Road network analysis with H2Network: Applications of the Spatial database H2GIS

Authors
Gwendall Petit, Adam Gouge, Nicolas Fortin, Erwan Bocher, Mireille Lecoeuvre
IRSTV – FR CNRS 2488

Contact: firstname.name@ec-nantes.fr

Otaniemi Campus, Aalto University, Espoo
June 10-13, 2014
Workshop Goal: Demonstrate practical applications of some of the research conducted during the Belgrand GEBD project

Purpose: Gather together information engineering techniques and expertise gained in various research projects on the city, mobility and the environment

Concrete objectives:
• Facilitate data access by clarifying their existence, use and access privileges
• Enable the archival, referencing and citation of data sets produced or enriched by public research
Outline

1. Introduction
2. OrbisGIS and H2GIS

Coffee break

3. H2Network
4. Use case
5. Conclusion
Introduction
Background

LAuRE law (December 30, 1996)
→ Air and the rational use of energy

“Everyone has the right to breathe air that does not harm their health”

Cities of more than 100,000 inhabitants required to establish “Urban Mobility Plans” (UMP) for transportation, traffic and parking, and reevaluate them every 5 years

→ Necessity of road network analysis becomes clear
Goals

1. Explain graph analysis techniques for creating and evaluating UMPs

2. Illustrate their usefulness in a concrete example
Tools

**OrbisGIS** - Geographical Information System

With

- **H2** - SQL database (Java)
- **H2GIS** - Spatial extension
- **H2Network** - Network analysis extension

All open source!
OrbisGIS

Open-source GIS

- Developed at the IRSTV
- Since 2007
- GPLv3
- 100% Java

V4.1 Espoo under active development with a new SQL engine to access and query data
OrbisGIS online

Official website: www.orbisdgis.org

GitHub: https://github.com/irstv/orbisdgis

Mailing lists (user & developer): http://orbisdgis.3871844.n2.nabble.com/

Twitter: https://twitter.com/OrbisGIS

Contact us: info@orbisdgis.org
The OrbisGIS framework

H2Network

JDBC API

H2 GIS

JTS Topology Suite

Java Network Analyzer (JNA)

PostGIS

H2 Database

OrbisGIS

OGRS 2014 - Road network analysis with H2Network
H2 Database

- A robust and powerful DBMS
- [http://www.h2database.com](http://www.h2database.com)
- Open-source … like PostGreSQL
- 100% Java
- Cross-platform
- No installation (live execution)
- Fully SQL compliant
- Complete documentation
- Connects to a wide range of other DBMSes
A spatial extension of H2 Database
- [http://www.h2gis.org](http://www.h2gis.org)
- Open-source
- 100% in Java
- Based on the JTS Topology Suite
- Cross-platform & no installation
- Standalone mode available (web interface)
- Implements all SFS functions and additional spatial functions
H2 and H2GIS comprise the new spatial SQL engine for OrbisGIS v4.1 Espoo to
  → access
  → manage and
  → query spatial data

H2GIS passes over 537 unit tests and has an active development community
H2Network

A set of graph analysis functions included in H2GIS

Based on **Java Network Analyzer** (JNA) developed at the IRSTV

Uses JGraphT as its graph model
Plan

1. Install OrbisGIS and get used to its user interface
2. Execute some example SQL queries on geographical data
3. Calculate shortest paths and distances in a road network shape file using H2Network
4. Do accessibility calculations for public services such as schools
5. Publish results in a map, using the latest Symbology Encoding specification
Let’s get started with OrbisGIS and H2GIS!
Install OrbisGIS

OrbisGIS V4.1 needs at least Java 7
http://www.java.com

1. Unzip “orbisgis-bin.zip”
2. In the “orbisgis-dist-4.1.0-SNAPSHOT-XX” folder launch OrbisGIS
   - Linux & Mac → $ ./orbisgis.sh
   - Windows → orbisgis.bat
Workspace manager

In OrbisGIS, you can organize your work in workspaces

Each of them has a
- path on the machine (or on a USB key)
- connection name
- JDBC URL (local or remote)
- username / password
Workspace manager

At this step it’s possible to choose your SQL engine!

You can connect to an H2(GIS) database or to a PostgreSQL/PostGIS database.

OrbisGIS will use the SQL engine you choose.

Which spatial SQL functions are used depends on this choice.
Accepting the default settings creates a new H2GIS database

Zero configuration!

Note: A “database.h2.db” file will be created in your workspace folder
OrbisGIS UI

3 main components:
- GeoCatalog
- Table of Contents (TOC)
- Map

→ minimize
→ maximize
→ un/redock
OrbisGIS UI

Other components:
- SQL console
- Groovy console
- Output window
- Plugin manager
Geocatalog

Where you manage all data (spatial or other)
- Add / Remove
- Import / Export
- Display attributes
- Filter data sources and system tables
Load data

Open data:
→ OSM (http://download.geofabrik.de/)
→ Urban Atlas (European Environment Agency)
All data have been cropped to the Helsinki area

Other data:
→ French departments (from the IGN’s GeoFLA DB)
Three choices:

1. Import data into the H2GIS DB
   - Read / Write access to imported tables
   - Export tables to flat files (i.e., .shp)
   - Import time varies with file size

2. Connect to an external DB such as PostGIS (in progress)
   - Read / Write access

3. Link to a flat file (i.e., .shp/.dbf)
   - Read only
   - Immediate (no import time)
Display geographic data in map

TOC = Table Of Contents

Functionalities:
- Display / Hide layers
- Display attributes
- Order them
- Apply styles (import / create)
- Connect to WMS streams
Navigation tools
Display attributes

Filter with a search engine

Note: Numeric fields are right-aligned
Display attributes

Actions on fields

… and on numeric fields

Table DATABASE.PUBLIC.DEPARTEMENT, statistics of the column ID_GEOFLA.
Row count : 96
Minimum : 1.0
Maximum : 96.0
Sum : 4656.0
Average : 48.5
Standard deviation : 27.85677655436824
Apply styles

Based on the new OGC Symbology Encoding (SE) standard (http://www.opengeospatial.org/standards/se)

→ Progression of SLD
→ First implementation in a GIS
→ Create / Import / Export .se files (.xml)
→ Presented at the UNO in Geneva, today!
Apply styles

Various thematic analyses
Apply styles

Value Classification - Area

General settings
- Nonspatial field: NOM_R...
- Line width unit: PX
- Enable border

Classification settings
- Use the fallback color
- Use a color scheme:
  - Gradient: Blue to Red
  - Color scheme:
    - Black
    - Green
    - Yellow
    - Orange
    - Red

Fallback symbol

Unique value classification

<table>
<thead>
<tr>
<th>Preview</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BASSE-NORMANDIE</td>
</tr>
<tr>
<td></td>
<td>BOURGOGNE</td>
</tr>
<tr>
<td></td>
<td>BRETAGNE</td>
</tr>
<tr>
<td></td>
<td>CENTRE</td>
</tr>
<tr>
<td></td>
<td>CHAMPAGNE-ARDENNE</td>
</tr>
<tr>
<td></td>
<td>CORSE</td>
</tr>
</tbody>
</table>

[Add] [Remove]

Previous  Next  Finish  Cancel
Open a WMS

1. WMS URL:
   http://kartat.espoo.fi/TeklaOgcWeb/WMS.ashx?

2. Choose layers

3. Choose CRS... and click “Finish”
Open a WMS
SQL Console

Write / Execute SQL instructions.

Functionalities:
- Execute / Stop query
- Find / Replace
- Save / Open
- Erase
- Display available functions
-- Syntaxic coloration

SELECT * FROM DEPARTEMENT WHERE NOM_DEPT='JURA';

-- You can add simple comments

/

You can also add comments

on different lines

*/

-- You can execute the script with "Ctrl + Enter"

-- Auto-completion with "Ctrl + Space"
H2GIS functions are shown in a searchable list → Drag & drop in console for documentation

→ Drag & drop to obtain all columns of a table
Some SQL exercises

Let’s practice (Spatial) SQL with data on Helsinki

→ OSM
→ Urban Atlas
Some SQL exercises

1. -- Working with the "urban_atlas_helsinki" layer

2.

3. -- "Where condition" on string fields

4. -- Select airports area

5. SELECT * FROM urban_atlas_helsinki

6. WHERE ITEM = 'Airports';

7.

8. -- "Where condition" on numeric fields

9. -- Select small area (less than 200 m²)

10. CREATE TABLE small_area AS SELECT * FROM urban_atlas_helsinki

11. WHERE SHAPE_AREA < 200;

12.

13.-- Mixed string and numeric conditions

14. CREATE TABLE small_forest AS SELECT * FROM URBAN_ATLAS_HELSINKI

15. WHERE ITEM = 'Forests'

16. AND SHAPE_AREA < 5000;
Some SQL exercises

1. -- Count the number of forests (code = 30000)

2. SELECT COUNT(*) AS nb_forests

3. FROM URBAN_ATLAS_HELSINKI

4. WHERE CODE='30000';

5. --> answer = 899

6.

7. -- Use "Group by" and "Order by" operators

8. SELECT CODE, ITEM, SUM(SHAPE_LEN) AS SHAPE_LEN, SUM(SHAPE_AREA) AS SHAPE_AREA

9. FROM URBAN_ATLAS_HELSINKI

10. GROUP BY CODE, ITEM

11. ORDER BY CODE ASC;
Some SQL exercises

1. -- Create spatial indexes (for speed) and primary keys

2.

3. -- On OSM_BUILDINGS

4. -- Create spatial index

5. CREATE SPATIAL INDEX ON OSM_BUILDINGS (the_geom);

6. -- Make the future primary key field non-nullable

7. ALTER TABLE OSM_BUILDINGS ALTER COLUMN OSM_ID SET NOT NULL;

8. -- Create the primary key

9. CREATE PRIMARY KEY ON OSM_BUILDINGS (OSM_ID);

10.

11. -- On OSM_ROADS

12. CREATE SPATIAL INDEX ON OSM_ROADS (the_geom);

13. ALTER TABLE OSM_ROADS ALTER COLUMN OSM_ID SET NOT NULL;

14. CREATE PRIMARY KEY ON OSM_ROADS (OSM_ID);
Some SQL exercises

1. -- Use spatial function in the "where" condition
2. -- Select buildings that are more than 1000 square meters
3. CREATE TABLE big_building AS SELECT *
4. FROM OSM_BUILDINGS
5. WHERE ST_AREA(the_geom)>1000;
6.
7. -- Select buildings that intersects an industrial zone
8. CREATE TABLE building_indus AS SELECT a.*
9. FROM OSM_BUILDINGS a, OSM_LANDUSE b
10. WHERE a.the_geom && b.the_geom
11. AND ST_INTERSECTS(a.the_geom, b.the_geom)
12. AND b.TYPE_OBJ='industrial';
Some SQL exercises

1. -- Select the intersection between buildings and roads

2. CREATE TABLE building_roads AS SELECT ST_INTERSECTION(a.the_geom, b.the_geom) as the_geom, OSM_ID, NAME, TYPE_OBJ

3. FROM OSM_BUILDINGS a, OSM_ROADS b

4. WHERE a.the_geom && b.the_geom

5. AND ST_INTERSECTS(a.the_geom, b.the_geom);

6. 

7. -- Count the number of lines that are more than 100 meters long

8. SELECT COUNT(*)

9. FROM building_roads

10. WHERE ST_DIMENSION(the_geom)=1 AND ST_LENGTH(the_geom)>100;

11.--> answer = 107
Some SQL exercises

1. -- Select buildings that are less than 100 meters far from a motorway

2. -- V1 - in two steps (15.177s)

3. CREATE TABLE buffer_area AS SELECT ST_UNION(ST_ACCUM(ST_BUFFER(the_geom, 100)))) AS the_geom

4. FROM OSM_ROADS

5. WHERE TYPE_OBJ='motorway';

6. CREATE TABLE building_in_buffer AS SELECT a.*

7. FROM OSM_BUILDINGS a, buffer_area b

8. WHERE a.the_geom && b.the_geom

9. AND ST_INTERSECTS(a.the_geom, b.the_geom);
Some SQL exercises

1. -- Select buildings that are less than 100 meters far from a motorway

2. -- optimized way (0.845s)

3. CREATE TABLE buffer_area AS SELECT ST_BUFFER(the_geom, 100) AS the_geom

4. FROM OSM_ROADS

5. WHERE TYPE_OBJ='motorway';

6. CREATE TABLE building_in_buffer AS SELECT DISTINCT a.*

7. FROM OSM_BUILDINGS a, buffer_area b

8. WHERE a.the_geom && b.the_geom

9. AND ST_INTERSECTS(a.the_geom, b.the_geom);

10. -- V2 - in one instruction (1.179s)

11. CREATE TABLE building_next_motorway AS SELECT DISTINCT a.*

12. FROM OSM_BUILDINGS a, OSM_ROADS b

13. WHERE a.the_geom && ST_BUFFER(b.the_geom, 100)

14. AND ST_INTERSECTS(a.the_geom, ST_BUFFER(b.the_geom, 100))

15. AND b.TYPE_OBJ='motorway';
And now H2Network!
Why H2Network?

• Network analysis in a GIS (OrbisGIS)
• Network analysis in a DBMS (H2)
• Compatibility with PostGIS
H2Network

Created to perform network analysis in OrbisGIS:

- Produce Node and Edge tables from geographical data
- Distances (Point to point)
- Distance matrices ($N$ points to $M$ points)
- Shortest paths
- Shortest path trees (optionally limited by radius)
- Accessibility analysis
- Betweenness centrality
- Closeness centrality
- Strahler stream order
Graph model choices

Java libraries:
- Graphhopper
  - API not very stable at the time
  - Somewhat difficult to extend
- JUNG
  - Community no longer active
- GraphStream
- Grph
- JGraphT

Non-Java
- PGRouting (C/C++)
  - Requires a PostgreSQL DB
Graph model choice: JGraphT

- Java
- Stable API (project started in 2003)
- Large community
- Many algorithms already implemented
- IRSTV had previously developed tools in JGraphT
H2Network, based on JNA

All algorithms implemented in the Java Network Analyzer library

Node and Edge tables are produced using SQL

H2Network = the bridge between H2GIS and JNA
JNA algorithms

Implemented on JGraphT graphs

Graph types
• oriented (directed, reversed, undirected)
• weighted (optional)

https://github.com/irstv/java-network-analyzer
Case study
Case study: time to access schools

In France, 73% of children go to school by car

Based on a study focused on the path to go to school, made by the French road security prevention association [www.preventionroutiere.asso.fr](http://www.preventionroutiere.asso.fr)

* = with an adult

Case study: time to access schools

In France, 84% of children are less than 10 min from school

Case study: time to access schools

**Context:** Reevaluating a UMP
- Every 5 years
- Examine various scenarios

What is the impact of modifying a UMP on accessibility to services?
Scenarios

S1  2010 OSM Reference situation

S2  S1 + Eric Tabarly Bridge joining two major neighborhoods of Nantes (Ile de Nantes / Malakoff), naturally separated by the Loire river

S3  S2 + Limited Traffic Zone reducing car traffic
Study area

*Nantes Métropole*
- 24 cities around the city of Nantes
- 590 000 inhabitants
- 524 km²

*Source:* www.openstreetmap.org - 2014
Open data

→ OSM from GeoFabrik (snapshot from May 21th, 2014)

→ Schools from the GeoPAL platform, a regional Spatial Data Infrastructure (downloaded on May 21th, 2014)

→ IRIS squares (200m) from INSEE website (downloaded on May 29th, 2014)
Methodology

Only open data are used!
Goal: Calculate the number of connected components of the OSM graph, using two H2Network functions:

→ ST_Graph
→ ST_ConnectedComponents
1. Evaluate network connectedness

![Number of strongly connected components](chart)

Before corrections

- 11791 strongly connected components
- 129 strongly connected components of size 10
- 4 strongly connected components of size 362
- 1 strongly connected component of size 4785

There are 3032 edges not belonging to any strongly connected component.
1. Evaluate network connectedness

Largest component
2. Correct the three road networks

Many edges are not in the largest connected component

Intersections between roads are not necessarily made

→ We propose a 3-step method to split lines and to produce nodes on intersections
2. Correct roads networks

1. Roads from OSM

2. Convert to Linestring

3. Intersect Linestrings
3. Compare network connectedness

Same quality analysis, but after the correction

- Number of strongly connected components
  - 116
  - 5
  - 1

- Number of edges composing connected component
  - 47695

There are 151 edges not belonging to any strongly connected component.
3. Compare network connectedness

Same quality analysis, but after the correction

Largest component
Corrections by hand
Corrections by hand
Corrections by hand
4. Connect services to road network

- Project to nearest road
- Recover ID of intersecting road
- Split road to create new vertex
- Create new ID (PK used by ST_Graph)
5. Produce the graph

**Definition:** A graph $G$ is an ordered pair

$$G = (V, E)$$

where $V$ is the set of vertices and

$$E = \{ \{v_i, v_j\} : v_k \in V \}$$

is the set of edges
5. Produce the graph

Definition: A directed graph has

\[ E = \{(v_i, v_j) : v_k \in V \} \]

Definition: A weighted graph is a graph equipped with a function

\[ w: E \rightarrow \mathbb{R} \]
Node and Edge tables

**CALL** ST_Graph('ROADS', 'THE_GEOM', 0.1);

→ **ROADS_NODES**, **ROADS_EDGES**

Idea: Assign integer IDs to
1. First/last coordinate of each LINESTRING
2. Assign integer IDs to each LINESTRING and its start/end vertices

→ **ROADS_EDGES** table used by other functions
5. Produce the graph

The result
Shortest paths

Shortest paths (SP) are returned as tables containing geometries and several attributes for identifying the path.

1. -- Recover edge geometries from ROADS, assuming PK key column is ID

2. CREATE TABLE ROADS_EDGES_GEOM AS

3. SELECT A.THE_GEOM, B.*

4. FROM ROADS A, ROADS_EDGES B

5. WHERE A.ID = B.EDGE_ID;
Shortest paths

ST_ShortestPath('edges', 'o[ - eo]', '[, 'w'], s, d)

<table>
<thead>
<tr>
<th>EDGES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges table <em>with geometries</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>o</th>
<th>Global orientation (directed, reversed or undirected)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>eo</th>
<th>Edge orientation (1 = directed, -1 = reversed, 0 = undirected)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>w</th>
<th>Edge weight column</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>Source id</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>d</th>
<th>Destination id</th>
</tr>
</thead>
</table>
Shortest paths

Calculate the shortest path from vertex 34723 (ID in ROADS_NODES) to vertex 15831

1. CREATE TABLE SP_34723_15831 AS
2. SELECT *
3. FROM ST_ShortestPath('ROADS_EDGES_GEOM', 'UNDIRECTED', 34723, 15831);

Multiple SPs (common in unweighted undirected) → ST_ShortestPath returns them all, each with a different PATH_ID
Shortest paths

1. -- Suppose ROADS_EDGES_GEOM contains a column EDGE_O specifying edge
2. -- orientations relative to the order of a LINESTRING's coordinates:
3. -- * 1 (directed),
4. -- * -1 (reversed),
5. -- * 0 (bidirectional).
6. -- Directed:

7. CREATE TABLE SP_34723_15831_DIR AS
8. SELECT *
9. FROM ST_ShortestPath('ROADS_EDGES_GEOM', 'DIRECTED - EDGE_O',
10. 34723, 15831);

11.-- Note: the shortest path from A to B is generally not the same as
12.-- the shortest path from B to A.
13.-- Reduce edge orientations globally: replace DIRECTED by REVERSED.
Shortest paths

1. -- Suppose ROADS_EDGES_GEOM contains a column WEIGHT specifying edge weights.

2. -- Weighted undirected:

3. CREATE TABLE SP_34723_15831_W AS

4. SELECT *

5. FROM ST_ShortestPath('ROADS_EDGES_GEOM', 'WEIGHT', 34723, 15831);

6. -- Weighted directed:

7. CREATE TABLE SP_34723_15831_W_DIR AS

8. SELECT *

9. FROM ST_ShortestPath('ROADS_EDGES_GEOM', 'DIRECTED - EDGE_O', 'WEIGHT', 34723, 15831);
## Distances / Distance matrices

\[
\begin{align*}
\text{ST}_\text{ShortestPathLength}('\text{EDGES}', 'o[ - eo][, 'w'], s, d) & \quad \text{One-to-One} \\
\text{ST}_\text{ShortestPathLength}('\text{EDGES}', 'o[ - eo][, 'w], s) & \quad \text{One-to-All} \\
\text{ST}_\text{ShortestPathLength}('\text{EDGES}', 'o[ - eo][, 'w], 'SDT') & \quad \text{Many-to-Many} \\
\text{ST}_\text{ShortestPathLength}('\text{EDGES}', 'o[ - eo][, 'w], s, 'ds') & \quad \text{One-to-Several}
\end{align*}
\]

<table>
<thead>
<tr>
<th>EDGES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Global orientation (directed, reversed or undirected)</td>
</tr>
<tr>
<td>eo</td>
<td>Edge orientation (1 = directed, -1 = reversed, 0 = undirected)</td>
</tr>
<tr>
<td>w</td>
<td>Edge weight column</td>
</tr>
<tr>
<td>s</td>
<td>Source id</td>
</tr>
<tr>
<td>d</td>
<td>Destination id</td>
</tr>
<tr>
<td>SDT</td>
<td>Source-Destination table (SOURCE, DESTINATION columns required)</td>
</tr>
<tr>
<td>ds</td>
<td>Comma-separated Destination string ('dest1, dest2, ...')</td>
</tr>
</tbody>
</table>
6. Measuring accessibility

What is accessibility?

**Geographic definition:**

*The “ease” with which destinations may be reached*

Accessibility

- Depends on the transportation network rather than the traveler
- Considers all possible itineraries towards destinations
- Defines a map of possible displacements

(Levy 2003)
6. Measuring accessibility

Graph theoretical definition: Fix a set $D \subseteq V$ of destinations. Fix a vertex $v$. Calculate the distance via the road network to each destination $d$ in $D$. Choose the closest destination.

Implementation: Reverse all edge orientations. Fix a destination $d$. Calculate the distance to each vertex $v$. Store $d$ if it is the closest destination found thus far. Repeat for all $d$ in $D$. 
Accessibility

\[
\text{ST\_Accessibility}('EDGES', 'o[ - eo]'[, 'w]', 'ds')
\]

\[
\text{ST\_Accessibility}('EDGES', 'o[ - eo]'[, 'w]', 'dt')
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EDGES</td>
<td>Edges table</td>
<td>ds</td>
</tr>
<tr>
<td>o</td>
<td>Global orientation (directed, reversed or undirected)</td>
<td>dt</td>
</tr>
<tr>
<td>eo</td>
<td>Edge orientation (1 = directed, -1 = reversed, 0 = undirected)</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>Edge weight column</td>
<td></td>
</tr>
</tbody>
</table>
6. Measuring accessibility

1. -- Given an input table DESTS(destination INT) of vertex IDs:

2. CREATE TABLE ACC AS

3. SELECT *

4. FROM ST_Accessibility('ROADS_EDGES', 'DIRECTED - EDGE_O',

5. 'WEIGHT', 'DESTS');
6. Measuring accessibility

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>CLOSEST_DEST</th>
<th>DISTANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31053</td>
<td>0056.67111647519</td>
</tr>
<tr>
<td>2</td>
<td>31854</td>
<td>6376.18559987808</td>
</tr>
<tr>
<td>3</td>
<td>31854</td>
<td>18706.46460767796</td>
</tr>
<tr>
<td>4</td>
<td>31854</td>
<td>1636.949501690324</td>
</tr>
<tr>
<td>5</td>
<td>31855</td>
<td>7001.3272390000615</td>
</tr>
<tr>
<td>6</td>
<td>31854</td>
<td>3068.5624214963564</td>
</tr>
<tr>
<td>7</td>
<td>31855</td>
<td>9504.055841371483</td>
</tr>
<tr>
<td>8</td>
<td>31854</td>
<td>9194.50691296766</td>
</tr>
<tr>
<td>9</td>
<td>31854</td>
<td>4300.071459342103</td>
</tr>
<tr>
<td>10</td>
<td>31854</td>
<td>12134.437708286616</td>
</tr>
<tr>
<td>11</td>
<td>31854</td>
<td>4408.840821523697</td>
</tr>
<tr>
<td>12</td>
<td>31853</td>
<td>9485.22169028394</td>
</tr>
<tr>
<td>13</td>
<td>31854</td>
<td>7184.123915352604</td>
</tr>
<tr>
<td>14</td>
<td>31854</td>
<td>2294.5673802372434</td>
</tr>
<tr>
<td>15</td>
<td>31853</td>
<td>9821.625445023896</td>
</tr>
<tr>
<td>16</td>
<td>31853</td>
<td>3605.9714393717568</td>
</tr>
<tr>
<td>17</td>
<td>31853</td>
<td>12207.791757100202</td>
</tr>
<tr>
<td>18</td>
<td>31854</td>
<td>14297.417113264925</td>
</tr>
<tr>
<td>19</td>
<td>31853</td>
<td>10417.70732625671</td>
</tr>
<tr>
<td>20</td>
<td>31854</td>
<td>6010.471073954411</td>
</tr>
<tr>
<td>21</td>
<td>31853</td>
<td>2237.409164101202</td>
</tr>
<tr>
<td>22</td>
<td>31854</td>
<td>26376.59686556816</td>
</tr>
<tr>
<td>23</td>
<td>31853</td>
<td>43117.777479581589</td>
</tr>
<tr>
<td>24</td>
<td>31854</td>
<td>15596.793319455752</td>
</tr>
<tr>
<td>25</td>
<td>31855</td>
<td>6288.157553294767</td>
</tr>
<tr>
<td>26</td>
<td>31853</td>
<td>6658.108647368625</td>
</tr>
<tr>
<td>27</td>
<td>31854</td>
<td>3504.071431340886</td>
</tr>
<tr>
<td>28</td>
<td>31854</td>
<td>1872.2841815807064</td>
</tr>
<tr>
<td>29</td>
<td>31854</td>
<td>3673.7603671363045</td>
</tr>
<tr>
<td>30</td>
<td>31854</td>
<td>13726.26708268152</td>
</tr>
<tr>
<td>31</td>
<td>31853</td>
<td>10778.140556335707</td>
</tr>
<tr>
<td>32</td>
<td>31854</td>
<td>33991.69320902031</td>
</tr>
<tr>
<td>33</td>
<td>31853</td>
<td>61511.105009461052</td>
</tr>
<tr>
<td>34</td>
<td>31854</td>
<td>12323.780242433188</td>
</tr>
<tr>
<td>35</td>
<td>31854</td>
<td>25593.036838438927</td>
</tr>
<tr>
<td>36</td>
<td>31853</td>
<td>6346.968010365054</td>
</tr>
<tr>
<td>37</td>
<td>31855</td>
<td>4110.48791904204</td>
</tr>
<tr>
<td>38</td>
<td>31855</td>
<td>16757.00480647751</td>
</tr>
<tr>
<td>39</td>
<td>31853</td>
<td>21374.545455985914</td>
</tr>
<tr>
<td>40</td>
<td>31853</td>
<td>2296.04577834098</td>
</tr>
<tr>
<td>41</td>
<td>31855</td>
<td>6856.995329033835</td>
</tr>
<tr>
<td>42</td>
<td>31854</td>
<td>10225.693493655372</td>
</tr>
<tr>
<td>43</td>
<td>31853</td>
<td>5543.298326621338</td>
</tr>
</tbody>
</table>
7. Continuous representation

Representation: Time contour lines

→ Time intervals based on study

0 5 10 15 20 min
7. Continuous representation

Convert points to contour lines
→ Delaunay constrained triangulation
→ Triangle contouring

→ *Use ‘time_intervals.se’ style file*
Scenario comparison

Time map
Scenario comparison

Time maps

Scenario 1

Scenario 2

Scenario 3
Scenario comparison

→ Classification intervals densified
Scenario comparison

Scenario 1

Scenario 2

Scenario 3

Time maps
Scenario comparison

Distance map

OGRS 2014 - Road network analysis with H2Network
Scenario comparison

Scenario 1

Scenario 2

Scenario 3

Distance maps
Before concluding
H2GIS in "standalone" mode

You can use H2GIS in standalone mode in a web browser

Simply extract h2gis-standalone-bin.zip, available on www.h2gis.org

No installation!

Lightweight!
(less than 6MB)

Android compliant
H2GIS in "standalone" mode

Commandes principales

Exemple de script SQL

Ajouter des pilotes de base de données

Des pilotes additionnels peuvent être configurés en déclarant l'emplacement du fichier Jar contenant ces pilotes dans les variables d'environnement H2DRIVERS ou CLASSPATH. Exemple (Windows): Pour ajouter la bibliothèque C:\Programs\hsqldb\hsqldb.jar, définir la valeur de la variable d'environnement H2DRIVERS en C:\Programs\hsqldb\hsqldb.jar.
Conclusion
Conclusion

Using open-source tools and open data we have demonstrated techniques for analyzing road networks.

These analyses are very useful for evaluating UMPs and can help public officials make good decisions.
Future work

Synchronize data with OSM’s servers

Automatize the process and use WPS to provide a web service which is always up-to-date

Measure accessibility relative to public transportation stops and service frequency at each stop
Note

This workshop features practical applications of some of the ideas found in the paper “H2Network: A tool for understanding the influence of urban mobility plans (UMP) on spatial accessibility”.

You are welcome to join us tomorrow at 16:00 during the Urban analysis and applications session.
Acknowledgments

The OrbisGIS-H2GIS team would like to thank the French Belgrand-GEBD project for supporting their research, as well as
Thank you for your attention

Questions


