Fast Shortest-path Distance Queries on Road Networks by Pruned Highway Labeling

Takuya Akiba (U Tokyo)
Yoichi Iwata (U Tokyo)
Ken-ichi Kawarabayashi (NII)
Yuki Kawata (U Tokyo)
Shortest-path Queries

Road Network

Query \((s, t)\)

Preprocess

Index

Distance \(d(s, t)\)

Goal

- Fast preprocessing time
- Small index size
- Fast query time
Labeling Methods

[Coen+,02],[Abraham+,11],[Abraham+,12],[Delling+,13]

Precompute label $L(v)$ for each vertex $v$

$L(1) = \{\ldots\}$
$L(2) = \{\ldots\}$

$\ldots$
$L(V) = \{\ldots\}$

Answer distance using only labels

Query $(s,t)$
Label $L(s)$
Distance $d(s,t)$

Widely used framework on both road networks and complex networks

- Hub-based Labeling [Abraham+,11]
- Highway-based Labeling Ours
- etc.
Hub-based Labeling [Abraham+, 11]

Label $L(v)$ is a set of pairs $(w, d(v, w))$

$L(s) = \{(v, \delta), ...\}$

Vertex
Distance
Hub-based Labeling [Abraham+, 11]

Label $L(v)$ is a set of pairs $(w, d(v, w))$

$$Query(s, t, L) = \min\{d(s, v) + d(v, t) | (v, d(s, v)) \in L(s), (v, d(t, v)) \in L(t)\}$$
Labeling for Paths

Can cover all shortest paths that hit the path

Use highways efficiently
Labeling for Paths

Similarity with distance oracles for planar graphs [Thorup,04]

Separate the graph by paths
Separation on Our Method

USA: |V|=24M, |E|=58M   Europe: |V|=18M, |E|=42M

Average Number of Common Paths in two Labels

Dijkstra Rank

Several common paths

USA: |V|=24M, |E|=58M   Europe: |V|=18M, |E|=42M
Highway-based Labeling
**Highway Decomposition**

**Definition 3.1**
A highway decomposition of a given graph $G$ is a family of ordered sets of vertices $\mathcal{P} = \{P_1, P_2, ..., P_N\}$ such that,

1. $P_i = (p_{i,1}, p_{i,2}, ..., p_{i,l_i})$ is a shortest path between two vertices $p_{i,1}$ and $p_{i,l_i}$,
2. $P_i \cap P_j = \emptyset$ for any $i$ and $j$ ($i \neq j$), and
3. $P_1 \cup P_2 \cup \cdots \cup P_N = V$
Highway-based Labeling

Label $L(v)$ is a set of triples $(i, d(p_{i,1}, p_{i,j}), d(v, p_{i,j}))$

- Index of path
- Distance from $p_{i,1}$
- Distance

$L(s) = \{(1, \delta_p, \delta), ... \}$
Highway-based Labeling

Label $L(\nu)$ is a set of triples $(i, d(p_{i,1}, p_{i,j}), d(\nu, p_{i,j}))$

**Distance on the path**

$$d(p_{i,j}, p_{i,k}) = |d(p_{i,1}, p_{i,j}) - d(p_{i,1}, p_{i,k})|$$
Highway-based Labeling

Label $L(v)$ is a set of triples $(i, d(p_{i,1}, p_{i,j}), d(v, p_{i,j}))$

Distance on the path

$$Query(s, t, L) = \min \{ d(s, p_{i,j}) + d(p_{i,j}, p_{i,k}) + d(p_{i,k}, t) |$$
$$\quad (i, d(p_{i,1}, p_{i,j}), d(s, p_{i,j})) \in L(s),$$
$$\quad (i, d(p_{i,1}, p_{i,k}), d(t, p_{i,k})) \in L(t) \}$$
Advantages

Can cover all vertices on the path by storing a few vertices on the path

→ Small labels
Pruned Highway Labeling
Naive Highway Labeling

$L_0 \rightarrow L_1 \rightarrow L_2 \rightarrow \cdots \rightarrow L_N = L$

**Empty Labels**

**Label for each path $P_i$**

- Conduct the Dijkstra search from each vertex $p_{i,j}$
- Add the distance to visited vertices
  - $L_i(\nu) = L_{i-1}(\nu) \cap (i, d(p_{i,1}, p_{i,j}), d(\nu, p_{i,j}))$

$L(\nu)$ contains the distance from $\nu$ to all vertices
→ Slow preprocessing & Huge labels
Example

$L_1(p_{2,1})$ contains all vertices on path $P_1$

Dijkstra search from path $P_1$
**Pruned Highway Labeling**

\[ L'_0 \rightarrow L'_1 \rightarrow L'_2 \rightarrow \cdots \rightarrow L'_N = L' \]

- **Empty Labels**
- **Label for each path**

**Label for path** \( P_i \)

- Conduct the *pruned* Dijkstra search from each vertex \( p_{i,j} \)
- Add the distance to visited vertices
  - \( L'_i(v) = L'_{i-1}(v) \cap (i, d(p_{i,1}, p_{i,j}), d(v, p_{i,j})) \)

**Fast preprocessing & Small labels**
Pruned Dijkstra search

• Assume we visit a vertex $v$ from the vertex $p_{i,j}$ with distance $\delta$.

• If $\text{Query}(v, p_{i,j}, L'_i) \leq \delta$, we prune the search:
  – Don’t add the triple $(i, d(p_{i,1}, p_{i,j}), \delta)$ to $L'_i(v)$
  – Don’t check edges from the vertex $v$.
Example

Dijkstra search from path $P_1$

$L'_1(p_{2,1})$ contains only one vertex on path $P_1$
Most vertices (white vertices) are not visited by pruning (orange vertices).
Effect of Pruning

Most triples are added at the beginning

USA: $|V|=24M, |E|=58M$   Europe: $|V|=18M, |E|=42M$
Pruned Labeling as a Paradigm

- Shortest-path queries on complex networks [Akiba+, SIGMOD’13]
- Reachability queries [Yano+, CIKM’13]
- Shortest-path queries on road networks (This paper)

Powerful label computation algorithm
Highway Decomposition

Choose a path that hits many shortest paths at the beginning

- Prune many searches

Choose a path based on some heuristics

1. Speed of an edge
2. Shortest path tree &
   Largest number of descendants
3. Skip unimportant vertices

- Highways
- Better separation
- Small labels

Difficult
Real Example of a Path

Use highways efficiently
Separate the graph by paths
Experiments
## Experimental Results

| USA (|V|=24M,|E|=58M) | Preprocessing [h:m] | Space [GB] | Query [ns] |
|----------------|--------------------|-------------|------------|
| CH [Geisberger+,08] | 0:27 | 0.5 | 130000 |
| TNR [Bast+,07] | 1:30 | 5.4 | 3000 |
| TNR+AF [Bauer+,10] | 2:37 | 6.3 | 1700 |
| HL local [Abraham+,11] | 2:24(×12) *2 | 22.7 | 627 |
| HL global [Abraham+,11] | 2:35(×12) *2 | 25.4 | 266 |
| HL-15 local [Abraham+,12] | - | - | - |
| HL-∞ global [Abraham+,12] | - | - | - |
| HLC-15 [Delling+,13] | 0:53 | 2.9 | 2486 |
| PHL-1 | 0:29 | 16.4 | 941 |

*1 CH,TNR,TNR+AF: AMD Opteron 270 (2.0 GHz)  HL,HLC: Intel Xeon X5680 (3.33 GHz)  PHL: Intel Xeon X5670 (3.06 GHz)

*2 parallelized to use 12 cores
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>Preprocessing</th>
<th>Space [GB]</th>
<th>Query [ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Europe</strong> (</td>
<td>V</td>
<td>=18M,</td>
<td>E</td>
</tr>
<tr>
<td>CH [Geisberger+,08]</td>
<td>0:25</td>
<td>0.4</td>
<td>180000</td>
</tr>
<tr>
<td>TNR [Bast+,07]</td>
<td>1:52</td>
<td>3.7</td>
<td>3400</td>
</tr>
<tr>
<td>TNR+AF [Bauer+,10]</td>
<td>3:49</td>
<td>5.7</td>
<td>1900</td>
</tr>
<tr>
<td>HL global [Abraham+,11]</td>
<td>2:45(×12) *2</td>
<td>21.3</td>
<td>276</td>
</tr>
<tr>
<td>HL-15 local [Abraham+,12]</td>
<td>0:05(×12) *2</td>
<td>18.8</td>
<td>556</td>
</tr>
<tr>
<td>HL-∞ global [Abraham+,12]</td>
<td>6:12(×12) *2</td>
<td>17.7</td>
<td>254</td>
</tr>
<tr>
<td>HLC-15 [Delling+,13]</td>
<td>0:50</td>
<td>1.8</td>
<td>2554</td>
</tr>
<tr>
<td><strong>PHL-1</strong></td>
<td>0:34</td>
<td>14.9</td>
<td>1039</td>
</tr>
</tbody>
</table>

*1 CH,TNR,TNR+AF: AMD Opteron 270 (2.0 GHz) HL,HLC: Intel Xeon X5680 (3.33 GHz) PHL: Intel Xeon X5670 (3.06 GHz)  
*2 parallelized to use 12 cores
Conclusion

• Labeling for Paths
  – Effective use of highways
  – Similarity with distance oracles
  – Efficient preprocessing based on pruned labeling
    → frequent update

• Experiments
  – Faster preprocessing time
  – Sufficiently fast query time