Accurate High-Performance
Route Planning

Peter Sanders       Dominik Schultes

Institut für Theoretische Informatik – Algorithmik II
Universität Karlsruhe (TH)

http://algo2.itl.uka.de/schultes/hwy/
How do I get there from here?

Applications

- route planning systems in the internet (e.g. www.map24.de)
- car navigation systems
- ...

Sanders/Schultes: Route Planning
Goals

- exact shortest (i.e. fastest) paths in large road networks
- fast queries
- fast preprocessing
- low space consumption
- scale-invariant, i.e., optimised not only for long paths
## Related Work

<table>
<thead>
<tr>
<th>method</th>
<th>query</th>
<th>prepr.</th>
<th>space</th>
<th>scale</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow$ basic $A^*$</td>
<td>−</td>
<td>+ +</td>
<td>+ +</td>
<td>+</td>
<td>[Hart et al. 68]</td>
</tr>
<tr>
<td>bidirected</td>
<td>−</td>
<td>+ +</td>
<td>+ +</td>
<td>+</td>
<td>[Pohl 71]</td>
</tr>
<tr>
<td>$\bigtriangledown$ heuristic hwy hier.</td>
<td>+</td>
<td>+ +</td>
<td>+</td>
<td>+</td>
<td>[commercial]</td>
</tr>
<tr>
<td>$\bigtriangledown$ separator hierarchies</td>
<td>o</td>
<td>?</td>
<td>−</td>
<td>−</td>
<td>[several groups 02]</td>
</tr>
<tr>
<td>$\bigtriangledown\rightarrow$ geometric containers</td>
<td>+ +</td>
<td>− −</td>
<td>+</td>
<td>+</td>
<td>[Wagner et al. 03]</td>
</tr>
<tr>
<td>$\bigtriangledown\rightarrow$ bitvectors</td>
<td>+ +</td>
<td>−</td>
<td>o</td>
<td>−</td>
<td>[Lauther... 04]</td>
</tr>
<tr>
<td>$\rightarrow$ landmarks</td>
<td>+</td>
<td>+ +</td>
<td>−</td>
<td>−</td>
<td>[Goldberg et al. 04]</td>
</tr>
<tr>
<td>$\bigtriangledown\rightarrow$ landmarks + reaches</td>
<td>+ +</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>[Goldberg et al. 06]</td>
</tr>
<tr>
<td>$\bigtriangledown$ highway hierarchies</td>
<td>+ +</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>here</td>
</tr>
</tbody>
</table>

$\rightarrow$ direct towards target  $\bigtriangledown$ exploit hierarchy
DIJKSTRA’s Algorithm

not practicable for large road networks
(e.g. Western Europe: \( \approx 18\,000\,000 \) nodes)

improves the running time, but still too slow
Naive Route Planning

1. Look for the next reasonable motorway
Naive Route Planning

1. Look for the next reasonable motorway
2. Drive on motorways to a location close to the target
Naive Route Planning

1. Look for the next reasonable motorway
2. Drive on motorways to a location close to the target
3. Search the target starting from the motorway exit
Commercial Approach

**Heuristic Highway Hierarchy**

- □ complete search in local area
- □ search in (sparser) highway network
- □ iterate \(\leadsto\) highway hierarchy

Defining the highway network:

use road category (highway, federal highway, motorway, ...)

+ manual rectifications

- □ delicate compromise

- □ speed \(\leftrightarrow\) accuracy
Our Approach

**Exact Highway Hierarchy**

- complete search in local area
- search in (sparser) highway network
- iterate $\sim$ highway hierarchy

Defining the highway network:

- minimal network that preserves all shortest paths

- fully automatic (just fix neighborhood size)
- uncompromisingly fast
Our Approach

**Exact Highway Hierarchy**

- **complete** search in **local** area
- **search in (sparser)** highway network
- **contract network**, e.g.,
- **iterate** ⇝ **highway hierarchy**
A Meaning of “Local”

- choose neighbourhood radius $r(s)$
  e.g. distance to the $H$-closest node for a fixed parameter $H$

- define neighbourhood of $s$:
  \[ \mathcal{N}(s) := \{ v \in V \mid d(s, v) \leq r(s) \} \]

- example for $H = 5$
Edge \((u, v)\) belongs to highway network iff there are nodes \(s\) and \(t\) s.t.

\[
\begin{align*}
\therefore & (u, v) \text{ is on the "canonical" shortest path from } s \text{ to } t \\
\text{and} & \\
\therefore & v \notin \mathcal{N}(s) \\
\text{and} & \\
\therefore & u \notin \mathcal{N}(t)
\end{align*}
\]
Canonical Shortest Paths

$S \mathcal{P}$ : Set of shortest paths

$S \mathcal{P}$ canonical $\iff$

$$\forall P = \langle s, \ldots, s', \ldots, t', \ldots, t \rangle \in S \mathcal{P} : \langle s' \rightarrow t' \rangle \in S \mathcal{P}$$
Canonical Shortest Paths

(a) Construction, started from $s_0$.

(b) Construction, started from $s_1$. 

\[ \mathcal{N}(s_0) \quad \mathcal{N}(s_1) \quad \mathcal{N}(t_1) \quad \mathcal{N}(s_1) \quad \mathcal{N}(u') \quad \mathcal{N}(t_0) \]
(c) Result of the construction.
Contraction

highway nodes and edges
Contraction

bypass node
Contraction

shortcuts
Contraction
Contraction

core
Contraction

Which nodes should be bypassed?

Use some heuristic taking into account

- the number of shortcuts that would be created and
- the degree of the node.
Construction

Example: Western Europe, bounding box around Karlsruhe
Complete Road Network
Highway Hierarchy: Level 1 and Level 2
Fast Construction

Phase 1: Construction of Partial Shortest Path Trees

For each node $s_0$, perform an SSSP search from $s_0$.

☐ A node’s state is either active or passive.

☐ $s_0$ is active.

☐ A node inherits the state of its parent in the shortest path tree.

☐ If the abort condition is fulfilled for a node $p$, $p$’s state is set to passive.

The search is aborted when all queued nodes are passive.
Abort Condition:

Path $P'$:

$p$ is set to passive iff

$$s_1 \prec p \land p \not\in \mathcal{N}(s_1) \land s_0 \not\in \mathcal{N}(p) \land |P' \cap \mathcal{N}(s_1) \cap \mathcal{N}(p)| \leq 1$$
**Fast Construction**

Abort Condition: Efficient Testing.

Equivalent Formulation:

$p$ is set to passive iff

\[ d(s_1, \bar{v}) \leq r(s_1) < d(s_1, w) \land \]
\[ \bar{v} \prec p \land d(\bar{u}, p) > r(p) \]
Theorem:
The tree roots and leaves encountered in Phase 1 witness all highway edges.
The highway edges can be found in time linear in the tree sizes.
Fast Construction

Problem: very long edges, e.g. ferries
Solution: An active node $v$ is declared to be a maverick if

$$d(s_0, v) > f \cdot r(s_0).$$

When all active nodes are mavericks, the search from passive nodes is no longer continued.

*$\supseteq$* superset of the highway network
Choose neighborhood sizes such that levels shrink geometrically
\[ \rightsquigarrow \text{linear space consumption} \]

**Arbitrarily Small Constant Factor (not implemented):**

- Large \( H_0 \) \[ \rightsquigarrow \] large level-0 radius \[ \rightsquigarrow \] small higher levels
- No \( r(\cdot) \) needed for level-0 search (under certain assumptions)
- Mapping to next level by hash table
Bidirectional version of Dijkstra’s Algorithm

Restrictions:

☐ Do not leave the neighbourhood of the entrance point to the current level.

Instead: switch to the next level.

☐ Do not enter a component of bypassed nodes.
Query (bidir. Dijkstra I)

Operations on two priority queues $\overset{\rightarrow}{Q}$ and $\overset{\leftarrow}{Q}$:

- void `insert`(nodeID, key)
- void `decreaseKey`(nodeID, key)
- nodeID `deleteMin()`

Node $u$ has key $\delta(u)$

(tentative) distance from the respective source node
Query (bidir. Dijkstra II)

query(s,t) {
    $\overrightarrow{Q}$.insert(s, 0); $\overleftarrow{Q}$.insert(t, 0);
    while ($\overrightarrow{Q} \cup \overleftarrow{Q} \neq \emptyset$) do {
        $\Rightarrow \in \{\overrightarrow{}, \overleftarrow{\}}$; //select direction
        $\overleftarrow{\Rightarrow}$
        $u := Q$.deleteMin();
        relaxEdges($\Rightarrow$, u);
    }
}
relaxEdges(↦, \(u\)) {

    foreach \(e = (u, v) \in \bar{E}\) do {
        \(k := \delta(u) + w(e);\)

        if \(v \in Q\) then \(Q\).decreaseKey(\(v, k\)); else \(Q\).insert(\(v, k\));
    }
}

Query (bidir. Dijkstra III)
Query (Hwy I)

Operations on two priority queues $\overrightarrow{Q}$ and $\overleftarrow{Q}$:

- void `insert`(nodeID, key)
- void `decreaseKey`(nodeID, key)
- nodeID `deleteMin()`

Node $u$ has key $(\delta(u), \ell(u), \text{gap}(u))$

- (tentative) distance from the respective source node
- search level
- gap to the next neighbourhood border

Lexicographical order: $<, >, <$
query\( (s, t) \) {
    \( \overrightarrow{Q}.\text{insert}(s, (0, 0, r_0^\rightarrow(\text{s}))) \); \( \overleftarrow{Q}.\text{insert}(t, (0, 0, r_0^\leftarrow(\text{t}))) \);
    \textbf{while} (\( \overrightarrow{Q} \cup \overleftarrow{Q} \) \( \neq \emptyset \)) \textbf{do} { 
        \( \Rightarrow \in \{\overrightarrow{\text{Q}}, \overleftarrow{\text{Q}}\} \); //select direction 
        \( \Rightarrow u := Q.\text{deleteMin}() \);
        relaxEdges(\( \Rightarrow, u \));
    }
}
relaxEdges(↦, u) {
    foreach e = (u, v) ∈ ⃗E do {
        gap := gap(u);
        if gap = ∞ then gap := r↦ℓ(u); // leave component
        for (ℓ := ℓ(u); w(e) > gap; ℓ++, gap := r↦ℓ(u)); // go “upwards”
        if ℓ(e) < ℓ then continue; // Restriction 1
        if e “enters a component” then continue; // Restriction 2
        k := (δ(u) + w(e), ℓ, gap−w(e));
        if v ∈ ⃗Q then ⃗Q.decreaseKey(v, k); else ⃗Q.insert(v, k);
    }
}
Query

Restriction 1

\[ N_0^{-}(s) \]

gap(s)

\[ s \rightarrow N_1^{-}(s_1) \]

\[ s_1 \]

\[ s \rightarrow N_2^{-}(s_2) \]

\[ s_2 = s_2' \]

Restriction 2

\[ N_1^{-}(s_1') \]

\[ N_2^{-}(s_2') \]

shortcut
Theorem:

We still find the shortest path.
Example: from Karlsruhe, Am Fasanengarten 5
to Palma de Mallorca
Sanders/Schultes: Route Planning
Bounding Box: 20 km  Level 0  Search Space
Bounding Box: 20 km  Level 1
Sanders/Schultes: Route Planning

Bounding Box: 20 km   Level 2
Bounding Box: 80 km  Level 4  Search Space
Optimisation: Distance Table

Construction:

- Construct fewer levels.  
  e.g. 4 instead of 9

- Compute an all-pairs distance table
  for the topmost level $L$.  
  $13\,465 \times 13\,465$ entries
Distance Table Query:

- Abort the search when all entrance points in the core of level $L$ have been encountered. $\approx 55$ for each direction
- Use the distance table to bridge the gap. $\approx 55 \times 55$ entries
Implementation

- C++
- heavy use of templates
- reused binary heap priority queue
- reused DIJKSTRA
- 5,750 lines of code
  incl. comments, excl. infrastructure (I/O, logging, ...
Implementation

- adjacency array graph representation
- forward + backward directed edges
- separate array for level specific data

```
<table>
<thead>
<tr>
<th>nodes</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td></td>
</tr>
<tr>
<td>r1, r2, r3, r4</td>
<td></td>
</tr>
<tr>
<td>r1, r2, r3</td>
<td></td>
</tr>
</tbody>
</table>
```

```plaintext
r0 → r1 → r2 → r3 → r4
          |       |
          v       v
```

```plaintext
r1, r2, r3 → r4
          v       v
```

```plaintext
r1, r2, r3 → edges
```

```plaintext
... → ... → ...
```
# Experiments

<table>
<thead>
<tr>
<th></th>
<th>W. Europe (PTV)</th>
<th></th>
<th>USA/CAN (PTV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes</td>
<td>18 029 721</td>
<td></td>
<td>18 741 705</td>
</tr>
<tr>
<td>#directed edges</td>
<td>42 199 587</td>
<td></td>
<td>47 244 849</td>
</tr>
<tr>
<td>construction [min]</td>
<td>15</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>search time [ms]</td>
<td>0.76</td>
<td></td>
<td>0.90</td>
</tr>
<tr>
<td>speedup (↔ DIJKSTRA)</td>
<td>8 320</td>
<td></td>
<td>7 232</td>
</tr>
</tbody>
</table>
Shrinking of the Highway Networks

Europe

#edges vs. level

- H = 40
- H = 60
- H = 80
Number of Levels

Europe

Memory Overhead per Node [byte]

Query Time [ms]

Memory Overhead

Query Time

Number of Levels

Europe
Queries – Time

Dijkstra Rank

Query Time [ms]

Europe
USA/CAN

2^{11} 2^{12} 2^{13} 2^{14} 2^{15} 2^{16} 2^{17} 2^{18} 2^{19} 2^{20} 2^{21} 2^{22} 2^{23} 2^{24}

0.0 0.5 1.0 1.5 2.0

0.0 0.5 1.0 1.5 2.0
Queries – Speedup

- Europe
- USA/CAN

Dijkstra Rank

Speedup (CPU Time)
Distance Table Accesses

Europe
USA/CAN
USA (Tiger)
Summary

- **exact** routes in large street networks e.g. $\approx 18$ million nodes
  $\Rightarrow$ quality advantage, advertisement argument

- **fast** search $< 1$ ms
  $\Rightarrow$ cheap, energy efficient processors in mobile devices
  $\Rightarrow$ low server load
  $\Rightarrow$ lots of room for additional functionality

- **fast** preprocessing $\approx 20$ min

- **low space consumption** $\ll$ data base

- **no manual** postprocessing of data
  $\Rightarrow$ less dependence on data sources

- **organic enhancement** of existing commercial solutions
Work in Progress

- computation of $M \times N$ distance tables
  joint work with [Knopp, Schulz]$^{1,2}$

- combination with a goal directed approach (landmarks)
  joint work with [Delling, Holzer]$^1$

---

$^1$Universität Karlsruhe, Algorithmics I Group

$^2$PTV AG
Future Work

- fast, local updates on the highway network (e.g. for traffic jams)
- implementation for mobile devices (flash access . . .)
- multi-criteria shortest paths
  joint work with Müller-Hannemann, Schnee
- flexible objective functions

3 Technische Universität Darmstadt, Algorithmics Group
We help transforming technology into products: consulting . . .

Joint projects for further features
Road Network of Europe