Algorithm Engineering for Large Graphs

Engineering Route Planning Algorithms

Peter Sanders Dominik Schultes

Universität Karlsruhe (TH)

Online Topological Ordering for Dense DAGs

Deepak Ajwani Tobias Friedrich Ulrich Meyer

MPII Saarbrücken Universität Frankfurt

Freiburg, July 4, 2007
Route Planning

Goals:

- exact shortest (i.e. fastest) paths in large road networks
- fast queries (point-to-point, many-to-many)
- fast preprocessing
- low space consumption
- fast update operations

Applications:

- route planning systems in the internet, car navigation systems,
- traffic simulation, logistics optimisation
Overview

**Highway Hierarchies**
- foundation
  - [ESA 05, ESA 06]

**HH Star**
- goal-directed
  - [DIMACS 06]

**Transit Node Routing**
- very fast queries
  - [DIMACS 06, ALENEX 07, Science 07]

**Hwy–Node Routing**
- allow edge weight changes
  - [WEA 07]

**Many-to-Many**
- compute distance tables
  - [ALENEX 07]
Highway Hierarchies

Construction: iteratively alternate between

- removal of low degree nodes
- removal of edges that only appear on shortest paths close to source or target

yields a hierarchy of highway networks in a sense, classify roads / junctions by ‘importance’
Highway Hierarchies

- foundation for our other methods
- directly allows point-to-point queries
- 16 min preprocessing
- 0.76 ms to determine the path length
- 0.93 ms to determine a complete path description
- reasonable space consumption (68 bytes/node) can be reduced to 17 bytes/node
Highway Hierarchies Star

joint work with D. Delling, D. Wagner

- combination of highway hierarchies with goal-directed search
- slightly reduced query times (0.68 ms)
- more effective
  - for approximate queries or
  - when a distance metric instead of a travel time metric is used
joint work with S. Knopp, F. Schulz, D. Wagner

[ALENEX 07]

- efficient many-to-many variant of the highway hierarchies query algorithm

- 10,000 × 10,000 table in one minute
Transit-Node Routing

[DIMACS Challenge 06, ALENEX 07, Science 07]

joint work with H. Bast, S. Funke, D. Matijevic

- very fast queries (down to 6\,\mu s, 1\,000\,000 times faster than Dijkstra)
- winner of the 9th DIMACS Implementation Challenge
- more preprocessing time (2:44 h) and space (251 bytes/node) needed
Highway-Node Routing

- performance similar to highway hierarchies
- outstandingly low space requirements
- conceptually very simple
- handles dynamic scenarios

[WEA 07]
Overlay Graph


- graph $G = (V, E)$ is given
- select node subset $S \subseteq V$
Overlay Graph


- graph $G = (V, E)$ is given
- select node subset $S \subseteq V$

- overlay graph $G' := (S, E')$ where

$$E' := \{(s, t) \in S \times S \mid \text{no inner node of the shortest } s-t\text{-path belongs to } S\}$$
Query

☐ bidirectional

☐ perform search in $G$ till search trees are covered by nodes in $S$
Query

- bidirectional
- perform search in $G$ till search trees are covered by nodes in $S$
- continue search only in $G'$
Static Highway-Node Routing

- extend ideas from
  - multi-level overlay graphs [HolzerSchulzWagnerWeiheZaroliagis00–07]
  - highway hierarchies [SS05–06]
  - transit node routing [BastFunkeMatijevicSS06–07]

- use highway hierarchies to classify nodes by ‘importance’
  i.e., select node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$
  (crucial distinction from previous separator-based approach)

- construct multi-level overlay graph

- perform multi-level query
Static Highway-Node Routing

- extend ideas from
  - multi-level overlay graphs [HolzerSchulzWagnerWeiheZaroliagis00–07]
  - highway hierarchies [SS05–06]
  - transit node routing [BastFunkeMatijevicSS06–07]

- use highway hierarchies to classify nodes by ‘importance’
  i.e., select node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$ 16 min
  (crucial distinction from previous separator-based approach)

- construct multi-level overlay graph 3 min, 8 bytes/node

- perform multi-level query 1.1 ms

(experiments with a European road network with $\approx 18$ million nodes)
Dynamic Scenarios

- change entire cost function
  (e.g., use different speed profile)

- change a few edge weights
  (e.g., due to a traffic jam)
Dynamic Highway-Node Routing

change entire cost function

keep the node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$

recompute the overlay graphs

<table>
<thead>
<tr>
<th>speed profile</th>
<th>default</th>
<th>fast car</th>
<th>slow car</th>
<th>slow truck</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>constr. [min]</td>
<td>1:40</td>
<td>1:41</td>
<td>1:39</td>
<td>1:36</td>
<td>3:56</td>
</tr>
<tr>
<td>query [ms]</td>
<td>1.17</td>
<td>1.20</td>
<td>1.28</td>
<td>1.50</td>
<td>35.62</td>
</tr>
<tr>
<td>#settled nodes</td>
<td>1 414</td>
<td>1 444</td>
<td>1 507</td>
<td>1 667</td>
<td>7 057</td>
</tr>
</tbody>
</table>
Dynamic Highway-Node Routing

change a few edge weights

☐ server scenario: if something changes,
  – update the preprocessed data structures
  – answer many subsequent queries very fast

☐ mobile scenario: if something changes,
  – it does not pay to update the data structures
  – perform single ‘prudent’ query that takes changed situation into account
Dynamic Highway-Node Routing

change a few edge weights, server scenario

<table>
<thead>
<tr>
<th>Road Type</th>
<th>Update Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>0.1 1 10 100</td>
</tr>
<tr>
<td>motorway</td>
<td>0.1 1 10 100</td>
</tr>
<tr>
<td>national</td>
<td>0.1 1 10 100</td>
</tr>
<tr>
<td>regional</td>
<td>0.1 1 10 100</td>
</tr>
<tr>
<td>urban</td>
<td>0.1 1 10 100</td>
</tr>
</tbody>
</table>

- add traffic jam
- cancel traffic jam
- block road
Dynamic Highway-Node Routing

change a few edge weights, mobile scenario

- keep the node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$
- keep the overlay graphs
- use auxiliary data to determine for each node $u$ a reliable level $r(u)$
- during a query, at node $u$
  - do not use edges that have been created in some level $> r(u)$
  - instead, downgrade the search to level $r(u)$
Summary

Highway Hierarchies: fast queries, fast preprocessing, low space,
few tuning parameters,
basis for many-to-many and transit-node / highway-node routing.

Many-to-Many: huge distance tables are tractable,
subroutine for transit-node routing.

Transit-Node Routing: fastest routing so far.

Highway-Node Routing: ‘simpler’ highway hierarchies,
fast queries, very low space,
efficiently dynamizable.