Lappeenranta University of Technology
School of Engineering Science
Degree Program in Intelligent Computing

Tikhon Belousko

SOFTWARE FOR TRANSPORT ACCESSIBILITY ANALYSIS: THE CASE OF MOSCOW

Examiners: Arto Kaarna D.Sc. (Tech.)
Vitaly Bragilevsky

Supervisors: Arto Kaarna D.Sc. (Tech.)
Vitaly Bragilevsky

ABSTRACT

Lappeenranta University of Technology
School of Engineering Science
Degree Program in Intelligent Computing

Tikhon Belousko

Software for Transport Accessibility Analysis: The Case of Moscow

2016

10 pages, 2 thingy, 3 stuff.

Examiners: Arto Kaarna D.Sc. (Tech.)
Vitaly Bragilevsky

Keywords: web-based maps, GIS, transport accessibility, vector tiles

There will be an abstract here!

PREFACE

I wish to thank my supervisor

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Finally, thank you to

Lappeenranta, October 19th, 2006

Tikhon Belousko

CONTENTS

1 INTRODUCTION

1.1 Background

In recent years, there have been many papers describing ...

1.2 Objectives and Delimitations

The aim of this paper is to generalize methods ...

1.3 Structure of the Report

The paper is divided into N main sections...

2 Architecture

2.1 Client

The first significant part of the application is the client which will be the interface for the user to manipulate map and data. The core functionality of the client is follows:

1. Panning and zooming map.
2. Selecting a point on the map to build a whole graph of the routes needed to move from selected point to all other points of the city. All of the routes are parametrized by color and width. This parameters are calculated utilizing information about transport type specified in this point and the coefficient which is defined as number of times this particular line is used in all of the directions calculated from this point.
3. Showing prevailing transport in all points of the grid. Prevailing transport is defined by calculating total time spent in each transport type. The maximum of all of the values will be taken as prevailing.
4. Showing fastest and slowest roads. The map should also have interactive slider which will be used to filter roads within particular speed range.
5. Showing most accessible points on the grid. The transport accessibility coefficient of the point can be defined as total time spent in a transport while moving from selected point to all other points. All of the values after that scaled to be between 0 and 1. This view should also have a filter by transport accessibility coefficient.

For the reason of the easy distribution it was selected to use browser based client. Thus client is implemented as single-page web application, which should support all modern browsers such as Safari, Google Chrome, Firefox, Microsoft Edge 13.

2.2 RESTful Server

For serving data to the client it was selected to utilize RESTful [1] architectural style, which represents HTTP approach to read, update and delete data. Due to the fact, that our application will only read data and not modify it, then we will only need to describe all

of the end-points for data access where data will be encoded in GeoJSON [2]. For our application following end-points were selected:

- **`GET /points`**
  The returned points are described as FeatureCollection of points with following properties
  - **name** is a string;
  - **id** is a unique identifier of the point encoded as string;
  - **prevalingTransport** could be “BUS”, “TROLLEYBUS”, “TRAM”, “SUBWAY”, “COMMUTER_TRAIN”, “SHARE_TAXI”, “DRIVING”, “WALKING”.
  - **accessibility** is a floating number in range from 0 to 1.

```
{
  "type": "FeatureCollection",
  "features": [
    "type": "Feature",
    "geometry": [0,0],
    "properties": {
      "id": 0,
      "name": "Location Name",
      "prevailingTransport": "SUBWAY",
      "accessibility": 0.6
    }
  ]
}
```

- **`GET /lines?point_id`** This endpoint is parametrized by point_id which means that if point id is presented then the client will receive only those lines which are associated with specified point. On the other hand, if no point id is not specified all lines are returned.

  The properties of the lines can be describe as follows:
  - **id** is a unique identifier of the line encoded as string;
  - **travelMode** is on of the “BUS”, “TROLLEYBUS”, “TRAM”, “SUBWAY”, “COMMUTER_TRAIN”, “SHARE_TAXI”, “DRIVING”, or “WALKING”.

- **weight** number of times this line was used in all of the routes across all of the directions' set.
- **duration** time needed to travel this line in seconds.
- **distance** total length of this line in meters.

```
{
  "type": "FeatureCollection",
  "features": [
    "type": "Feature",
    "geometry": [[0,0], [0,0]],
    "properties": {
      "id": 0,
      "width": 1,
      "weight": 200
      "duration": 100,
      "distance": 400,
      "travelMode": "BUS"
    ]
  }
}
```

It is important to note that we could also add such filtering parameters as `speed` and `travelMode`, but it was revealed during set of experiments that filtering on server side can take significant amount of time which is up to few hundreds seconds. Hence, this parameters were dropped and filtering is performed on the client.

Suggested architecture implies that client will perform rendering using GeoJSON data, although it was investigated that for our purposes rendering becomes rather slow and after series of optimizations the best rendering time was close to 3 seconds. The next improvement to the current scheme will be adding tile server which can significantly speed up rendering time.

2.3 Tile Server

Graphical map tiles are usually rectangular images in raster or vector format. Most of the popular map library providers utilize tiles [3, 4] for rendering their maps. Raster tiles are

basically images which do not allow to change color of roads and landscapes. In contrast, vector tiles are not just images but structures which contain geometries and metadata such as roads, rivers, places in special compact format. Vector tiles only rendered when requested by the client.

There are several benefits of using vector tiles. The first, advantage is small size of the tiles, which allows rendering of high resolution and caching. The second advantage is an ability to change style of the layers: adjust colors of the geometry, line width, set borders to polygons, set background patterns. Finally, maps based on vector tiles allow smooth transitions between zoom levels. Although, the cost of usage of the vector tiles is the lack of compatibility with older browsers. At the moment most of the libraries demand WebGL support and Mapbox GL JS does not support Internet Explorer below 9-th version.

REFERENCES

[1] Representational state transfer. https://en.wikipedia.org/wiki/Representational_state_transfer, 2016. [Online; accessed 19-April-2016].

[2] Allan Doyle Sean Gillies Tim Schaub Christopher Schmidt Howard Butler, Martin Daly. The geojson format specification. http://geojson.org/geojson-spec.html, 2008. [Online; accessed 19-April-2016].

[3] Map types: Tile coordinates. https://developers.google.com/maps/documentation/javascript/maptypes?hl=en#TileCoordinates, April 14, 2016. [Online; accessed 19-April-2016].

[4] Vector tiles. https://www.mapbox.com/vector-tiles/. [Online; accessed 19-April-2016].