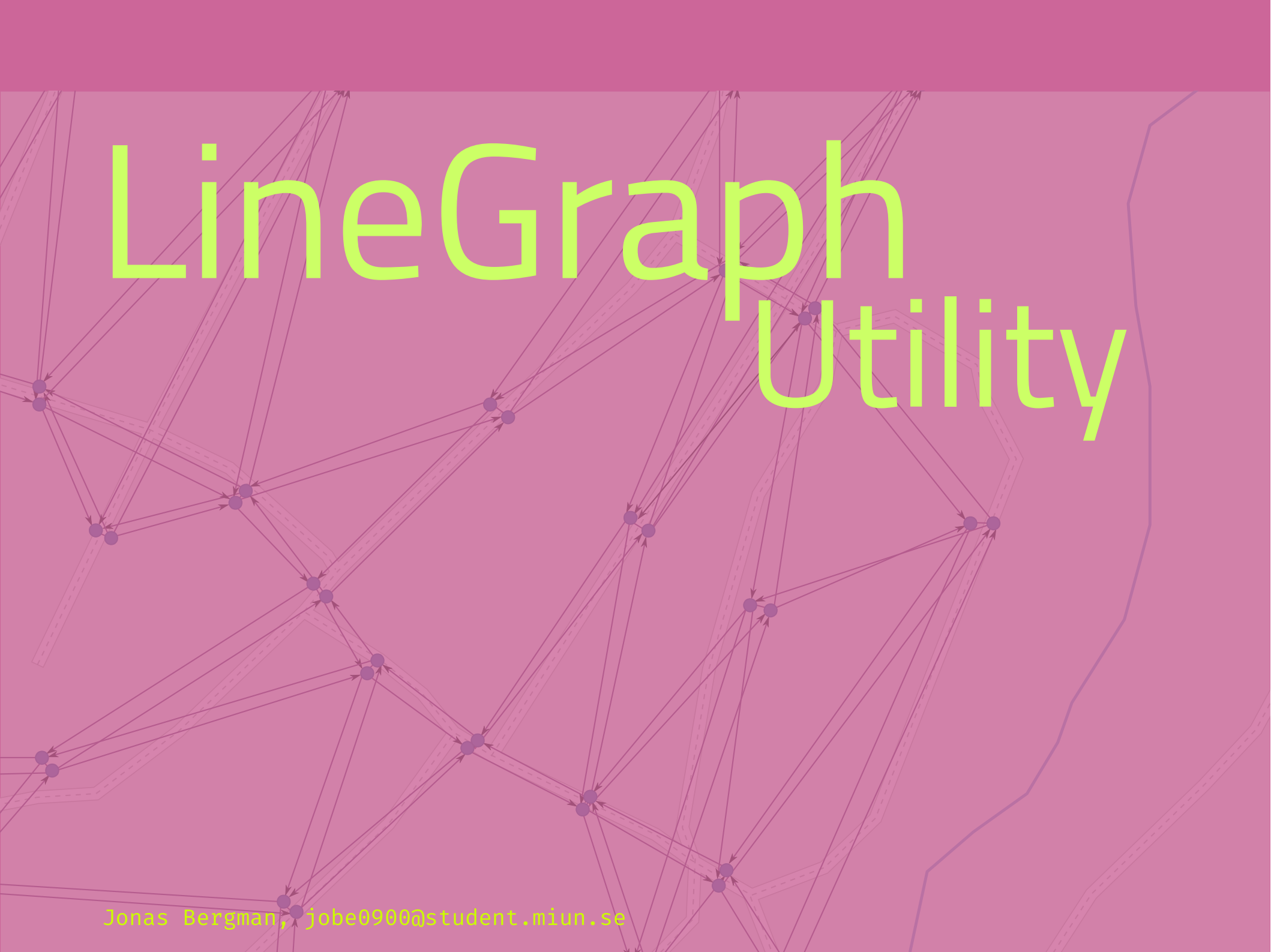
late *specification*

complex *restrictions*


6
future
work

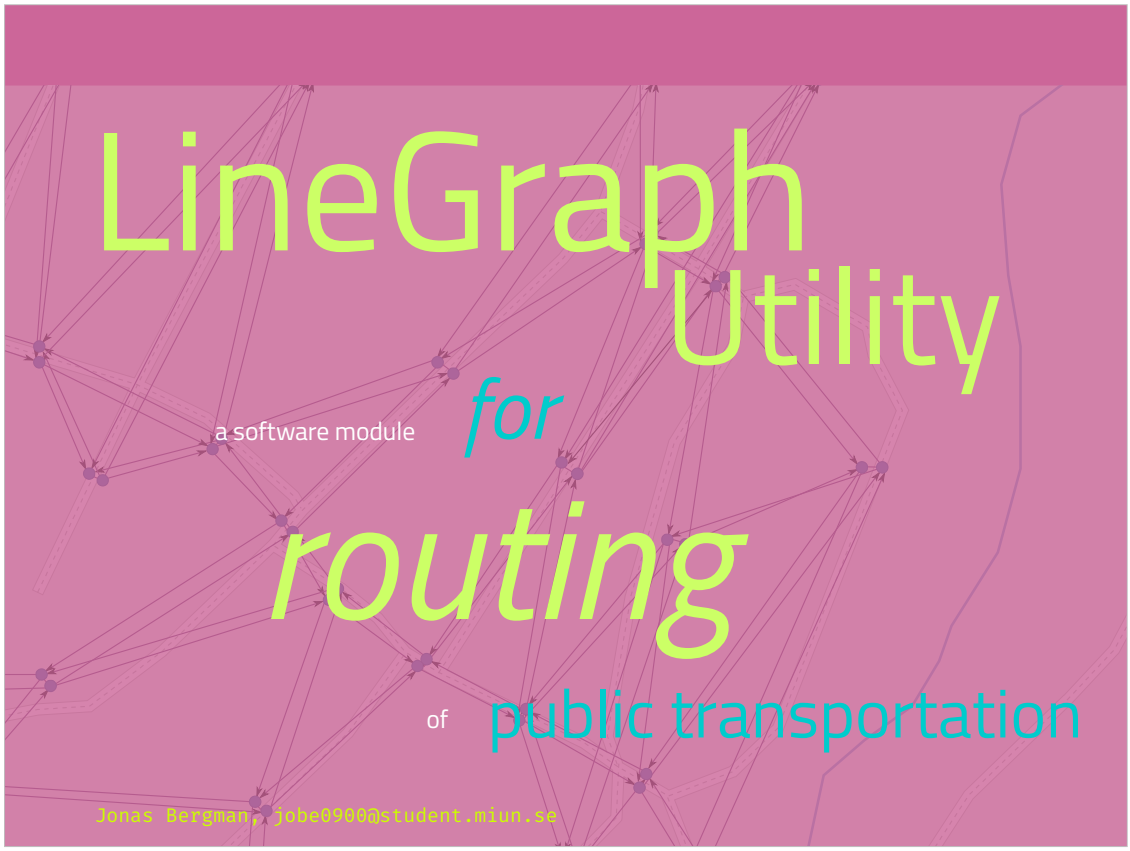
re-model restrictions

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



LineGraph Utility

Jonas Bergman, jobe0900@student.miun.se

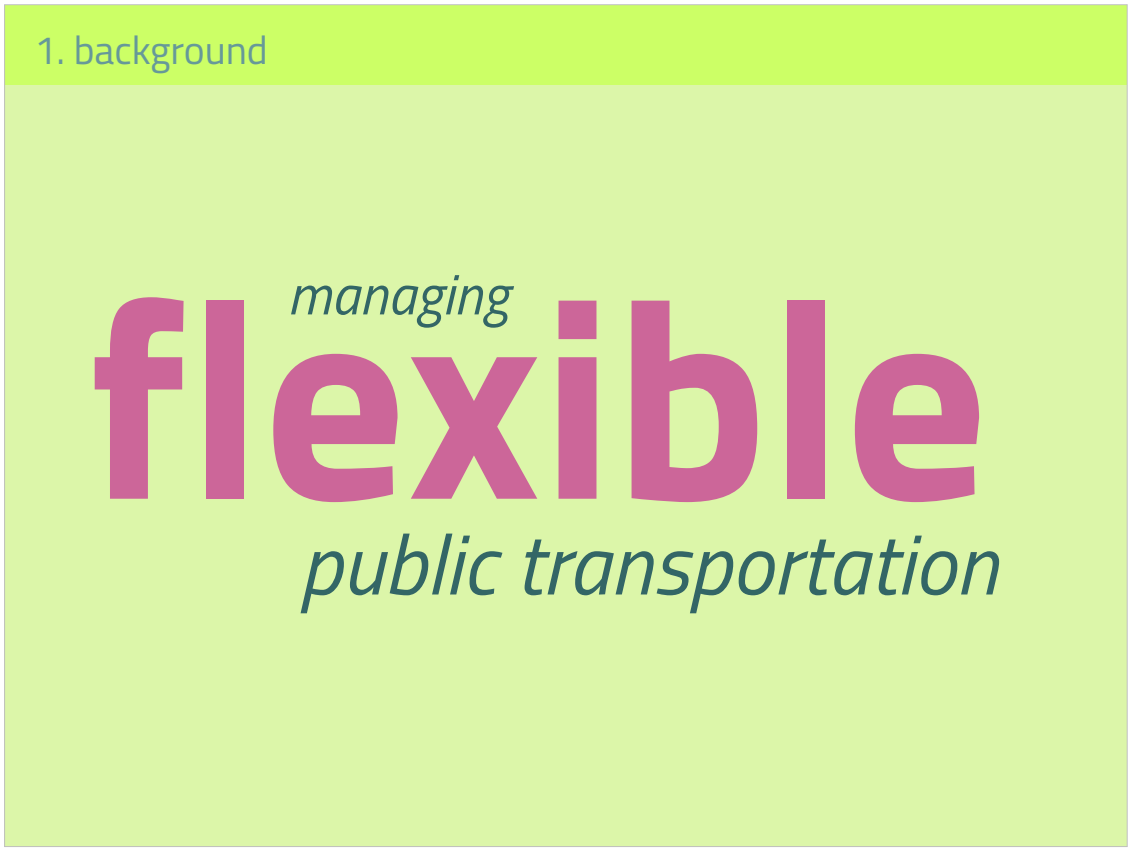


This is a module,
I have no overview of complete project.

Aim: (I think)
No waiting at bus stops
Vehicles gets directed to customers
Drivers needs real-time directions
Updated with current traffic situation


1
background

1. background



Flexible:

- no timetables
- customers ask for ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation:

no *timetables*

Flexible:

- no timetables
- customers ask for ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation:
- no timetables

commission

rides

Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation:

- no timetables
- commission rides

calculate *routes*

Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation:

- no timetables
- commission rides
- calculate routes

update driving **instructions**

Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation:

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

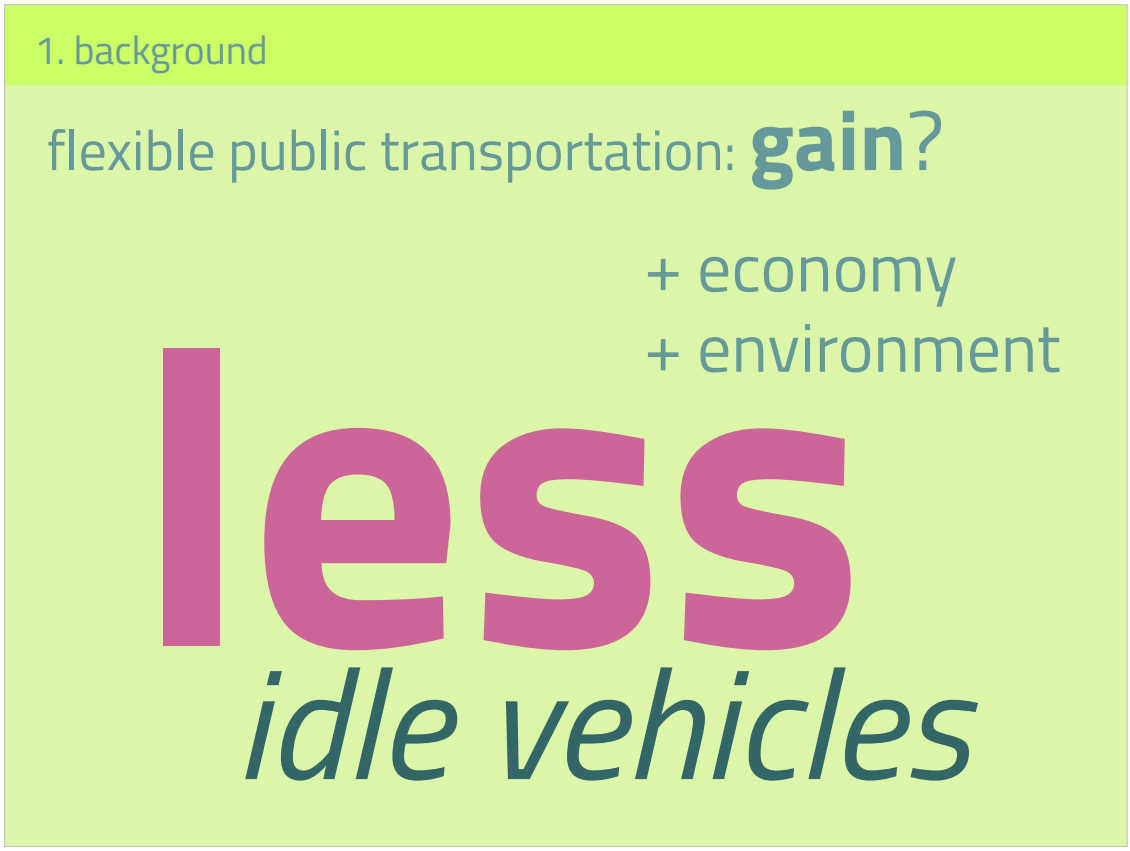
gain?

Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting



Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

1. background

flexible public transportation: **gain**?

- less idle vehicles

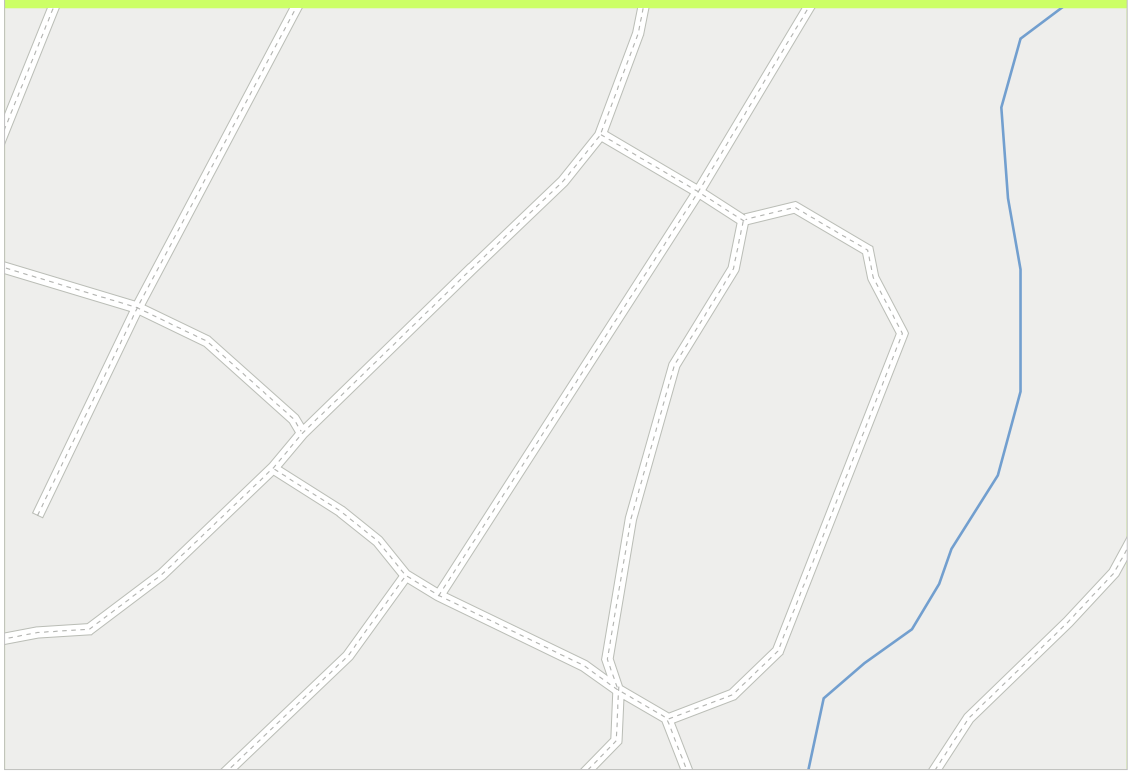
less *waiting*

Flexible:

- no timetables
- customers order ride via app or phone
- System calculates best routes
- Drivers gets real-time instructions

Gain:

- Company / World
  - Efficient use of fleet (less idle vehicles)
    - Environment++
    - Economy
- Customers
  - Less waiting

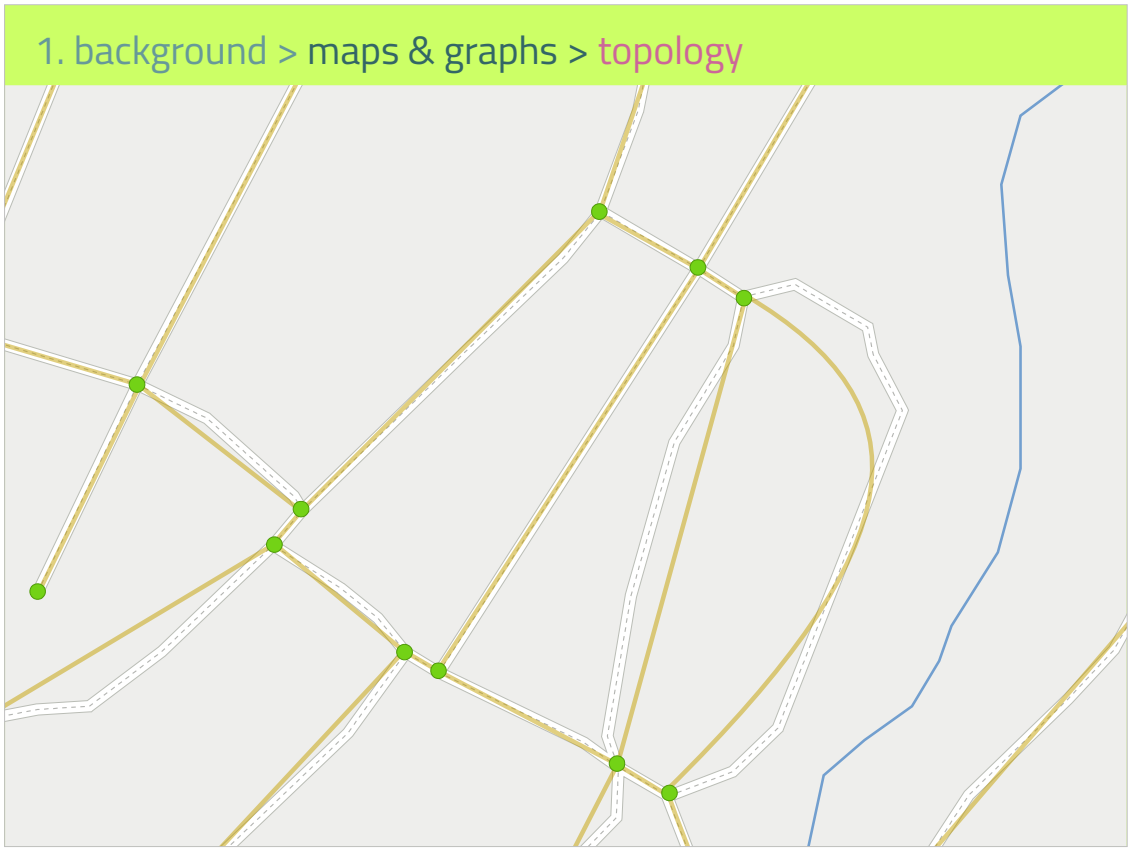


Before moving on:

About MAPS & GRAPHS

Map:

Projection of reality on 2D-surface. Geometry.
Semantics about connection of roads.



When routing:

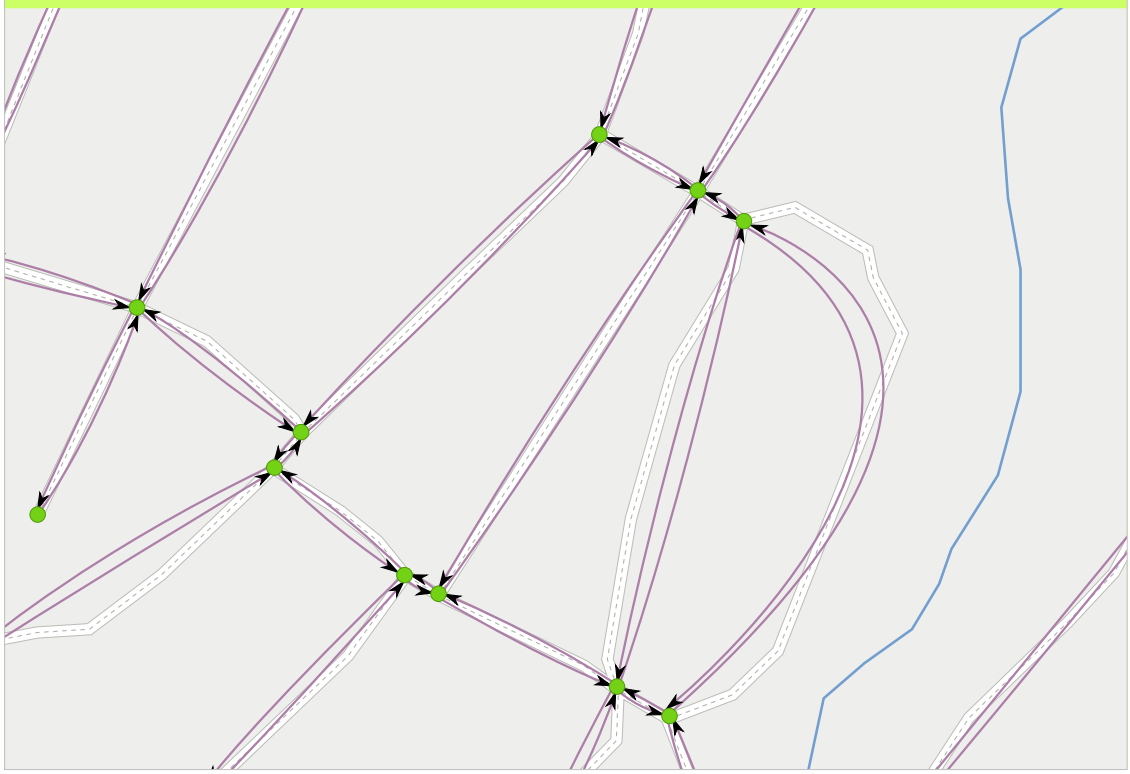
- Interesting:
  - Connections
  - Relations
- Not:
  - Geometry

TOPOLOGY

- Undirected graph
- Static as road network is stable

Terminology

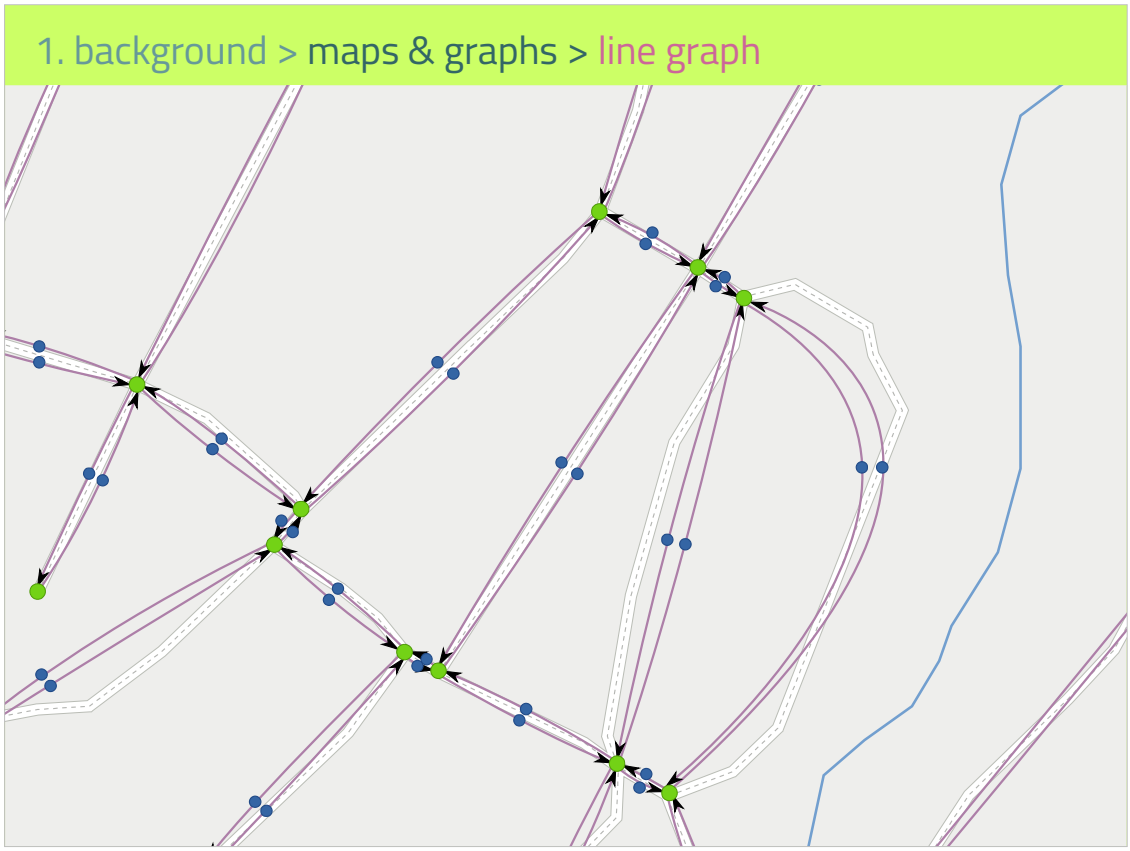
- Vertex = node, point, dot
- Edge = arc, line



To be able to apply dynamic restrictions to the static topology we construct a DIRECTED graph.

Specifies possible travel on roads in the map.

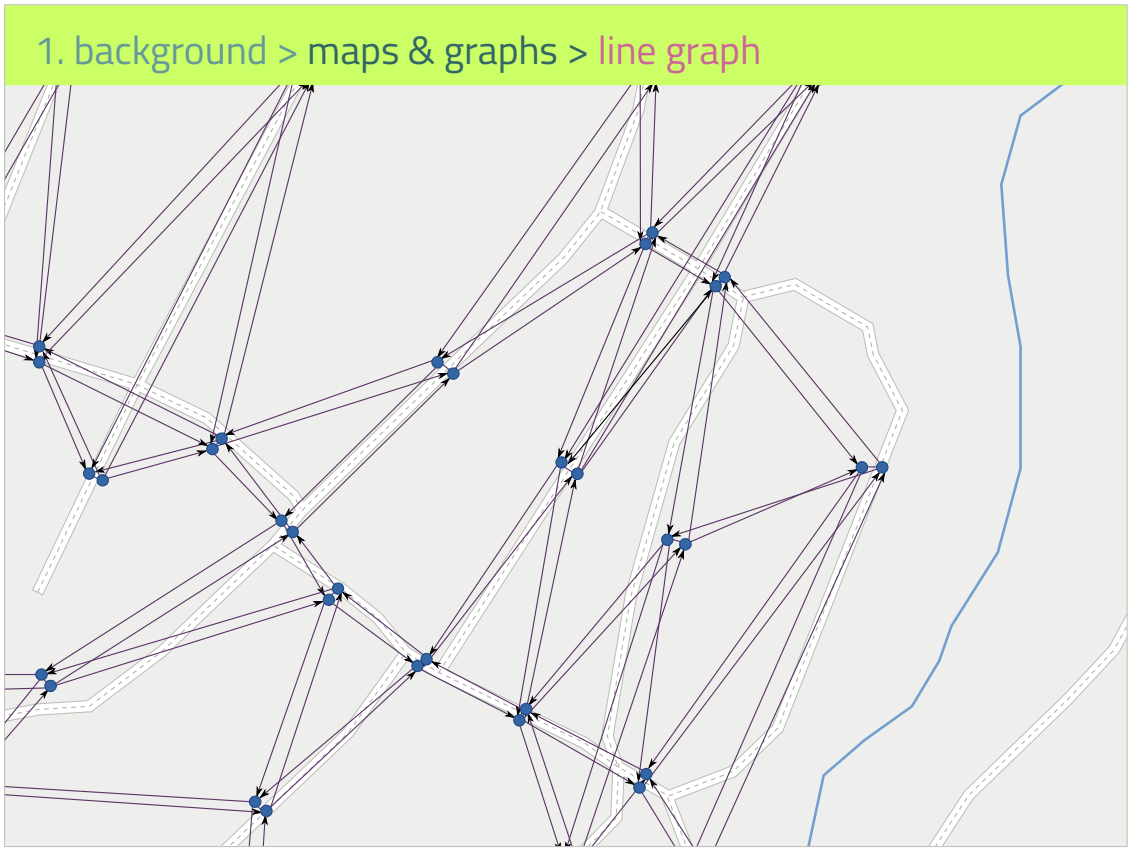
Each lane becomes an edge in correct direction.
One-way road = edge in one direction.
Several lanes = several parallel edges.



Last step: transform directed graph to LINE graph.

Specifies possible turns in the map.

Edge -> Node



Nodes connected with Lines where travel is allowed.

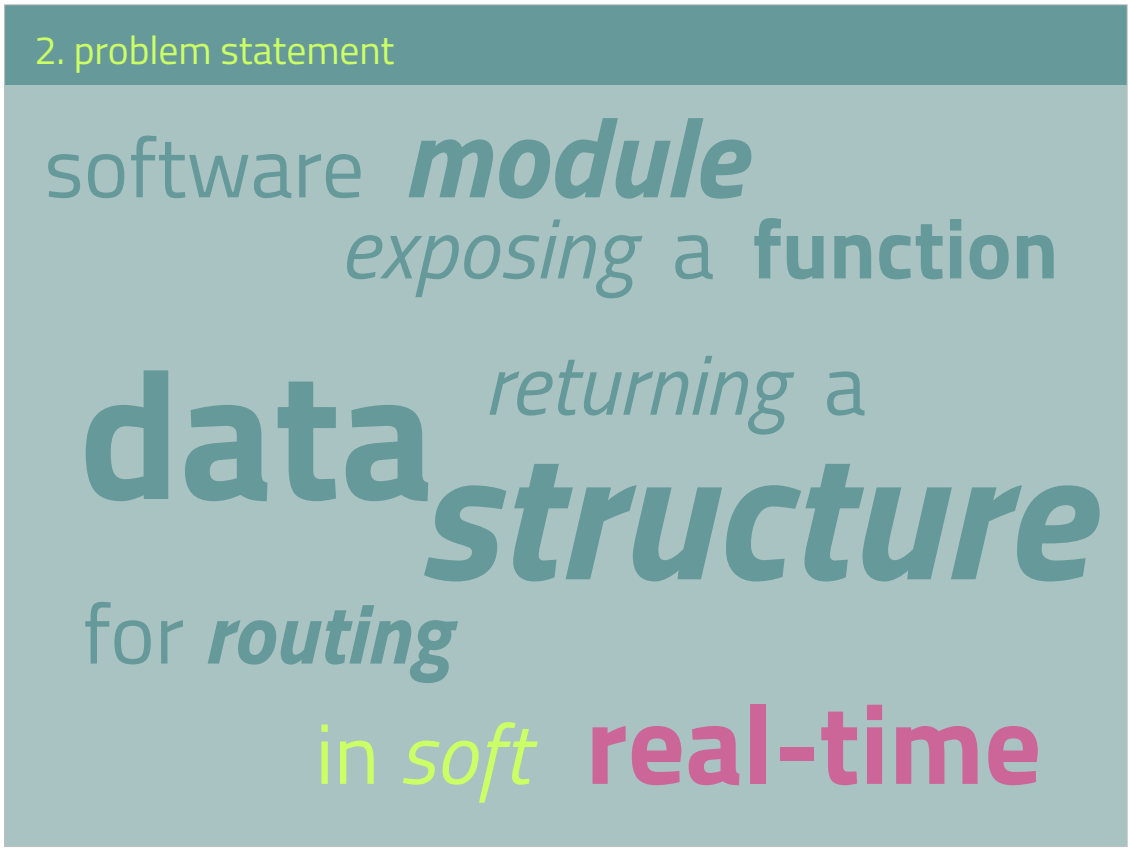
NOTE terminology:

Node + Line

Instead of

Vertex+ Edge


2
problem
statement



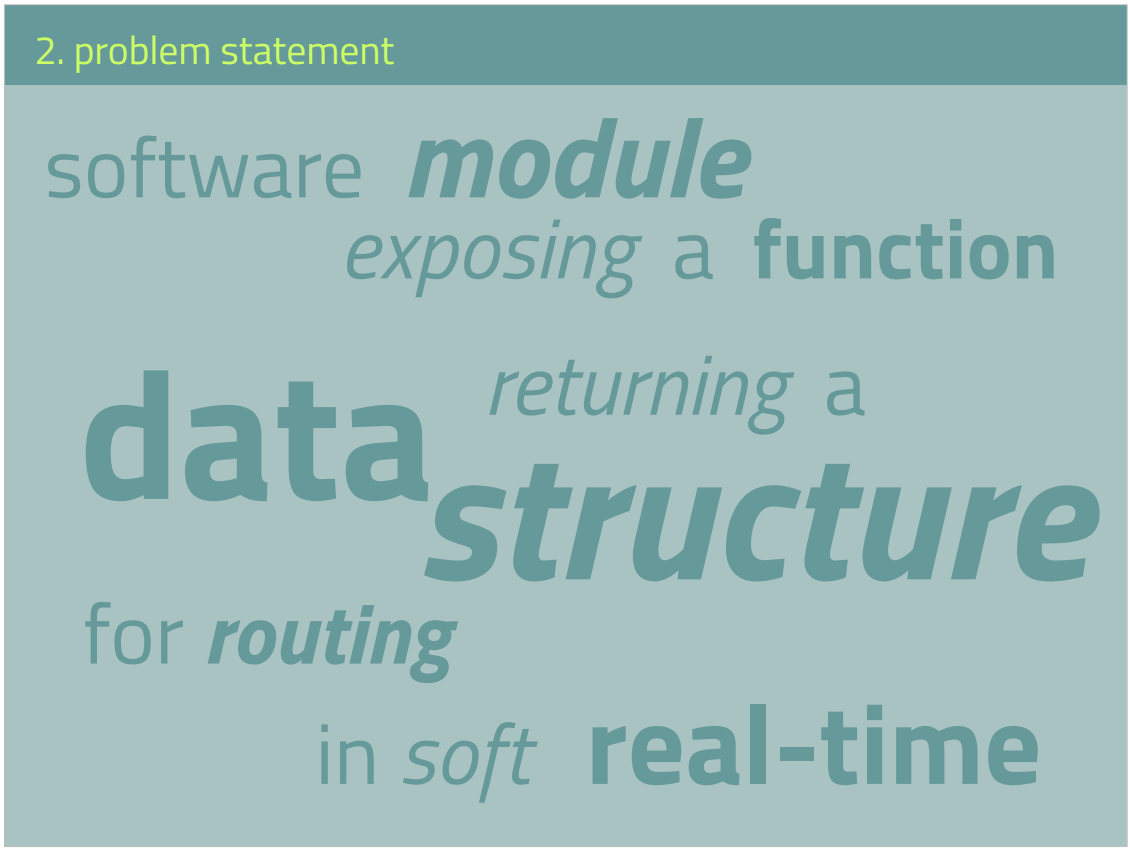
Software module exposing function, returning data structure for routing in soft real-time.

Load map data.
- Build topology

Consider restrictions and conditions.
- Directed graph.
- Restrictions.

Return data structure for routing decisions.
- LineGraph.

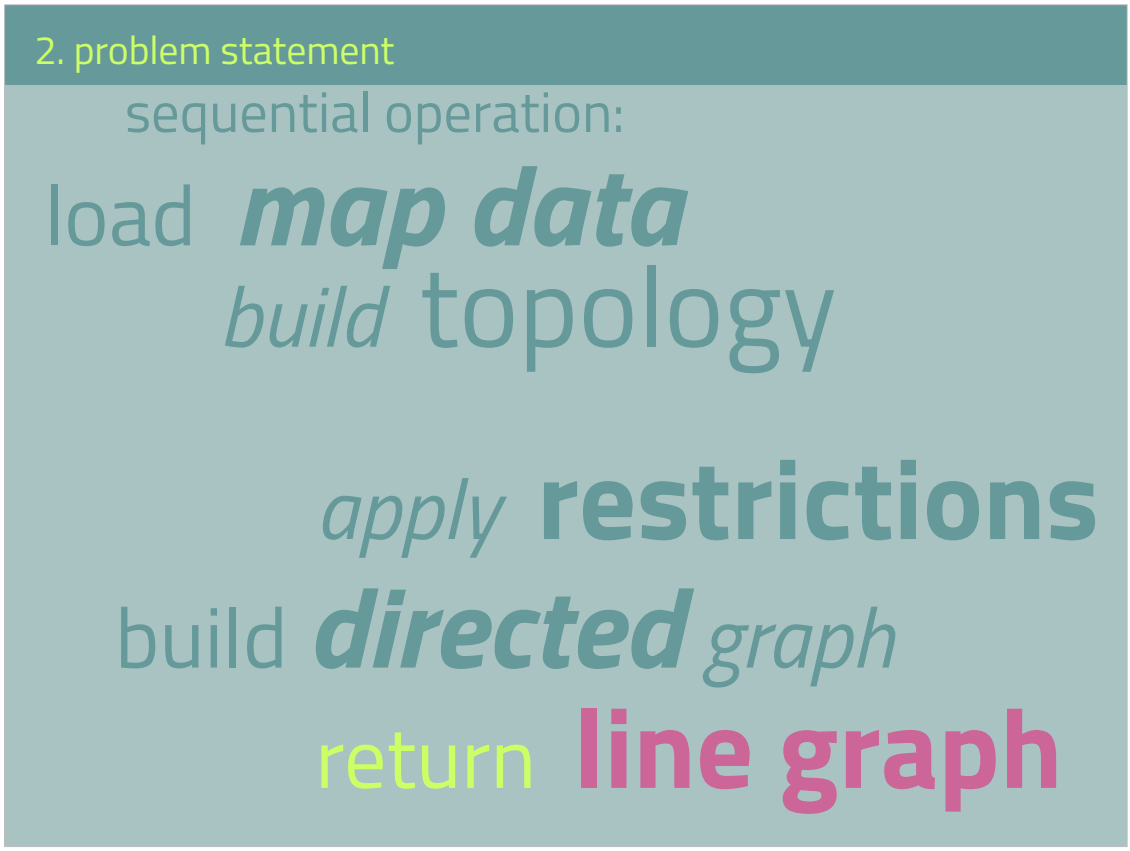
Given set of tools:
- C++, Boost Graph Library,
- OpenStreetMap (crowd sourced, open).
- PostGis (PostgreSQL + spatial data).
- BDD/TDD



Software module exposing function, returning data structure for routing in soft real-time.

Load map data.
    Build topology
Consider restrictions and conditions.
    Directed graph.
    Restrictions.
Return data structure for routing decisions.
    LineGraph.

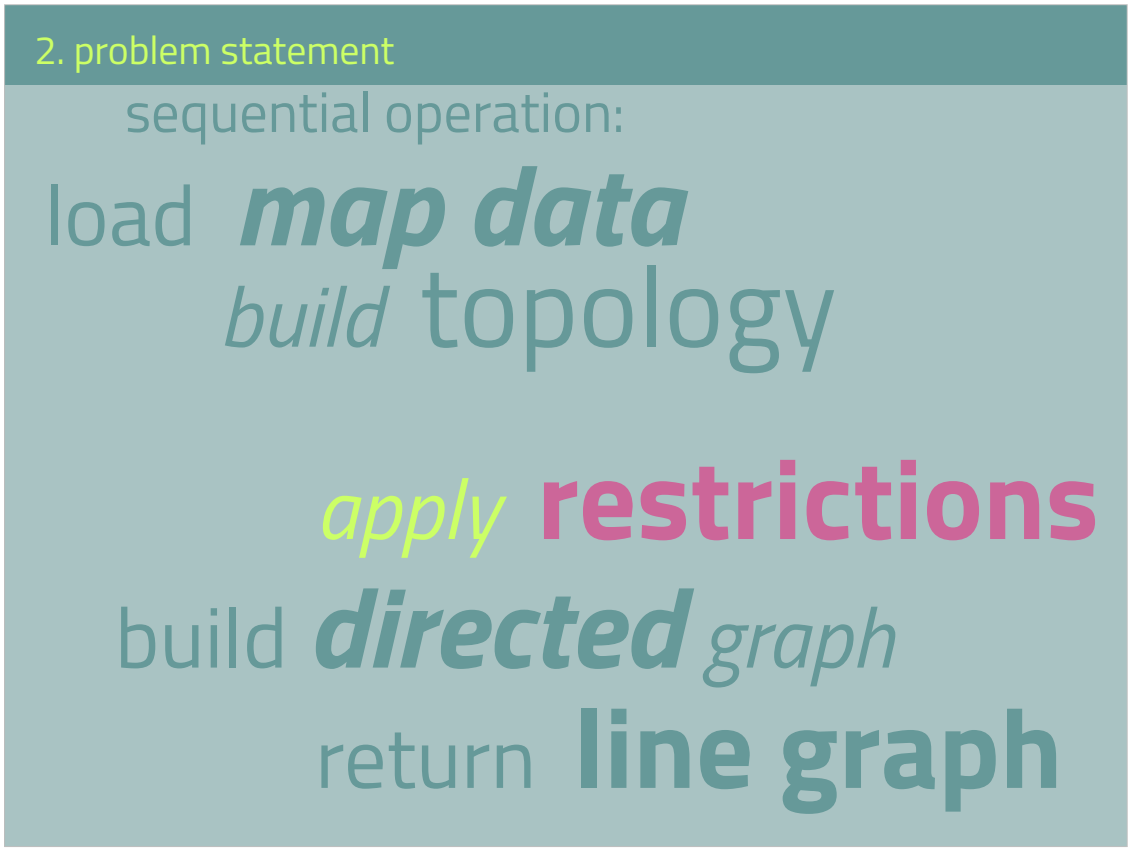
Configurable settings in JSON.



HOW:
Sequential operation:

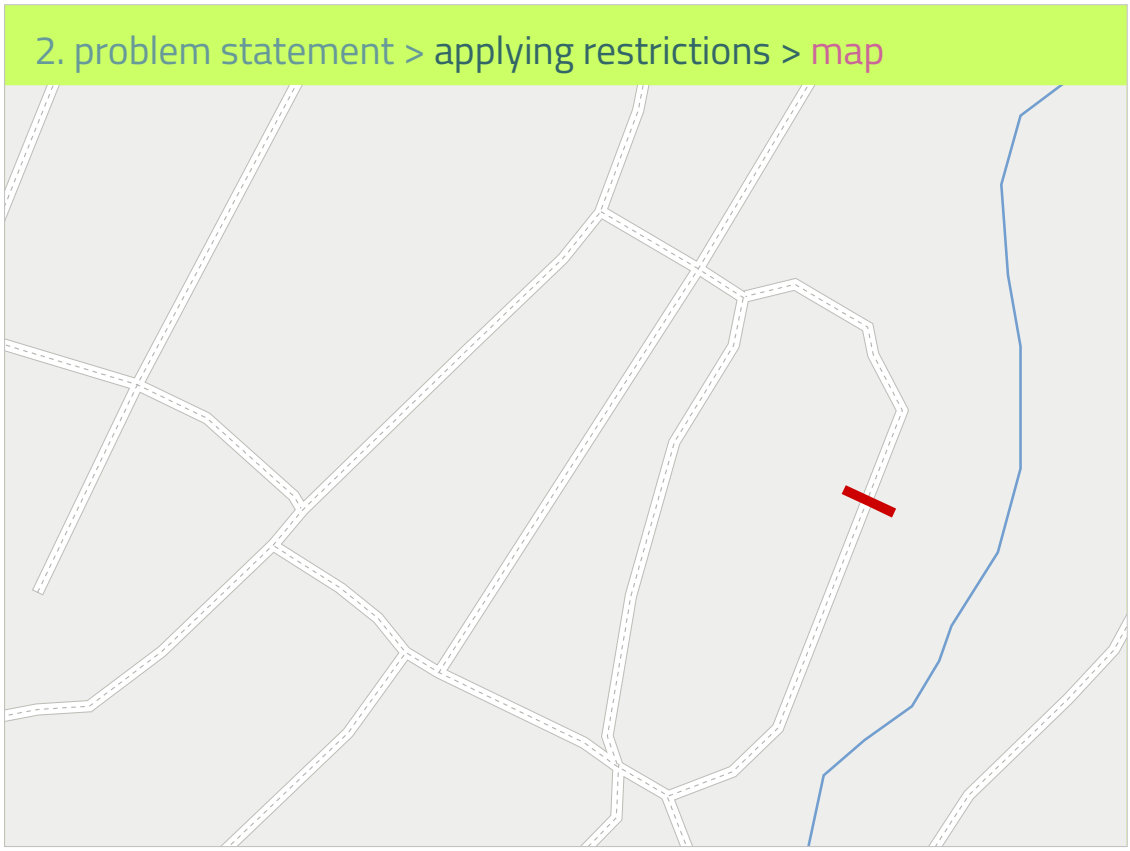
Preliminary step:
Load map data.
Build topology

On demand
Consider restrictions and conditions.
Directed graph.
Restrictions.
Return data structure for routing decisions.
LineGraph.
Calculate costs =
Travel time on from edge
(length + speed + surface) +
Restriction costs (speed bumps...) +
Turn costs (angle + vehicle props ...)



Diversion:

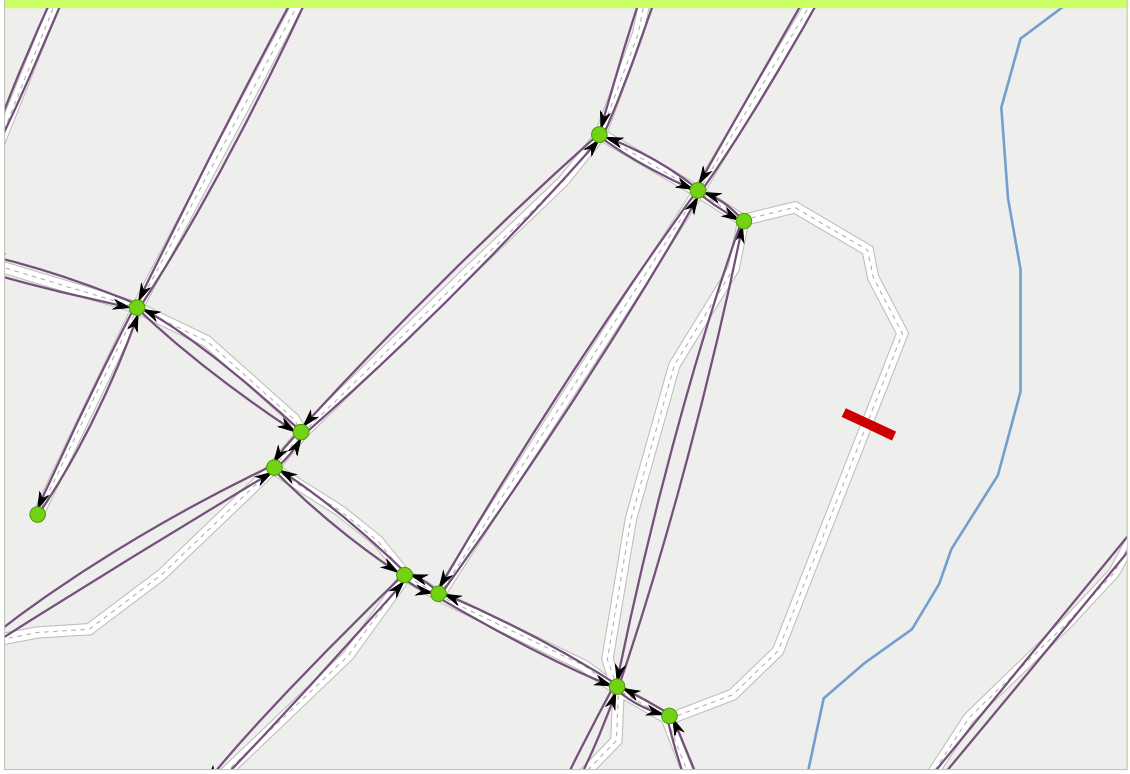
Applying restrictions to the graphs.



Dynamic restriction like road work = closed road.

Dynamic restriction like road work = closed road.

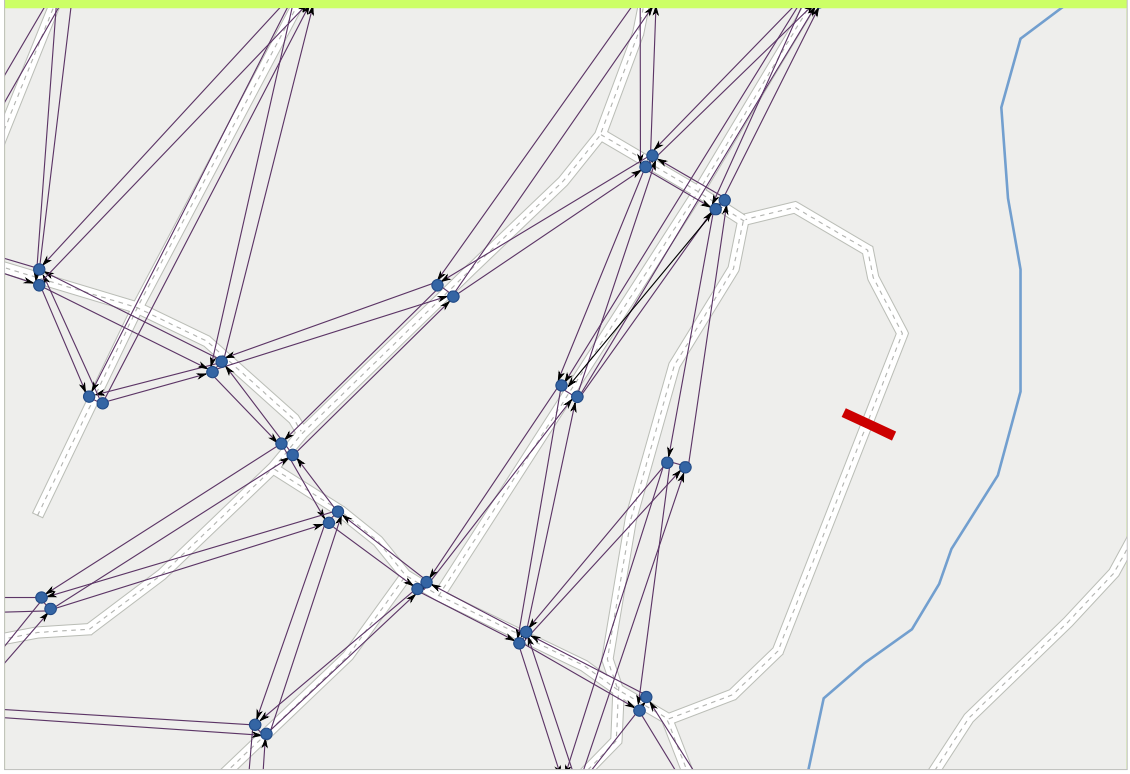
Topology not affected.



Dynamic restriction like road work = closed road.

Topology not affected

Directed graph = travel along roads → no edges.

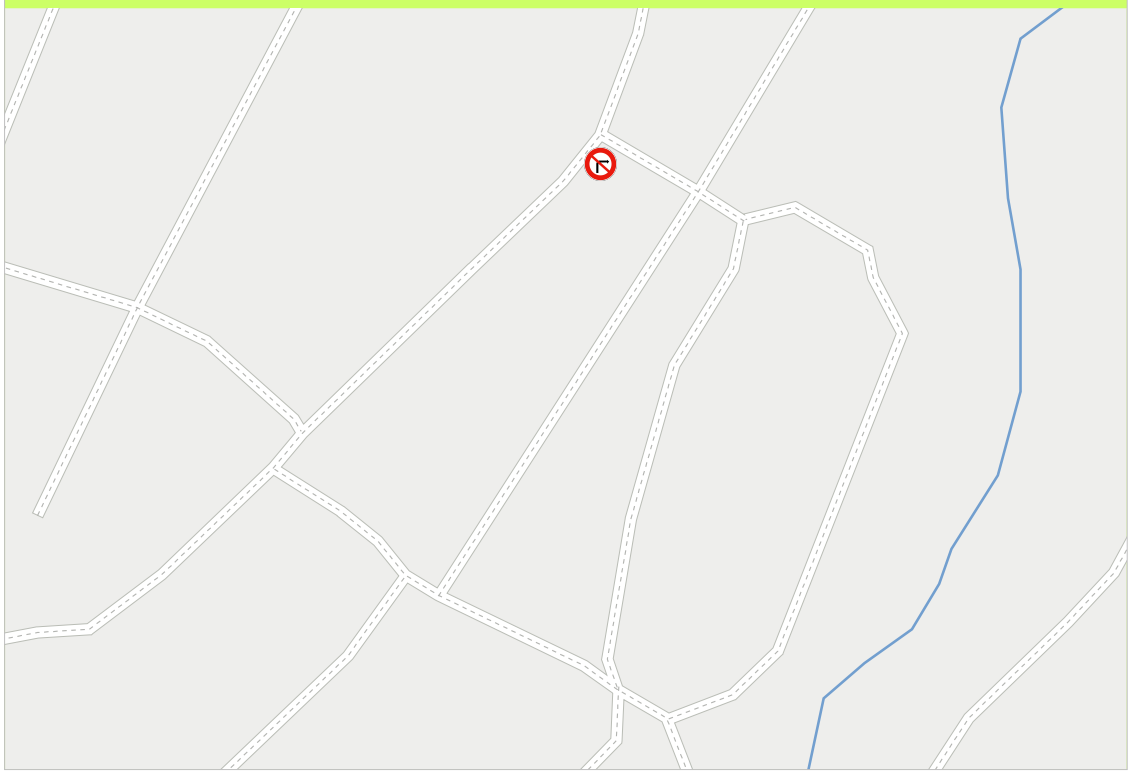


Dynamic restriction like road work = closed road.

Topology not affected

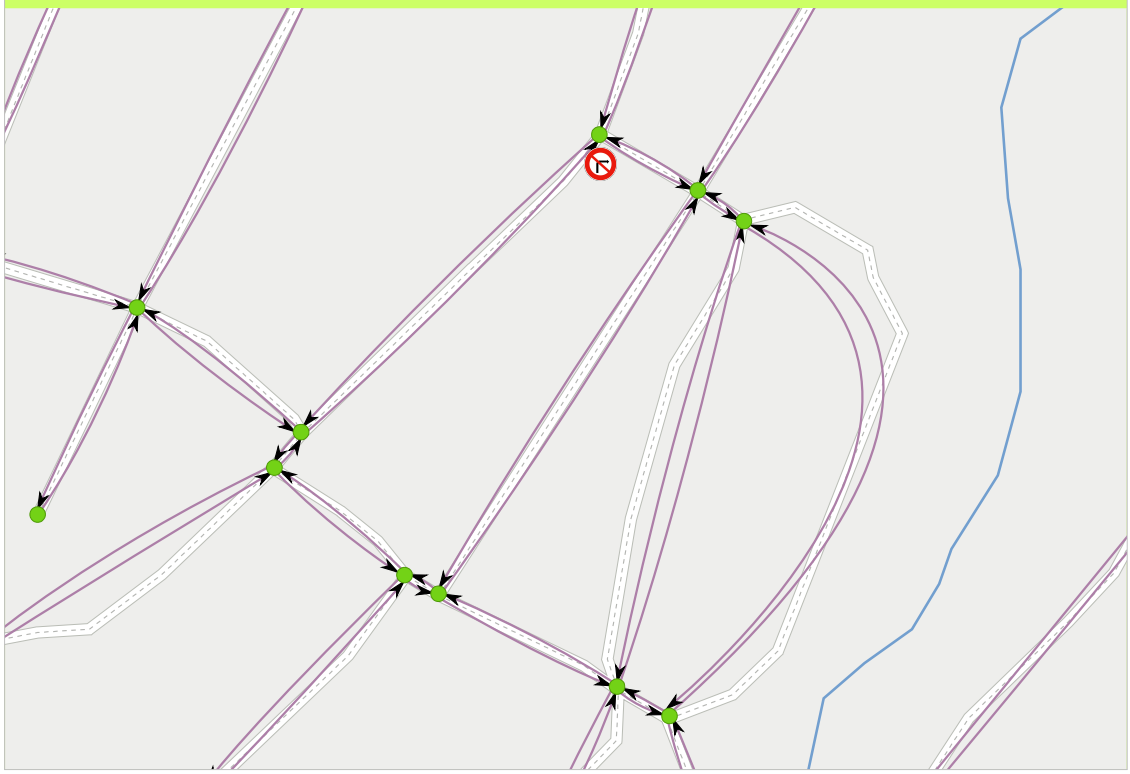
Directed graph = travel along roads → no edges.

Line graph = transformed graph.
No edges → no nodes → no lines



Static restriction like “No turn”

Topology not affected

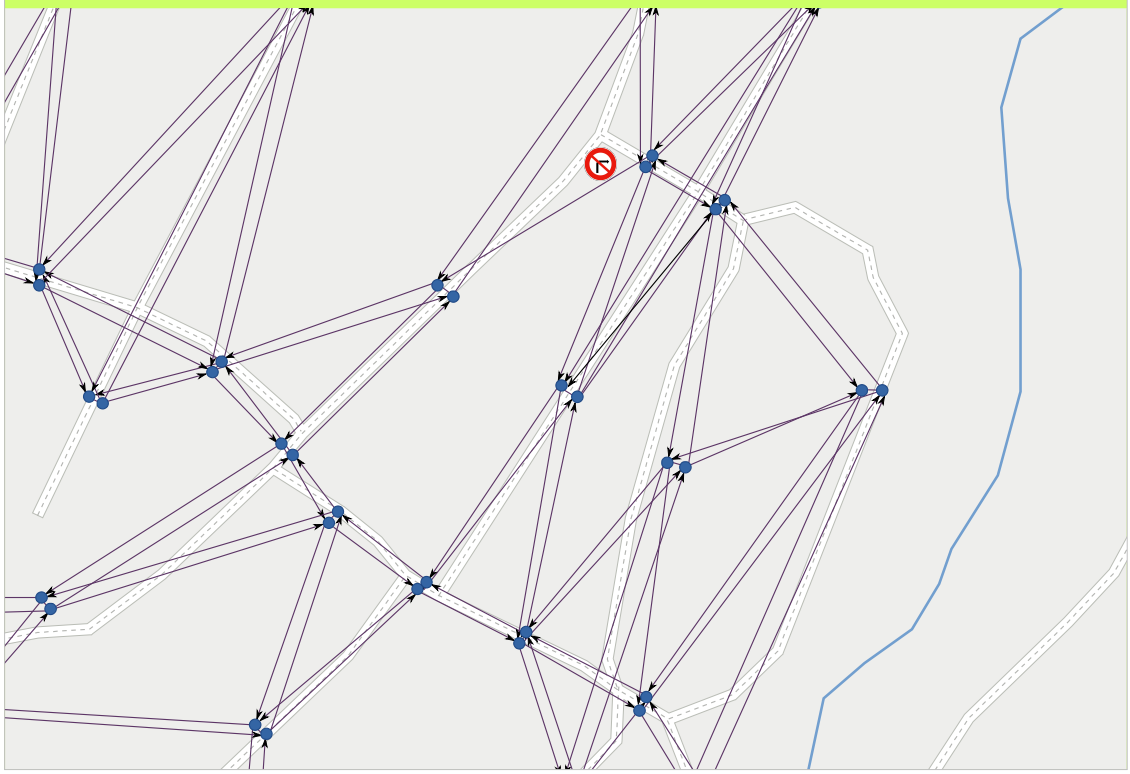


Static restriction like “No turn”

Topology not affected

Directed graph = travel along roads → not affected.

2. problem statement > applying restrictions > line graph



Static restriction like “No turn”

Topology not affected

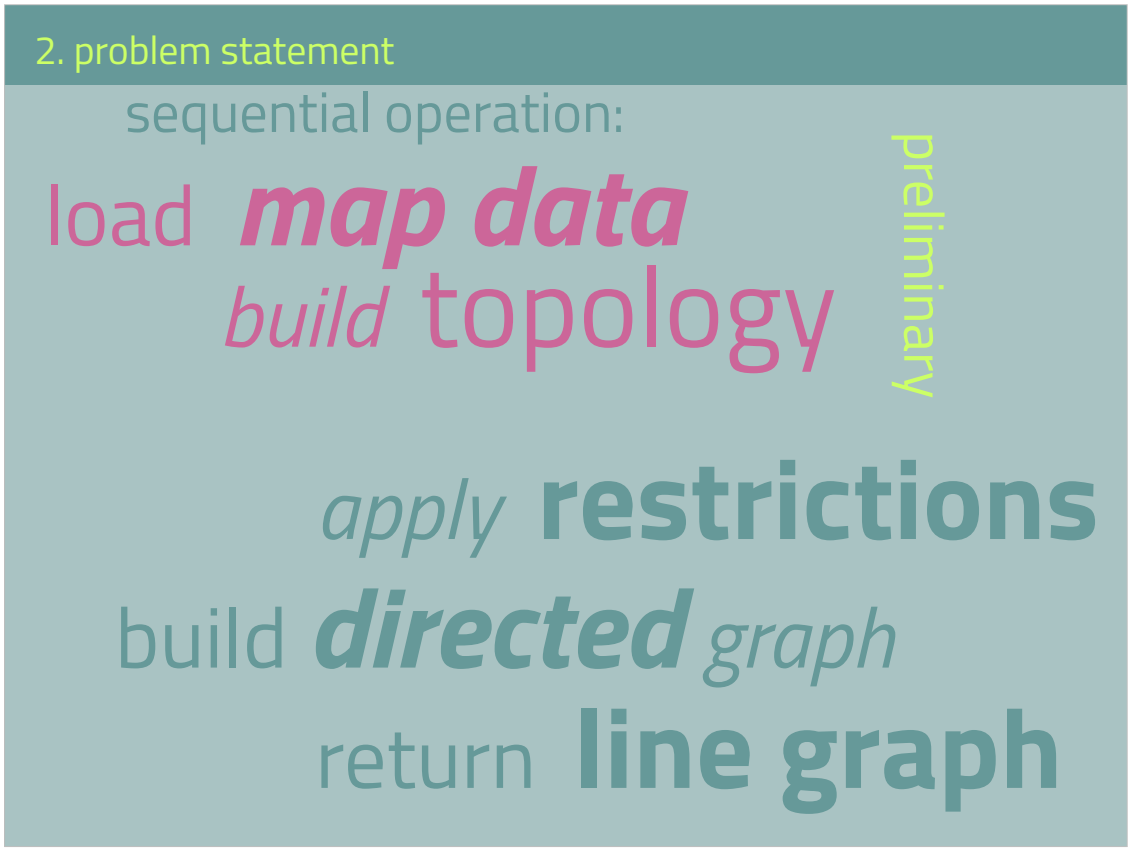
Directed graph = travel along roads → not affected.

Line graph = allowed turns.

Edges → nodes

BUT

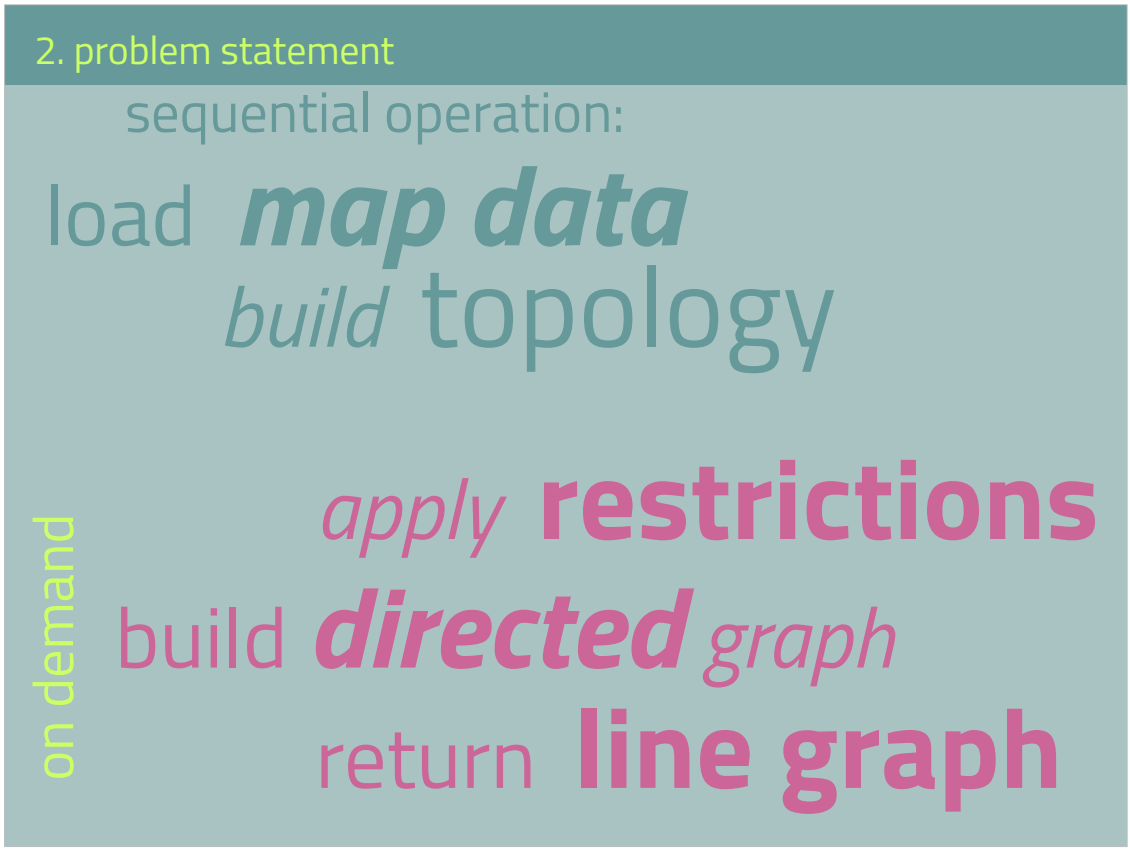
Not allowed turn → no line



HOW:
Sequential operation:

Preliminary step: (static road network)
Load map data.
Build topology

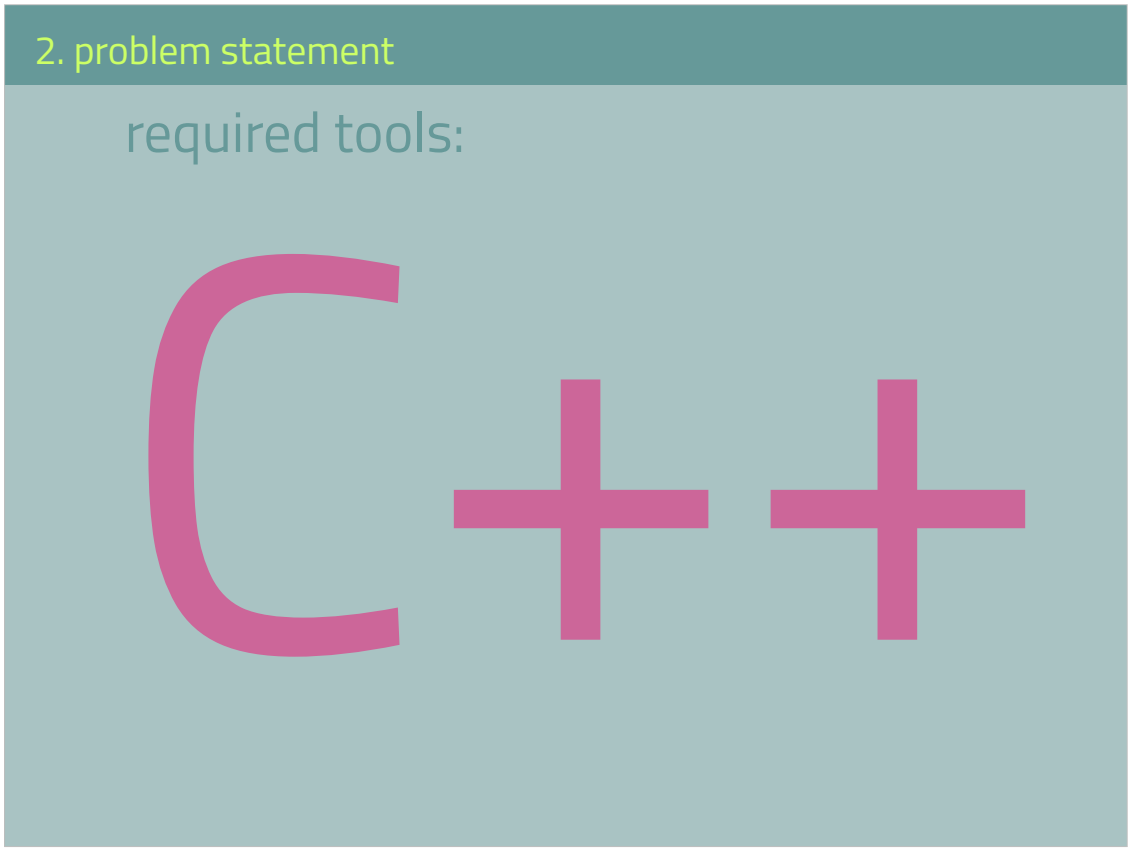
On demand: (dynamic)
Consider restrictions and conditions.
Directed graph.
Restrictions.
Return data structure for routing decisions.
LineGraph.



HOW:
Sequential operation:

Preliminary step: (static road network)
Load map data.
Build topology

On demand: (dynamic)
Consider restrictions and conditions.
Directed graph.
Restrictions.
Return data structure for routing decisions.
LineGraph.



Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).

2. problem statement

required tools:

map *data*

Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).

2. problem statement

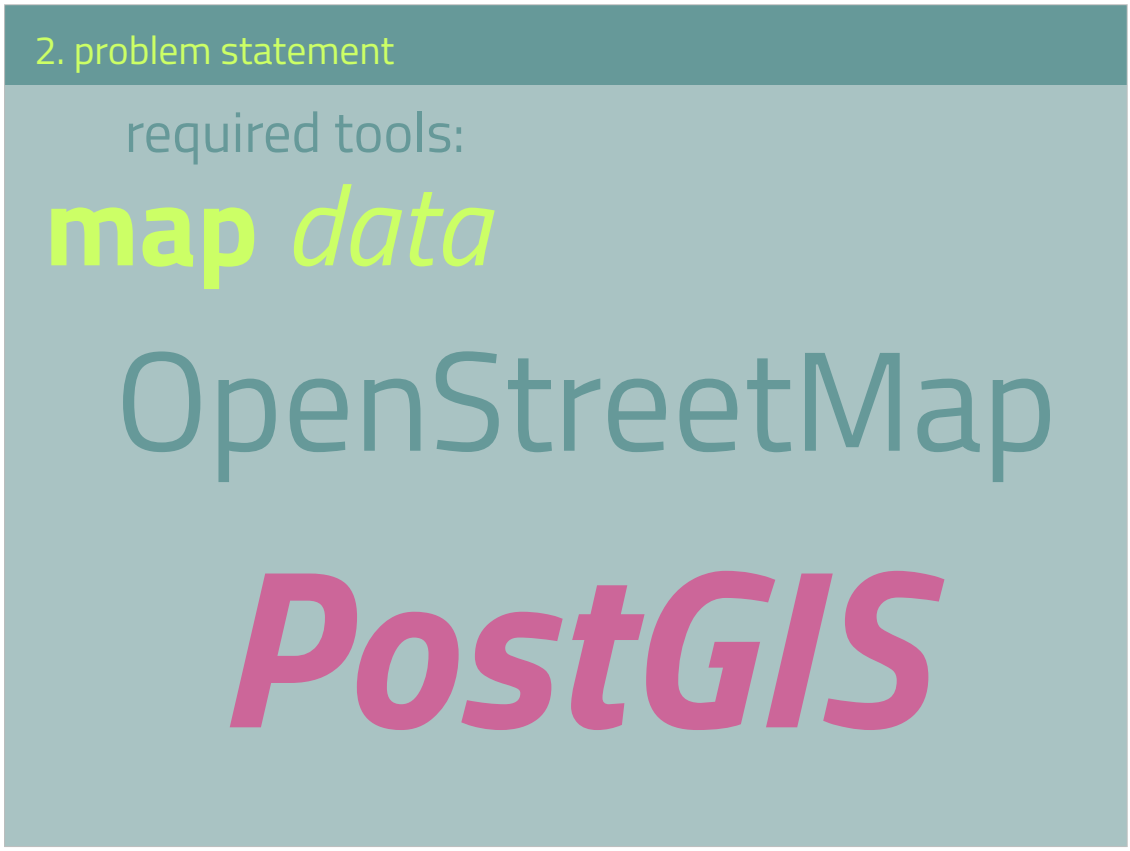
required tools:

map *data*

OpenStreetMap

Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).



Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).

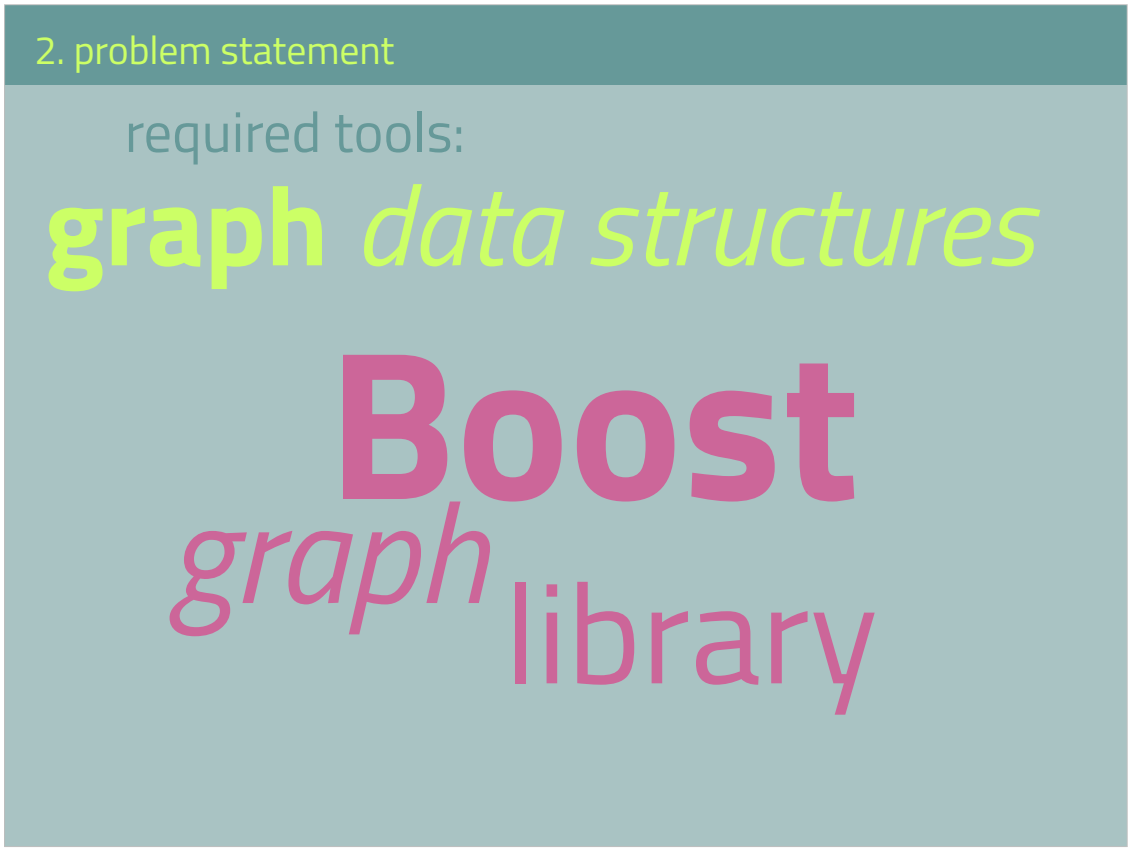
2. problem statement

required tools:

graph *data structures*

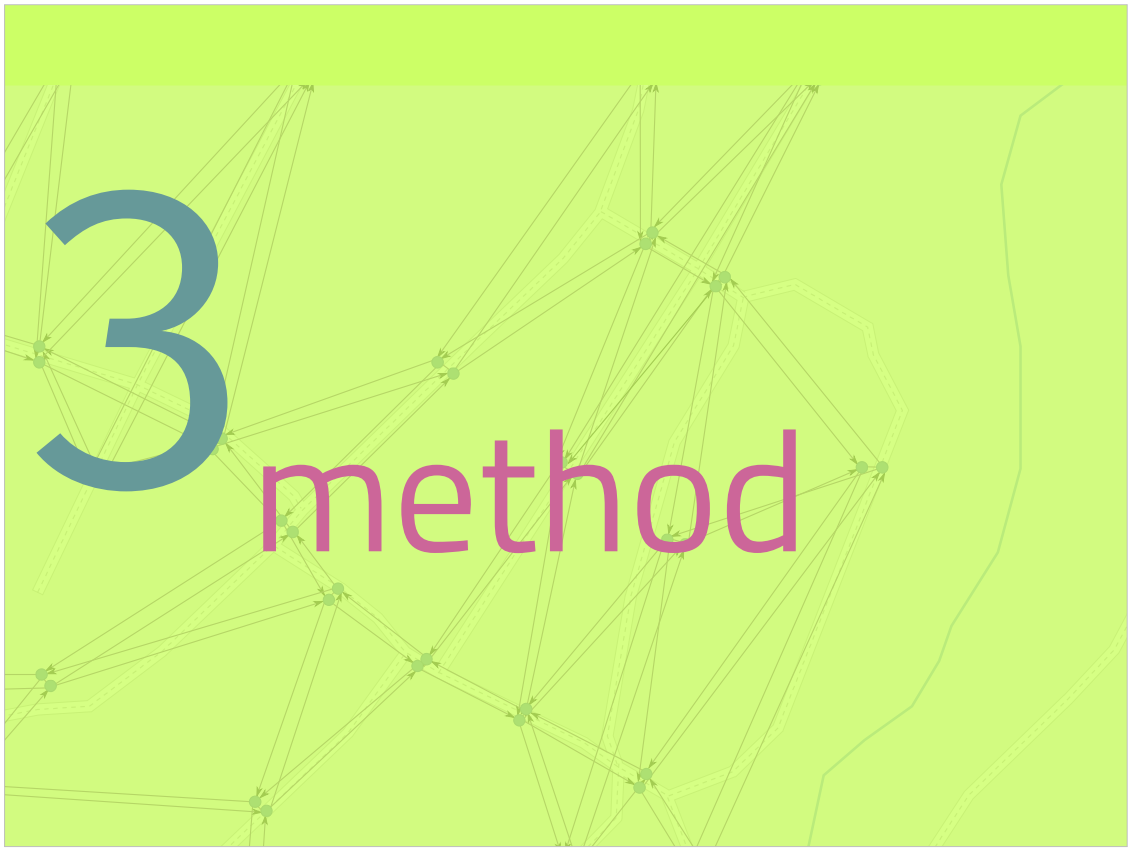
Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).



Given set of tools:

C++, Boost Graph Library,
OpenStreetMap (crowd sourced, open).
PostGis (PostgreSQL + spatial data).


3
method

Requirement

BDD or TDD

3. method

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
```

BDD test cases:

Scenario
Given
When
Then

Tools

Some were given, others I have spent a lot of time investigating.

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);
      }
    }
  }
}
```

Catch

Header only
Easy to use

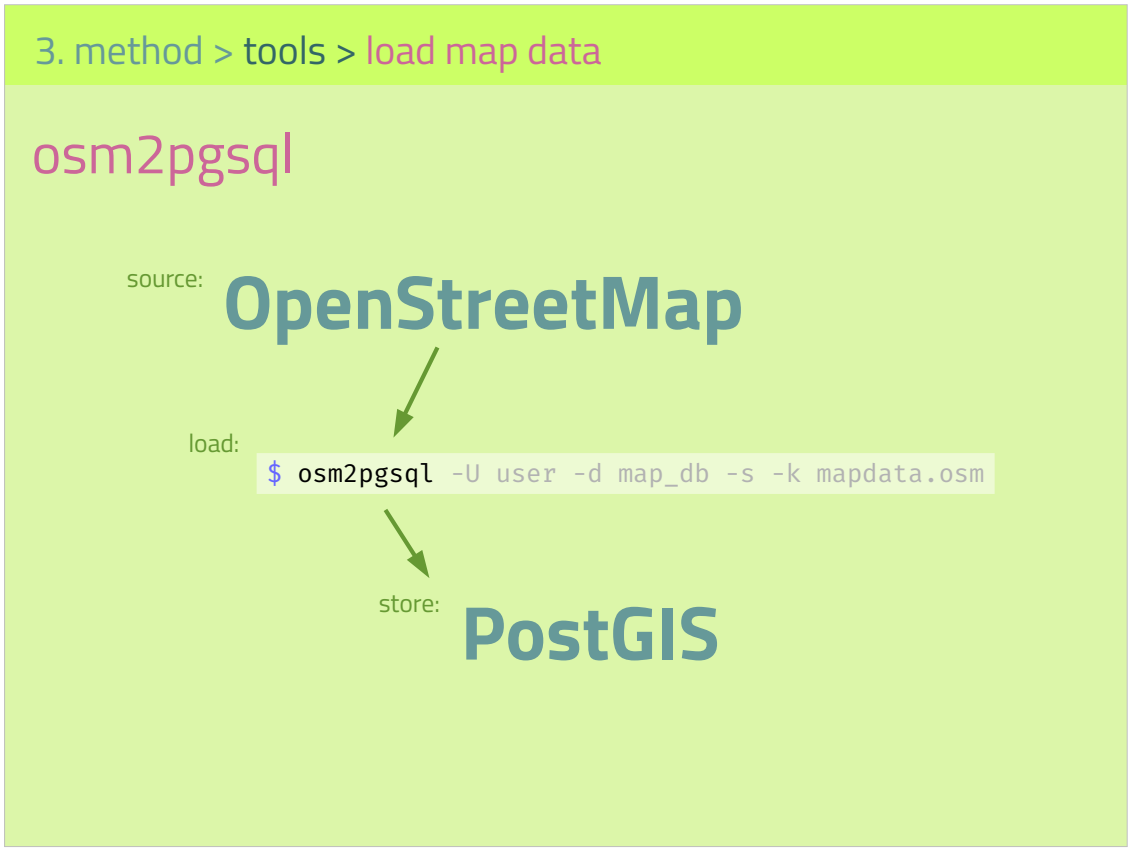
Boost Property Tree

```cpp
#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;
}
```

Reading settings from json

Boost Property Tree
As already using Boost in project

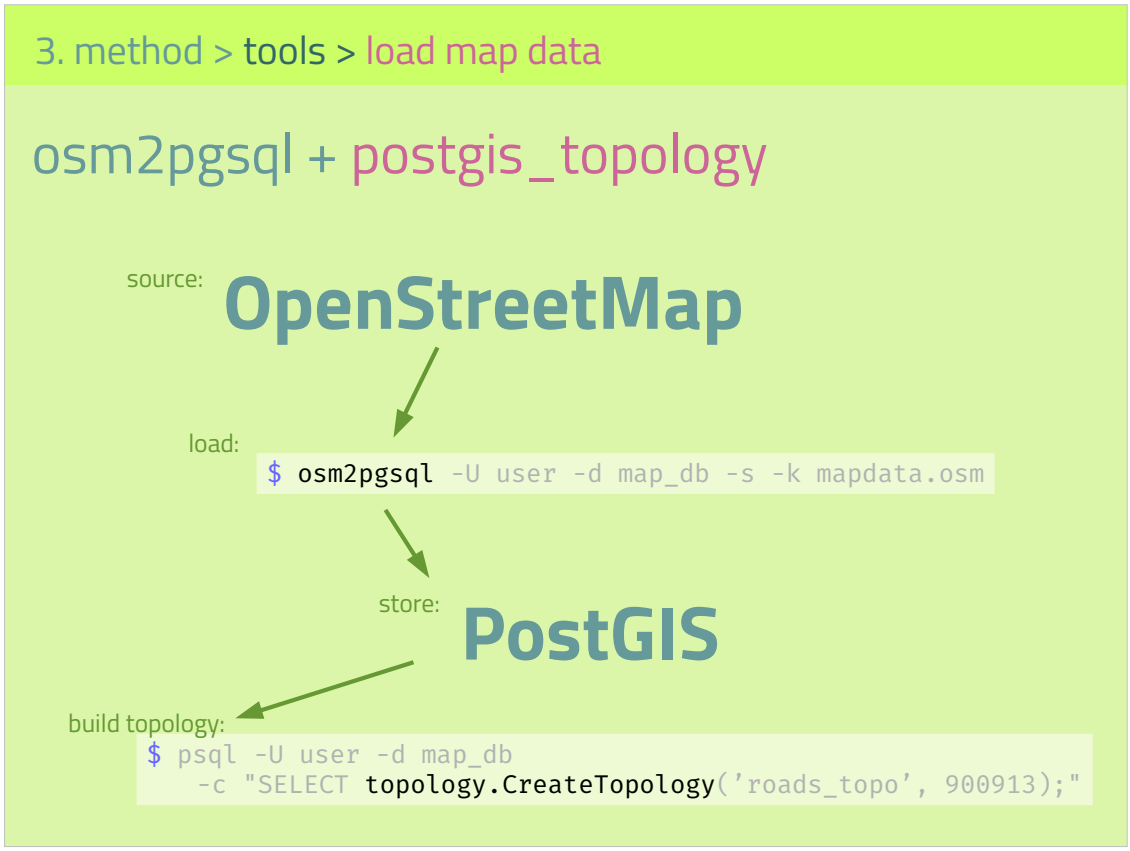


Loading map data

From .osm file
Into PostGIS

Lots of different tools

Osm2pgsql OK, but weak on relations and conditional tags...



Loading map data

Build topology

postgis_topology

Extension to PostGIS

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
```

Work against DB

- Libpqxx
  - Official library
  - Needs linking

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;
```

BGL

Hard to read with property lists

Bundled templates...

3. method > tools > data structures

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;
```

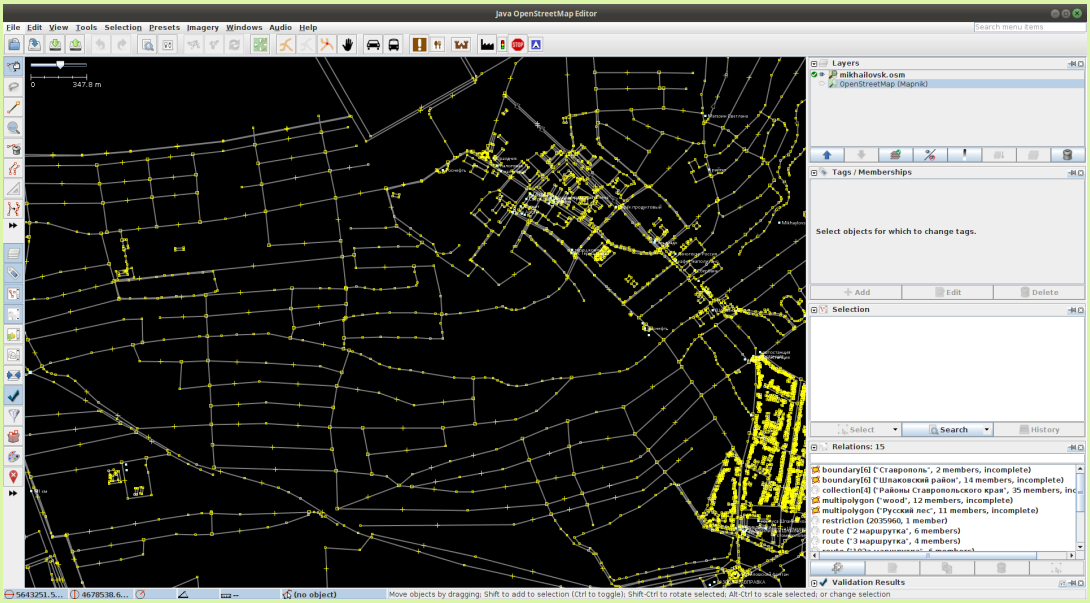
BGL

Bundled properties are more understandable

Happy to have found them!

3. method > tools > edit map data

JOSM



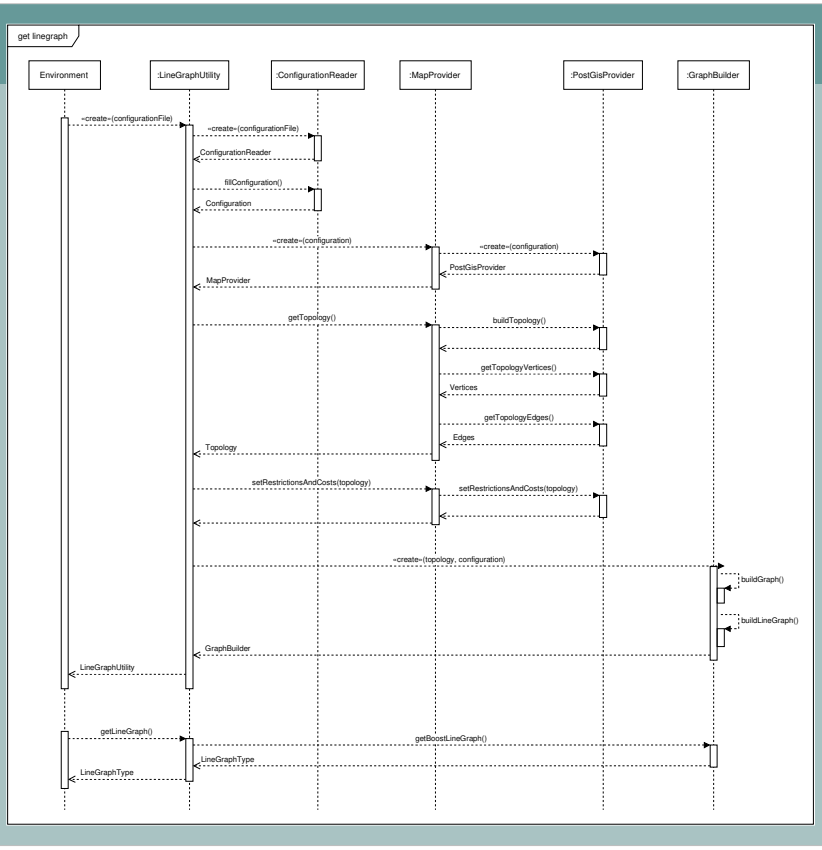
JOSM

Edit map data

Add restrictions for testing


4
results

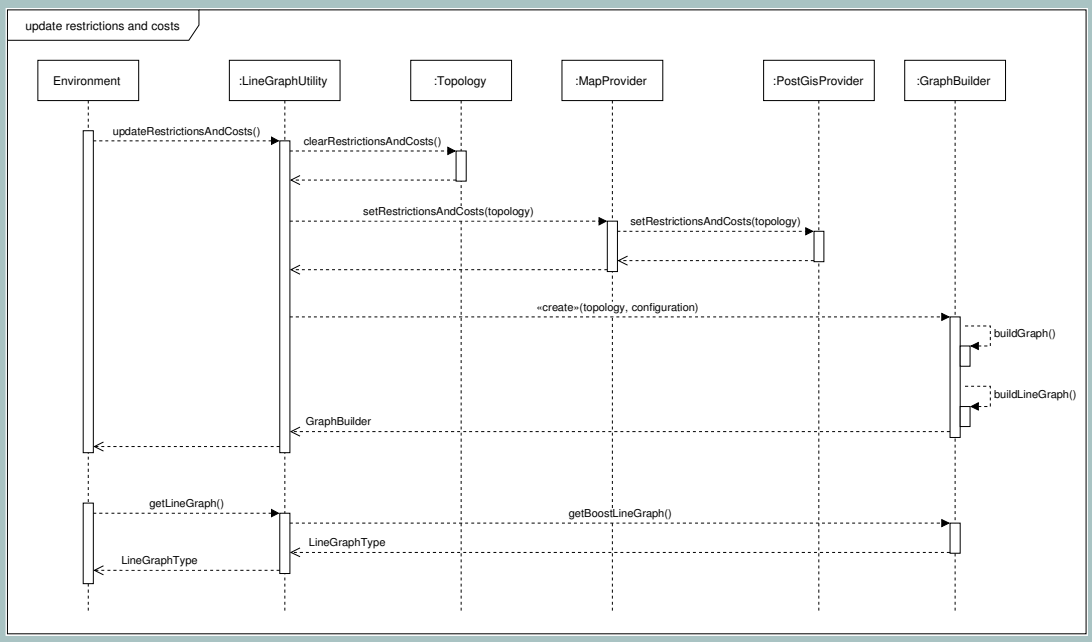
4. results > design



Main use case

- Instantiate LineGraphUtility
  - Load Configuration
  - Instantiate correct MapProvider
  - Get Topology
  - Apply Restrictions and Costs
  - Instantiate GraphBuilder
- Get LineGraph

4. results > design

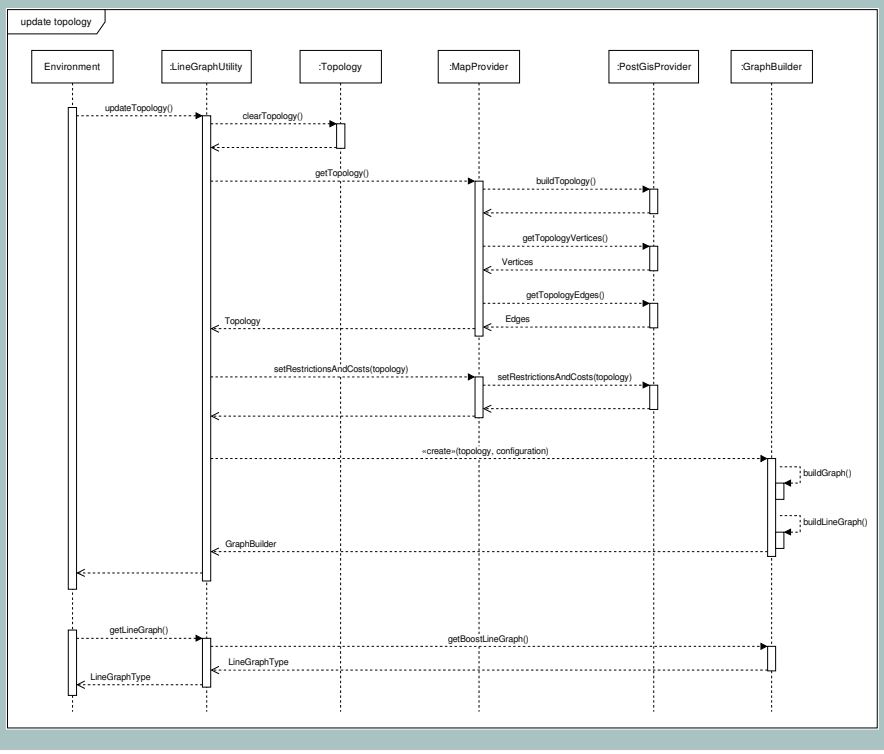


Update Restrictions and Costs

Clear Restrictions and Cost from Topology
Apply Restrictions and Costs
Instantiate GraphBuilder

(Get LineGraph)

4. results > design

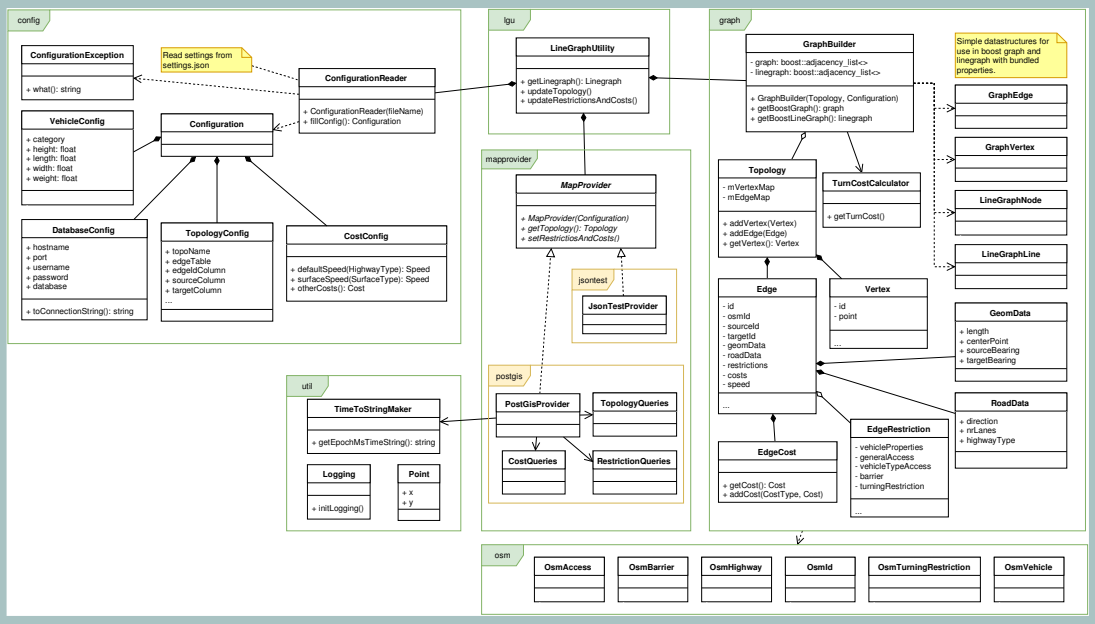


Update Topology

Clear Topology
Read in new Topology
Apply Restrictions and Costs
Instantiate GraphBuilder

(Get LineGraph)

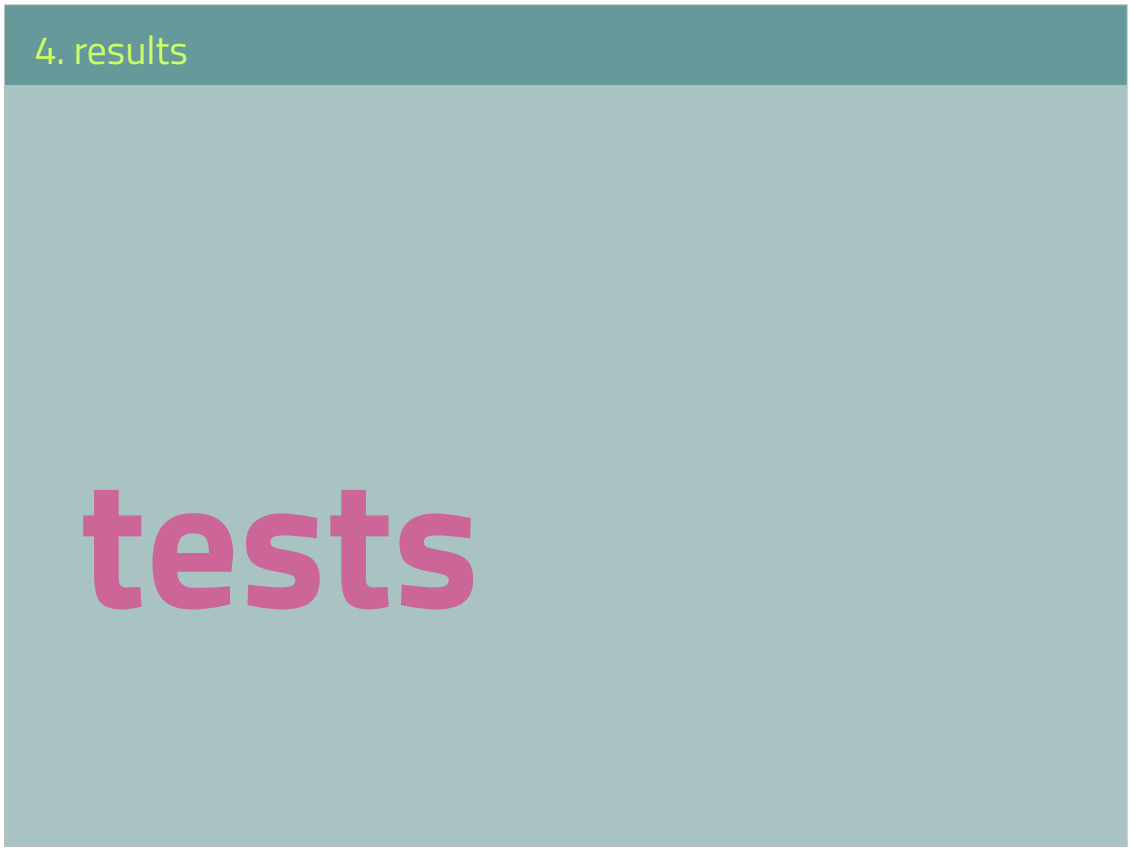
4. results > design



Class diagram

Packages:

- config
- graph
- lgu
- mapprovider
- osm
- util



Results:

Tests performed during development asserts that software produces wished results.

Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)

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
```

Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)

Example conditional restriction

Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)
And Conditions

Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)
And Conditions

Performance test

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

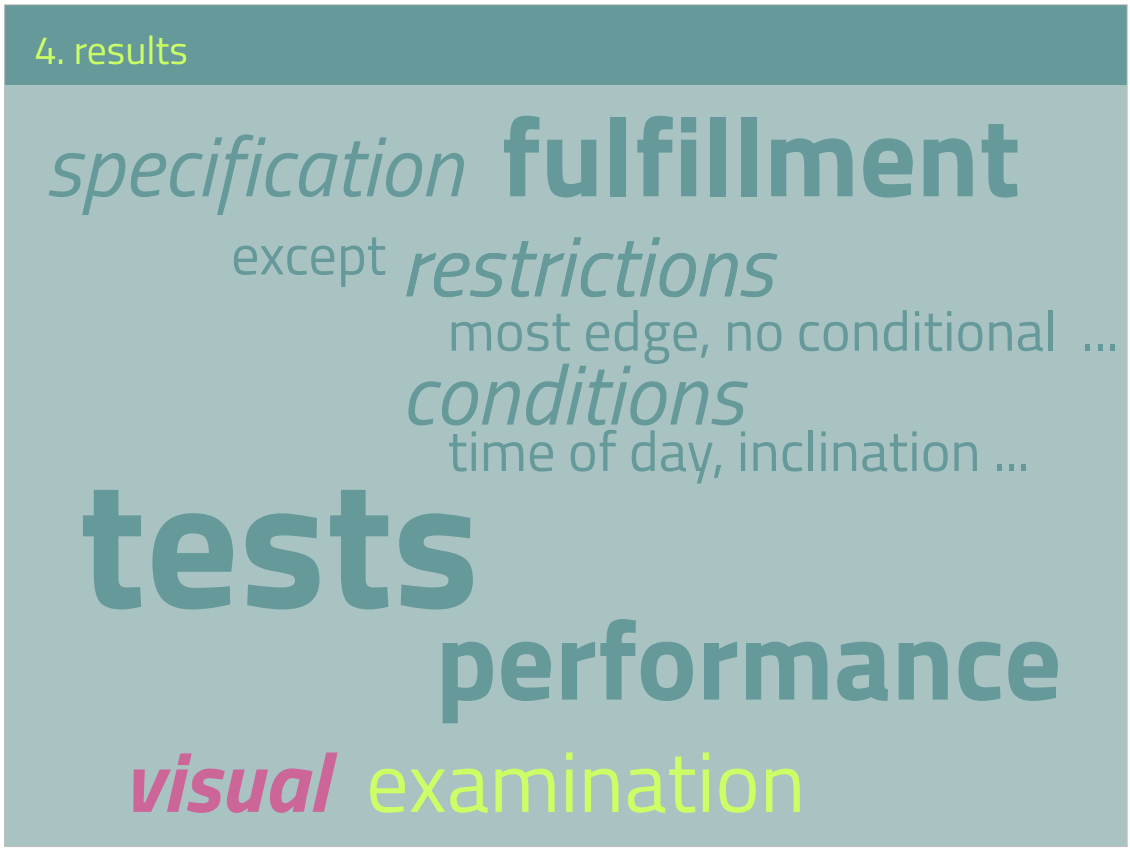
Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
- Except for Restrictions (conditional)
- And Conditions

Performance test
- Result:
  - Fetch a Line Graph
  - Topology pre-built / on demand

Soft real-time?



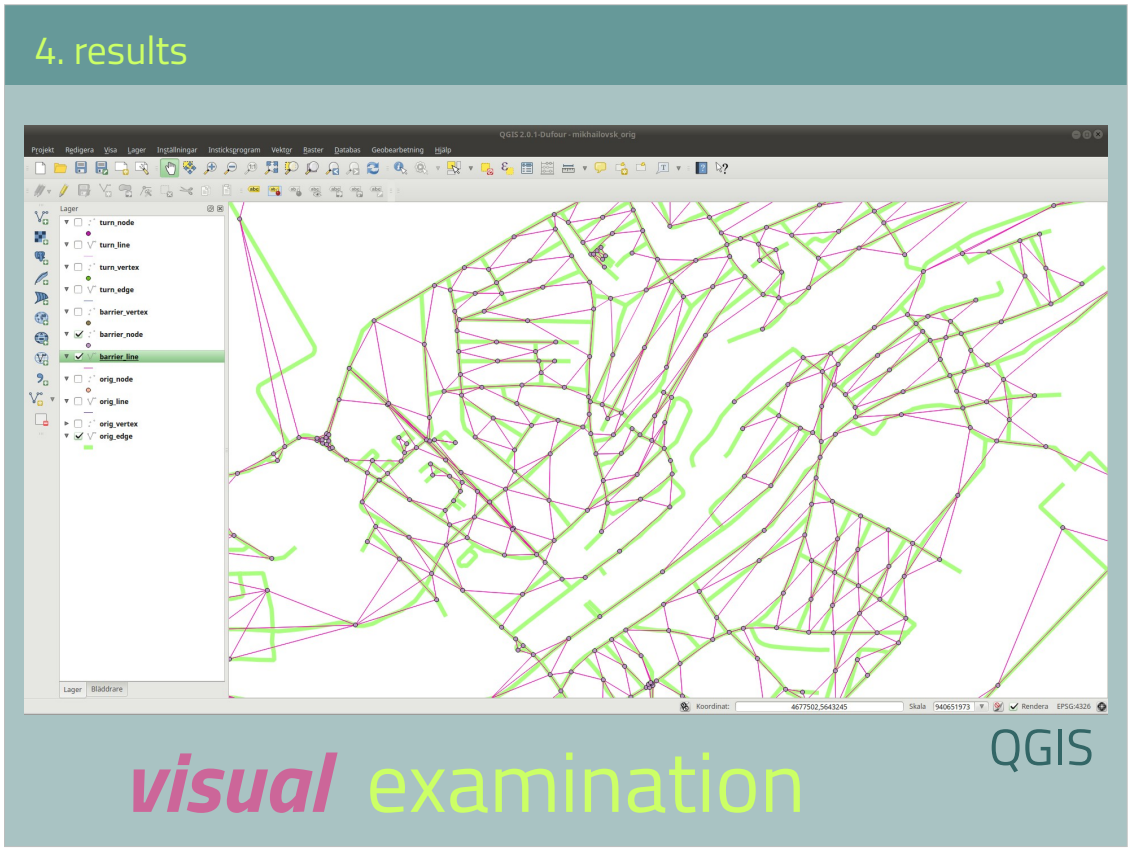
Results:

Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)
And Conditions

Performance test

Visual examination



Results:

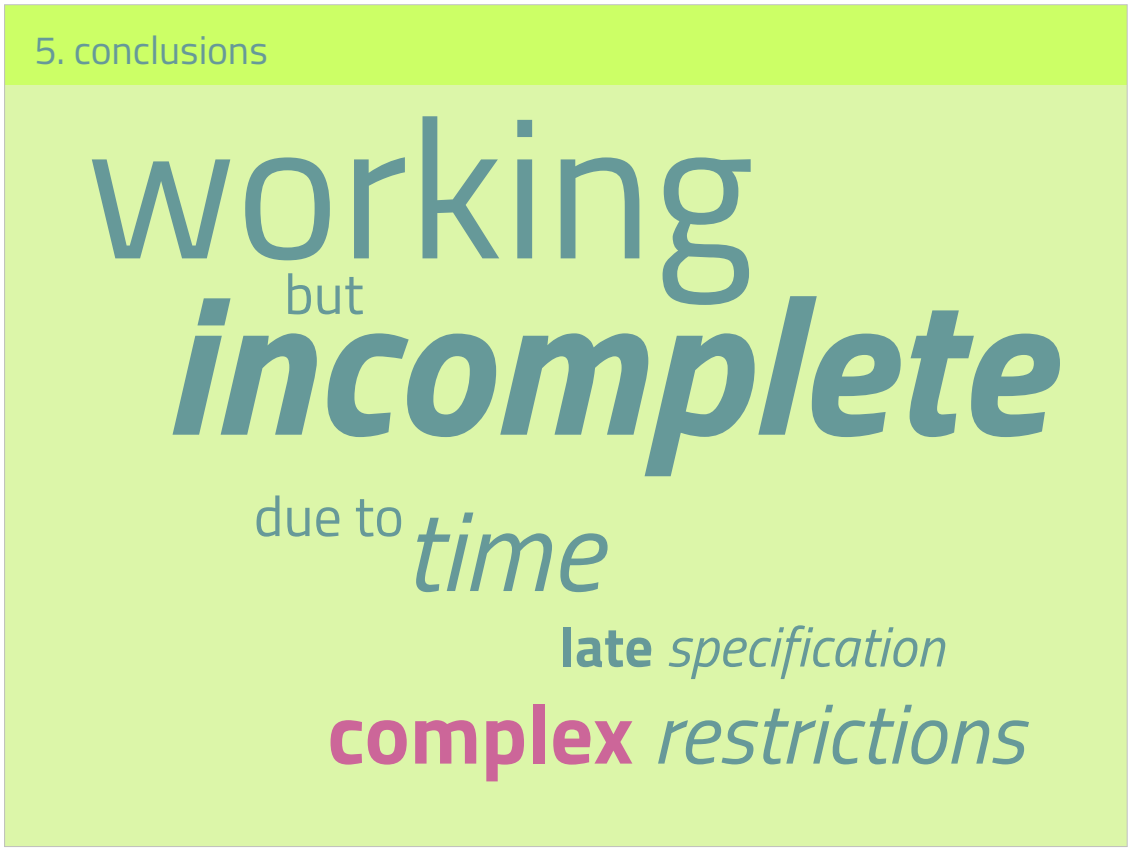
Tests performed during development asserts that software produces wished results.

The specification is fulfilled,
Except for Restrictions (conditional)
And Conditions

Performance test

Visual examination
Load data in QGIS, looks correct?


5
conclusions



Module:

- Working but incomplete

- Late from start
  - Stress, suboptimal decisions (?)

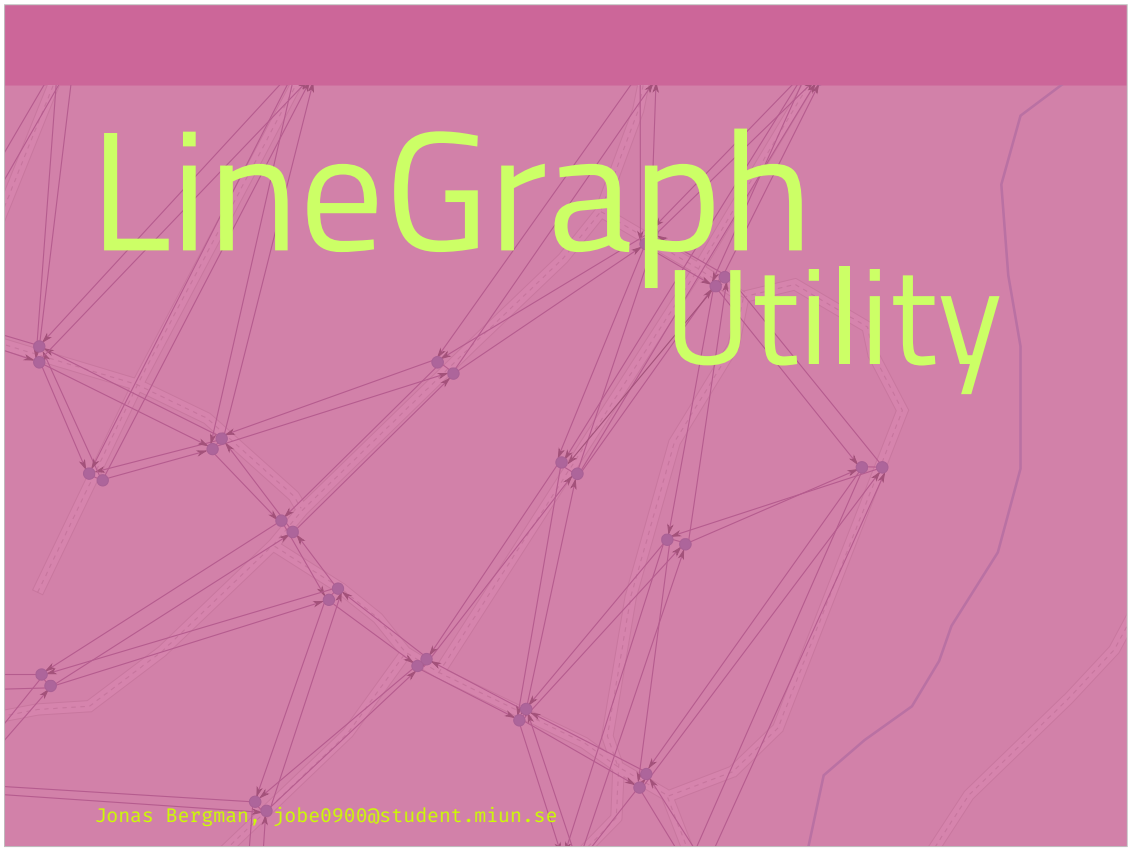
- Business logic for restrictions
  - Much more complex than thought


6
future
work

What remains:

It is possible to keep on working with the restrictions, but needs re-modeling.

Possibly using the syntax as a base for a more generic restriction class.



Have presented my exam project.