Experimental Study on Speed-Up Techniques for Timetable Information Systems

Reinhard Bauer (rbauer@ira.uka.de)
Daniel Delling (delling@ira.uka.de)
Dorothea Wagner (wagner@ira.uka.de)

ITI Wagner, Universität Karlsruhe

ATMOS 2007: November, 16th 2007
The Dynamic Timetable Information Problem

given:
- timetable of *continental* railway + bus system
- delays

task:
- find connection between two stations with *minimum travel time* at certain time of a day
- *incorporating* current delays
- in general: use Dijkstra’s algorithm in graphs constructed from timetable data

problem:
- Dijkstra’s algorithm far too slow!
- several seconds of query time
- idea: use *speed-up* techniques
state of the art:

- **very** fast techniques for road networks
- fast techniques for *dynamic* scenarios

(some) open questions:

1. do speed-up techniques perform *similar* on timetable graphs?
2. which *model* should be preferred for usage of speed-up techniques?
3. how to *adapt* them to such graphs?
4. how to incorporate delays in timetable graphs?

here:

- check **1** and **2**
Outline

1. Motivation
2. Speed-Up Techniques
3. Timetable Information
4. Experiments
5. Conclusion
2. Speed-Up Techniques
Speed-up Techniques

observations:

- some nodes are more **important** than others
- Dijkstra also visits nodes in “wrong” direction

idea:

- **two-stage** algorithm:
  - offline: **preprocessing**, compute additional information
  - online: **speed-up** query with this additional information
idea [GoldbergHarrelson05]:

- preprocessing:
  - choose some landmarks from the graph ($\approx 16$)
  - compute distances from and to all landmarks
- search
  - use landmarks and triangle inequality to compute a lower bound to target
  - pushes search towards target

$\Rightarrow$ goal-directed search

remarks:

- known as ALT
- works in dynamic scenarios (talk in Utrecht, published at WEA’07)
Example: Dijkstra vs. ALT
Reach-Based Routing

idea [Gutman04]:
- preprocessing:
  - identify important nodes
  - important nodes are in the middle of long shortest paths
- search:
  - prune the search at unimportant nodes

remarks [Goldberg et. al. 06]:
- can be enriched by shortcuts (RE-algorithm)
- harmonizes well with ALT (REAL-algorithm)
Example: Dijkstra vs. REAL
idea [Lauther04, Möhring et. al. 05, 06]:

- preprocessing:
  - partition graphs in cells $c$
  - check for each edge $e$ and cell $c$ whether $e$ is important for $c$

- search:
  - only use important edges

remarks:

- long preprocessing times ...
Example: Dijkstra vs. Arc-Flags
18.0 million nodes
42.2 million edges
average running times of random queries

<table>
<thead>
<tr>
<th>technique</th>
<th>preprocessing</th>
<th>query time</th>
<th>speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>-</td>
<td>6890.0 ms</td>
<td>1</td>
</tr>
<tr>
<td>bidirected</td>
<td>-</td>
<td>2790.0 ms</td>
<td>2.5</td>
</tr>
<tr>
<td>Landmarks</td>
<td>15 min</td>
<td>53.6 ms</td>
<td>129</td>
</tr>
<tr>
<td>Reach</td>
<td>82 min</td>
<td>3.5 ms</td>
<td>1969</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>1028 min</td>
<td>1.6 ms</td>
<td>4306</td>
</tr>
<tr>
<td>Reach+Landmarks</td>
<td>143 min</td>
<td>1.1 ms</td>
<td>6264</td>
</tr>
<tr>
<td>HH</td>
<td>16 min</td>
<td>0.7 ms</td>
<td>9842</td>
</tr>
<tr>
<td>HPML</td>
<td>20 h</td>
<td>20 µs</td>
<td>344 500</td>
</tr>
<tr>
<td>Transit Nodes</td>
<td>8 h</td>
<td>5 µs</td>
<td>1 378 000</td>
</tr>
</tbody>
</table>
General Observations for Speed-Up Techniques

- rely on **static** network or only **minor** changes to the graph
- most techniques use **bidirectional** search
- fastest techniques exploit **hierarchical** properties
  - how to step **down** a hierarchy?
  - ⇒ some even do **not** work in unidirectional manner
- 2-core of a road network: \( \approx 65\% \) of original size
  - ⇒ recent techniques use **contraction**
  - ⇒ speed-up derives from:
    - hierarchical properties of the network
    - bidirectional search
    - contraction
    - exception: Arc-Flags
      - but for the price of **long** preprocessing times
3. Timetable Information
Timetable Information: Two Models

time-dependent:
- a node models a station
- an edge models a connection between two stations
- edge weight depends on time

time-expanded:
- a node models an event
- an edge models a direct connection between two events
- edge weights are static

⇒ arrival is not known in advance
⇒ bidirectional search prohibited
Pros and Contras (for Speed-Up Techniques)

**time-dependent approach:**
- size of the graphs
- easy dynamization
- no fast technique for time-dependent networks
- no bidirectional search

**time-expanded approach:**
- static edge weights
- easier adaption (?)
- size of the graphs
- 2-core: 100% of original size (65% for road networks)
- no bidirectional search
- target node not known in advance (ALT,REAL)
- complicated dynamization (?)
4. Experiments
Experiments

purpose:
- **check** performance of speed-up techniques on
  - time-expanded networks
  - condensed networks (time-dependent model with **static** edge weights)
- is it worth the effort to use **contraction-based** techniques?
- do we need **bidirectional** search?
  ⇒ determine most promising technique for further investigations

remark:
- queries between two random nodes
  ⇒ no valid connection in time-expanded networks
- but gives hints on the performance
Experiments

input:
- timetable graphs of real-world data sets
- time-expanded model and condensed model
- road networks of similar size for comparison
- in paper: further synthetic data sets

evaluate:
- time for preprocessing
- speed-up over Dijkstra’s algorithm
Road Networks vs. Time-Expanded Timetable Graphs

<table>
<thead>
<tr>
<th></th>
<th>road networks</th>
<th></th>
<th>railway network</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prep.</td>
<td>query</td>
<td>prep.</td>
<td>query</td>
</tr>
<tr>
<td></td>
<td>[min]</td>
<td>[spd]</td>
<td>[min]</td>
<td>[spd]</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BiDijkstra</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>uni ALT</td>
<td>23</td>
<td>21</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>ALT</td>
<td>23</td>
<td>87</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>uni Arc-Flags</td>
<td>976</td>
<td>102</td>
<td>1 008</td>
<td>7</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>1 952</td>
<td>2 326</td>
<td>2 016</td>
<td>63</td>
</tr>
<tr>
<td>RE</td>
<td>18</td>
<td>352</td>
<td>63</td>
<td>26</td>
</tr>
<tr>
<td>uni REAL</td>
<td>41</td>
<td>584</td>
<td>68</td>
<td>25</td>
</tr>
<tr>
<td>REAL</td>
<td>41</td>
<td>905</td>
<td>68</td>
<td>49</td>
</tr>
<tr>
<td>HH</td>
<td>4</td>
<td>1 974</td>
<td>12</td>
<td>30</td>
</tr>
</tbody>
</table>

- preprocessing similar
- speed-up **much less** on road networks
- bidirectional search **worse** than undirectional
### Long-Distance vs. Local Traffic (Time-Expanded)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>long distance</th>
<th>local</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prep.</td>
<td>query</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BiDijkstra</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>uni ALT</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>ALT</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>uni Arc-Flags</td>
<td>1008</td>
<td>7</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>2016</td>
<td>63</td>
</tr>
<tr>
<td>RE</td>
<td>63</td>
<td>26</td>
</tr>
<tr>
<td>uni REAL</td>
<td>68</td>
<td>25</td>
</tr>
<tr>
<td>REAL</td>
<td>68</td>
<td>49</td>
</tr>
<tr>
<td>HH</td>
<td>12</td>
<td>30</td>
</tr>
</tbody>
</table>

- Performance on local network for most techniques even worse.
Road Networks vs. Condensed Timetable Graph

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Road Network</th>
<th>Railway Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>0</td>
<td>3.12</td>
</tr>
<tr>
<td>BiDijkstra</td>
<td>0</td>
<td>1.63</td>
</tr>
<tr>
<td>uni ALT</td>
<td>6</td>
<td>0.53</td>
</tr>
<tr>
<td>ALT</td>
<td>6</td>
<td>0.21</td>
</tr>
<tr>
<td>uni Arc-Flags</td>
<td>20</td>
<td>0.17</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>40</td>
<td>0.06</td>
</tr>
<tr>
<td>RE</td>
<td>4</td>
<td>0.21</td>
</tr>
<tr>
<td>uni REAL</td>
<td>10</td>
<td>0.20</td>
</tr>
<tr>
<td>REAL</td>
<td>10</td>
<td>0.11</td>
</tr>
<tr>
<td>HH</td>
<td>6</td>
<td>0.05</td>
</tr>
</tbody>
</table>

- similar performance on road networks and condensed networks
5. Conclusion
speed-up techniques “fail” on time-expanded networks

- blow-up between time-expanded and time-dependent graphs cannot be compensated
- even without modification (which may cost performance)

⇒ try to tailor speed-up techniques to time-expanded networks

- road networks similar to condensed networks

⇒ good performance on condensed networks

⇒ use time-dependent approach (?)
Further Results in the Paper

- extensive experimental evaluation of speed-up techniques
- influence of metrics, contraction, and bidirectional search in road networks
- many metrics and types of graphs
Ongoing and Future Work

- develop a fast undirectional speed-up technique
  - avoid the bidirectional search problem
  - easier to use in timetable graphs
  - will be presented at ALENEX’08 (SHARC-Routing)
- develop fast time-dependent speed-up technique
  - approximate algorithm → ALENEX’08 (SHARC-Routing)
  - ongoing: exact algorithm
- model delays in timetable networks (cooperation with CTI)
- develop robust technique with respect to such delays (cooperation with CTI)
- tailor speed-up techniques to timetable graphs
- ...
- solution for the dynamic timetable information problem
Thank you for your attention!