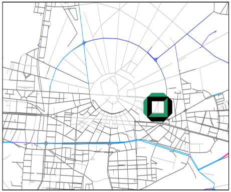
Study Thesis
Visualisation of Very Large Graphs



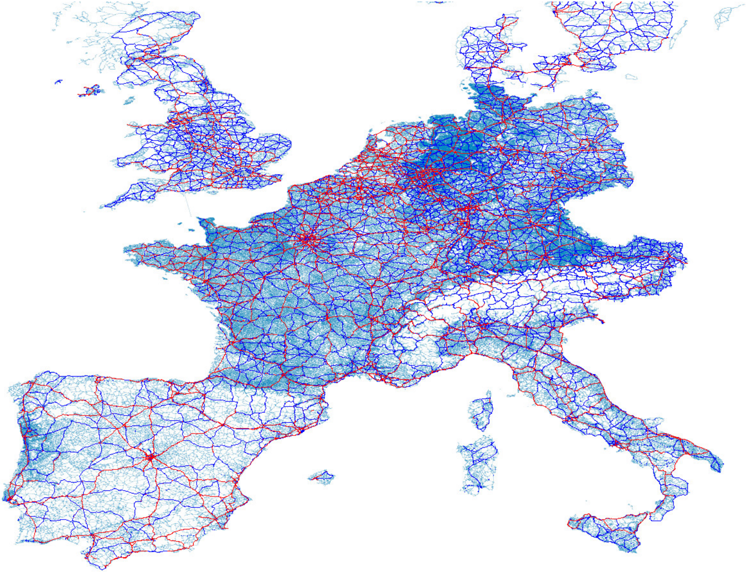
Timo Bingmann

Aug 2, 2006

Road Map

1. Introduction
   - Motivation: Street Network of Europe
   - Library Features
2. Architecture & Data Structures
   - Basic Architecture
   - Data Structures: R-Tree and Adjacency-Array
   - Queries: Serialization and Query Parser
3. Experiments
4. Demo

Motivation



Motivation

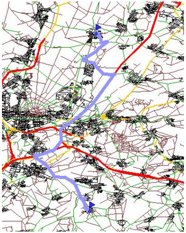
Visualisation of a street network of Europe.

Magnitudes

- About 18 million vertices and 22 million edges.
- Last picture: only about 3 million edges.

Application

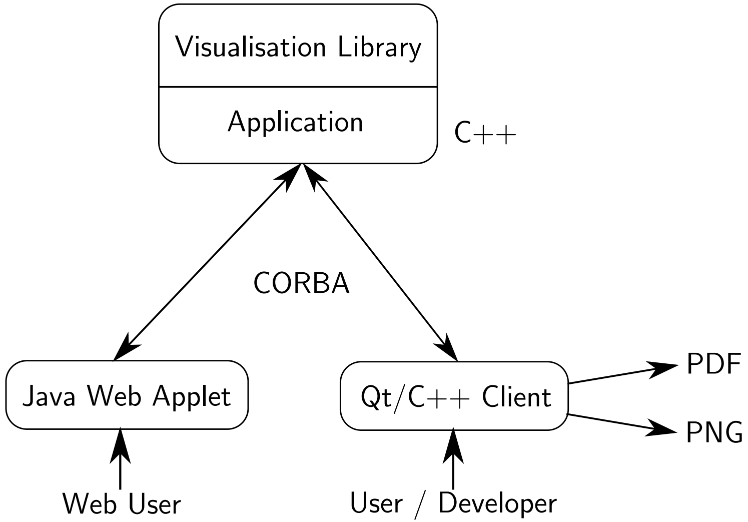
- Route planning
  ⇒ drawing of paths.



Visualisation Library

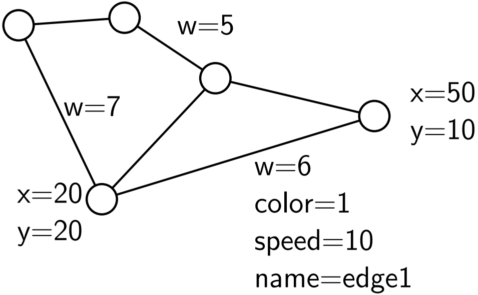
- Supports any two dimensional layouted graph.
- Very fast query speeds: < 1 sec.
- Seamless integration into existing applications.
- Easily animate calculation mechanisms of algorithms.
- Fast and user-friendly browsing at presentations or via the Internet.
  $\Rightarrow$ Java web applet.
- High-quality exports of sections to PDF or PNG for presentations and papers.

Basic Architecture



Supported Graphs

- Two dimensional layouted graph
- An additional z-axis (significance)
- Attributes on vertices and edges: coordinates and drawing parameters.
- Each attribute has a type like `bool`, `char`, `integer` or `string`.

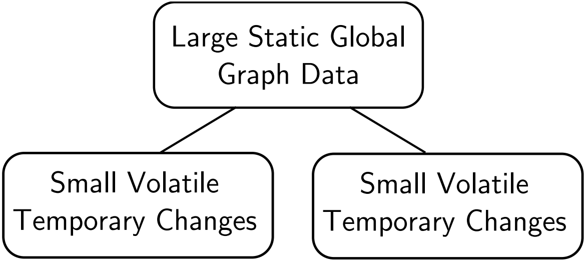


Analysis: Route Planning

The route planning algorithm operates on a street network.

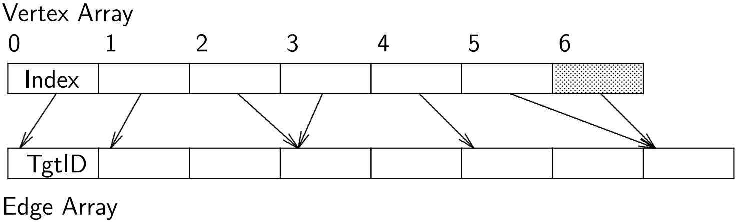
- Large volume of unchanging graph data.
  Route planning never changes streets.
- Only small set of edges are marked by the algorithm.
- Marked edges are undone after viewing.
- ⇒ Separate static graph data from temporary changes.

Separation


Large Static Global Graph Data
Small Volatile Temporary Changes
Small Volatile Temporary Changes

- Temporary changes are an overlay graph.
  ⇒ efficient rollback of changes.
- Can apply compact data structures to static graph data. ⇒ adjacency array
- Support of multiple simultaneous clients.
  ⇒ multi-threading support.

Adjacency Array



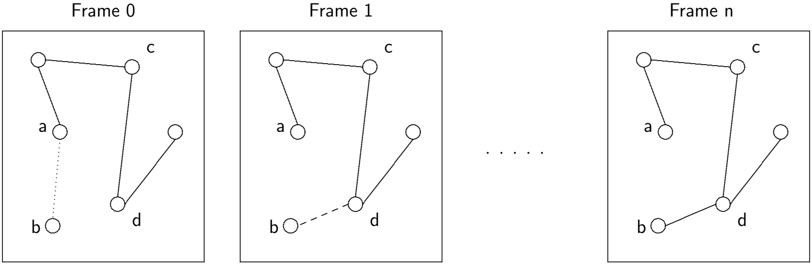
- Compact and easy to serialize.
- Array has to be rebuilt to apply changes.
- Attribute values are stored in a similar fashion.
- `GraphLoader` class for direct loading of arrays.

Changelist

- Save temporary changes in flexible `hash_map` structures.
- Support convenient functions to change graph data:
  - `addVertex(vid)`
  - `setVertexAttr(vid, attrid, value)`
  - `delVertex(vid)`
  - `addEdge(src, tgt)`
  - `setEdgeAttr(src, tgt, attrid, value)`
  - `delEdge(src, tgt)`

Animation Timeline

- Changes can be animated by setting time frame markers in the sequence of function calls.



```
delEdge(a,b);
advanceTimeFrame();
```

```
setEdgeAttr(c,d,1,50);
addEdge(b,d);
advanceTimeFrame();
```

Index Structure

Required

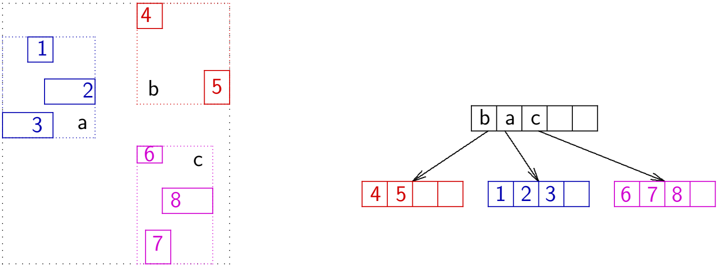
Spatial index structure to accelerate range queries on the graph. Needs to support zooming and extraction in z-order.

Selected

R-Tree

R-Tree

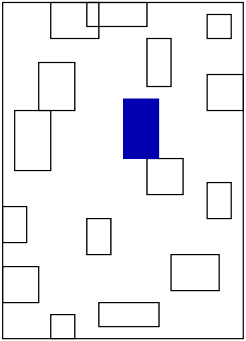
- Based on B-Tree, but contains rectangles instead of numbers.
- Efficient for very large number of rectangles through high fan-out.



R-Tree Properties

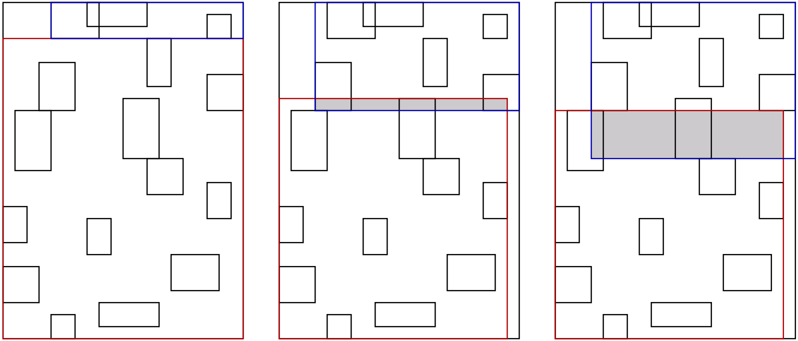
- Define M maximum and m minimum number of rectangles in a node. Let $m \leq \frac{M}{2}$.
- Every node contains between m and M rectangles or it is the root.
- The root contains at least two rectangles or it is a leaf.
- Every rectangle in an inner node is the minimum bounding-box of the rectangles contained in its subtree.
- All leaves are on the same level.

R-Tree Splitting

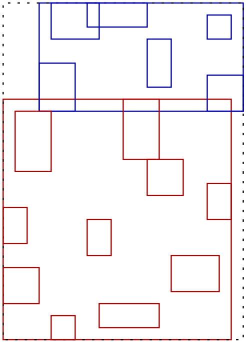


- How to find a good split when a node overflows?

R-Tree Splitting

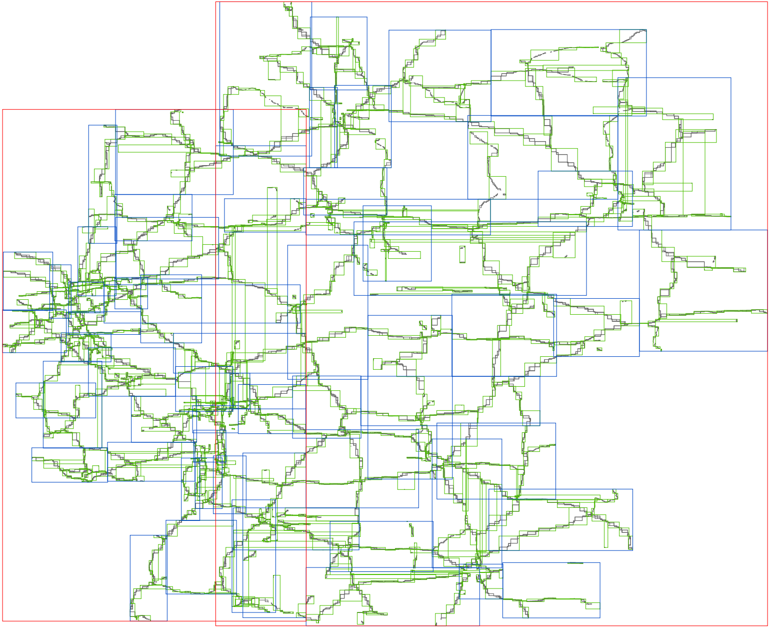


R-Tree Splitting



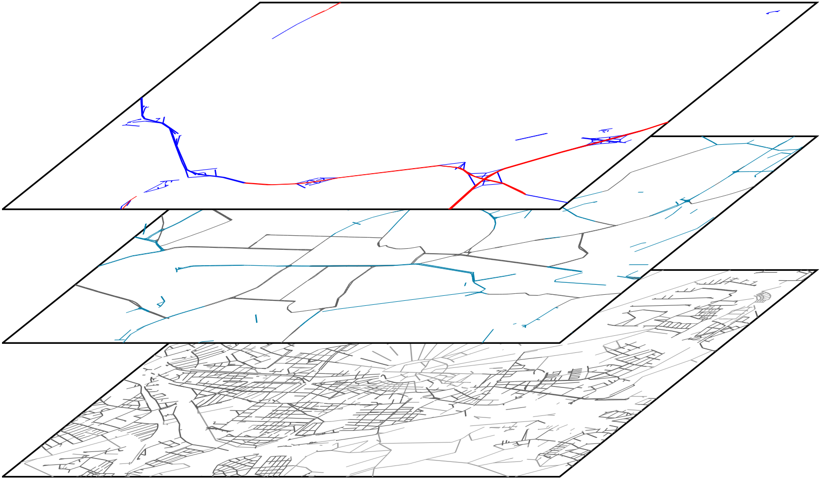
- Library contains three R-tree Variants:
  R-Tree with quadratic Split, R-Tree with linear Split and R*-Tree.

Germany's Autobahnen

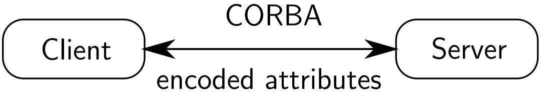


Multilevel R-Tree

Multiple R-Trees are used to support extraction in z-order.

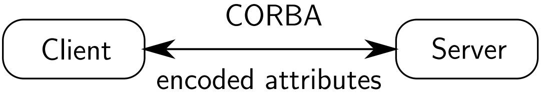


getArea Query



- Vertices and edges are extracted and sent to the client in a serialized binary format.
- Change function calls are sent as an animation script.
- Visualisation library is not limited to CORBA as middleware.

getArea Query



- Send only attributes required to draw the graph.
- Screen coordinate transformation is calculated on the server. Transfered as `short`.
- User can set a filter to limit the drawn edges.

Parser

Server contains an arithmetic parser used to parse

- attribute selection strings

```
(x - 5411) * 0.331 cast short, ..., speed
```

- and user filter strings.

```
edges:  (speed < 5 and distance > 50) or (speed >= 5)
```

id	1	2	3	4	5	6
x	5641	5560	5755	5708	5638	5236
y	4845	4853	5002	4905	4998	4821
speed	5	6	1	3	4	2
distance	42	12	6	66	36	22
(x−5411)*0.331	76	49	113	98	75	−57

Integration

Easy integration into existing programs.

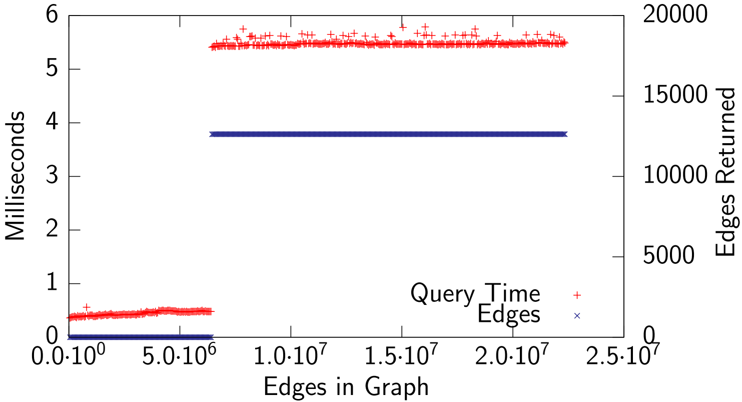
- Well-designed C++ namespace with lots of doxygen documentation.
- Animation is automatically created from sequence of function calls.
- Accelerated loading from snapshot data files containing the complete server state.

Map Sizes

Map	Vertices	Base Graph	R-Trees
	Edges	Attributes	Total
Luxembourg	30 747	538 KB	517 KB
	38 143	531 KB	1586 KB
Belgium	463 795	8 269 KB	7 895 KB
	594 715	8 142 KB	24 307 KB
Netherlands	893 407	15 920 KB	15 174 KB
	1 144 337	15 675 KB	46 769 KB
Germany	4 378 447	77 210 KB	73 643 KB
	5 504 454	76 111 KB	226 964 KB
Europe	18 029 722	315 385 KB	301 322 KB
	22 339 557	311 176 KB	927 883 KB

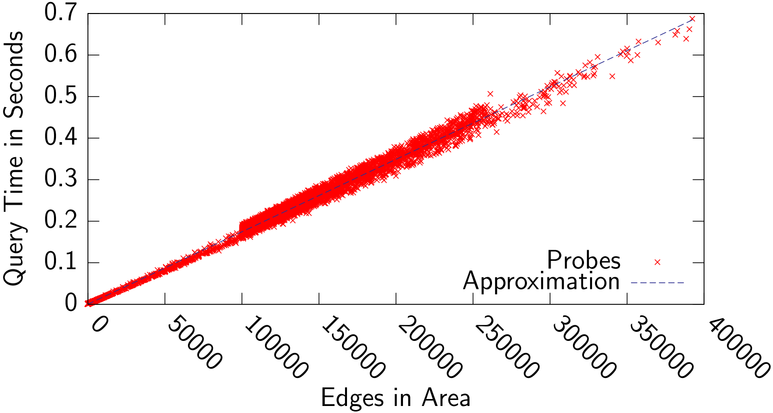
Table: Map Sizes

Query Speed



- Street network of Europe built incrementally.
- Query time measured on a fixed view of Karlsruhe with surrounding cities.

Query Speed



- Query time of 1000 random areas on the street network of Europe.

Demo

- Qt client with user-defined drawing rules.
- Java web client with integrated route planning algorithm.

  http://algo2.iti.uni-karlsruhe.de/schultes/hwy/demo/