Algorithmic Challenges in Multi-Modal Route Planning

10Y Google in Zurich Anniversary
Dorothea Wagner | May 19, 2014
Motivation

important application, e.g.,
- navigation systems for cars,
- Google Maps, Bing Maps, . . . ,
- timetable information.

focus of basic research on
- clean mathematical models,
- provable quality guarantees,
- rigorous performance evaluation.

Find methods for route planning in transportation networks with provably optimal solutions regarding the quality of the routes.
Problem

request:
- find the best connection in a transportation network

idea:
- network as graph \( G = (V, E) \)
- edge weights are travel times
- shortest paths in \( G \) equal quickest connections
- classic problem (Dijkstra)

problems:
- transport networks are huge
- Dijkstra too slow (> 1 second)
Speed-Up Techniques

observations:
- Dijkstra visits all nodes closer than the target
- unnecessary computations

idea:
- two-phase algorithm:
  - offline: compute additional data during preprocessing
  - online: speed-up query with this data
- 3 criteria: preprocessing time and space, speed-up over Dijkstra
Showpiece of Algorithm Engineering

Algorithmics

Design
Experiment
Implement
Analyze
Showpiece of Algorithm Engineering

- Realistic machine models
- Real-world Data

Design → Falsifiable Hypotheses

Analyze → Performance guarantees & algorithm dependability

Experiment → Implement

Dorothea Wagner – Algorithmic Challenges in Multi-Modal Route Planning
May 19, 2014
Speed-Up Techniques

many techniques:

- Arc-Flags [Lau04]
- Multi-Level Dijkstra [SWW00, HSW08]
  - Customizable Route Planning (CRP) [DGPW11]
- ALT: A*, Landmarks, Triangle Inequality [GH05, GW05]
- Reach [GKW07]
- Contraction Hierarchies (CH) [GSSD08]
- Transit Node Routing (TNR) [ALS13]
- Hub Labeling (HL) [ADGW12]
- ...
Shortcuts

[SWW99, SS05, GSSD08]

observation:
- nodes with low degree are not important

contract graph
- iteratively remove such nodes
- add shortcuts to preserve distances between non-removed nodes

query:
- bidirectional
- prune edges heading to less important nodes
idea: solely use contraction

approach:
- heuristically order nodes by "importance"
- contract nodes in that order
- node $v$ contracted by

\begin{verbatim}
forall edges $(u, v)$ and $(v, w)$ do
  if $(u, v, w)$ unique shortest path then
    add shortcut $(u, w)$ with weight $\text{len}(u, v) + \text{len}(v, w)$;
\end{verbatim}

- query only looks at edges to more important nodes
Example: CH Preprocessing
Example: CH Preprocessing

\begin{figure}
\begin{tikzpicture}
\node (1) at (0,0) [circle,draw] {1};
\node (2) at (-2,-2) [circle,draw] {2};
\node (3) at (2,-2) [circle,draw] {3};
\node (4) at (-1,2) [circle,draw] {4};
\node (5) at (1,2) [circle,draw] {5};
\node (6) at (0,-2) [circle,draw] {6};
\draw [blue] (2) -- (1) -- (3);
\draw (2) -- (4);
\draw (3) -- (5);
\draw (4) -- (6);
\draw (5) -- (6);
\end{tikzpicture}
\end{figure}
Example: CH Preprocessing
Example: CH Preprocessing
Example: CH Preprocessing
Example: CH Preprocessing
modified bidirectional Dijkstra

upward graph \( G^\uparrow := (V, E^\uparrow) \) with \( E^\uparrow := \{(u, v) \in E : u < v\} \)

downward graph \( G^\downarrow := (V, E^\downarrow) \) with \( E^\downarrow := \{(u, v) \in E : u > v\} \)

forward search in \( G^\uparrow \) and backward search in \( G^\downarrow \)
modified bidirectional Dijkstra

upward graph \( G^\uparrow := (V, E^\uparrow) \) with \( E^\uparrow := \{(u, v) \in E : u < v\} \)

downward graph \( G^\downarrow := (V, E^\downarrow) \) with \( E^\downarrow := \{(u, v) \in E : u > v\} \)

forward search in \( G^\uparrow \) and backward search in \( G^\downarrow \)
CH Query

- modified bidirectional Dijkstra
- upward graph \( G^\uparrow := (V, E^\uparrow) \) with \( E^\uparrow := \{ (u, v) \in E : u < v \} \)
- downward graph \( G^\downarrow := (V, E^\downarrow) \) with \( E^\downarrow := \{ (u, v) \in E : u > v \} \)
- forward search in \( G^\uparrow \) and backward search in \( G^\downarrow \)
modified bidirectional Dijkstra

upward graph \( G^\uparrow := (V, E^\uparrow) \) with \( E^\uparrow := \{ (u, v) \in E : u < v \} \)

downward graph \( G^\downarrow := (V, E^\downarrow) \) with \( E^\downarrow := \{ (u, v) \in E : u > v \} \)

forward search in \( G^\uparrow \) and backward search in \( G^\downarrow \)
question: What is a good contraction order?

- up to now: solely heuristical [GSSD08]
- no guarantees
question: What is a good contraction order?
- up to now: solely heuristical [GSSD08]
- no guarantees

WeakCH [BCRW13]
- balanced separator nodes are important
  → resulting CH is called weak
- $O(n^\alpha)$ separators $\rightarrow O(n^\alpha)$ nodes in the search space
- order is independent of metric
(Multi-Level) Overlays \cite{SWW00, HSW08}

**observation:** many (long-distance) paths share large subpaths

**idea:** percompute partial solutions

**overlay graph:**
- select important nodes (separators, path coverage, heuristic)
- compute shortcut-edges:
  - skip unimportant nodes
  - conserve distances to important nodes

**queries:**
- multi-level Dijkstra variant
- ignore edges towards less important nodes

analogous: hierarchies with several levels of nodes of varying importances
# Experimental Evaluation

**input:** road network of Europe

- approx. 18M nodes
- approx. 42M edges

---

### Preprocessing

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>—</td>
<td>—</td>
<td>2 550 000</td>
<td>—</td>
</tr>
<tr>
<td>ALT [GH05, GW05]</td>
<td>0:42</td>
<td>2.2</td>
<td>24 521</td>
<td>104</td>
</tr>
<tr>
<td>CRP [DGPW11]</td>
<td>≪ 0:01</td>
<td>&lt; 0.1</td>
<td>1 650</td>
<td>1 545</td>
</tr>
<tr>
<td>Arc-Flags [Lau04]</td>
<td>0:20</td>
<td>0.3</td>
<td>408</td>
<td>6 250</td>
</tr>
<tr>
<td>CH [GSSD08]</td>
<td>0:05</td>
<td>0.2</td>
<td>110</td>
<td>23 181</td>
</tr>
<tr>
<td>TNR [ALS13]</td>
<td>0:20</td>
<td>2.1</td>
<td>1.25</td>
<td>2 040 000</td>
</tr>
<tr>
<td>HL [ADGW12]</td>
<td>0:37</td>
<td>18.4</td>
<td>0.56</td>
<td>4 553 571</td>
</tr>
</tbody>
</table>

---

In use at Bing, Google, Tomtom, . . .

---

Dorothea Wagner – Algorithmic Challenges in Multi-Modal Route Planning

May 19, 2014
Experimental Evaluation

**input:** road network of Europe
- approx. 18M nodes
- approx. 42M edges

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra [Dij59]</td>
<td>—</td>
<td>—</td>
<td>2 550 000</td>
<td>—</td>
</tr>
<tr>
<td>ALT [GH05, GW05]</td>
<td>0:42</td>
<td>2.2</td>
<td>24 521</td>
<td>104</td>
</tr>
<tr>
<td>CRP [DGPW11]</td>
<td>≪0:01</td>
<td>&lt;0.1</td>
<td>1 650</td>
<td>1 545</td>
</tr>
<tr>
<td>Arc-Flags [Lau04]</td>
<td>0:20</td>
<td>0.3</td>
<td>408</td>
<td>6 250</td>
</tr>
<tr>
<td>CH [GSSD08]</td>
<td>0:05</td>
<td>0.2</td>
<td>110</td>
<td>23 181</td>
</tr>
<tr>
<td>TNR [ALS13]</td>
<td>0:20</td>
<td>2.1</td>
<td>1.25</td>
<td>2 040 000</td>
</tr>
<tr>
<td>HL [ADGW12]</td>
<td>0:37</td>
<td>18.4</td>
<td>0.56</td>
<td>4 553 571</td>
</tr>
</tbody>
</table>

In use at Bing, Google, Tomtom, ...
idea:

- CH topology is the same regardless of metric
- quickly introduce new metric
From Theory to Practice: Customizable Contraction Hierarchies

[DSW14]

idea:
- CH topology is the same regardless of metric
- quickly introduce new metric

an edge in the CH
idea:
- CH topology is the same regardless of metric
- quickly introduce new metric

establish lower triangle inequality
idea:
- CH topology is the same regardless of metric
- quickly introduce new metric

establish lower triangle inequality
idea:
- CH topology is the same regardless of metric
- quickly introduce new metric

do this for all lower triangles
**Timetable Queries**

**input:** a timetable is a set of *(elementary)* connections

(IIR 2269, Karlsruhe Hbf, Pforzheim Hbf, 10:05, 10:23)  
(IIR 2269, Pforzheim Hbf, Muehlacker, 10:25, 10:33)  
(IIR 2269, Muehlacker, Vaihingen(Enz), 10:34, 10:40)  
(IIR 2269, Vaihingen(Enz), Stuttgart Hbf, 10:41, 10:57)  
...  
(IICE 791, Stuttgart Hbf, Ulm Hbf, 11:12, 12:06)  
(IICE 791, Ulm Hbf, Augsburg Hbf, 12:08, 12:47)  
(IICE 791, Augsburg Hbf, Muenchen Hbf, 12:49, 13:21)

with train-ID, departure stop, arrival stop, departure time and arrival time.

**also:** *(short)* footpaths for transfers, for example from the main train platforms to the subway platforms; minimum change times
Timetable Queries

- inherently time-dependent: discrete departure times
- more query scenarios:
  - depart now: earliest arrival time?
  - depart later: earliest travel time?
  - multi-criteria: number of transfers, price, ...
- different network structure: less hierarchical, less well-separated, very different schedules at night, ...

![Diagram showing travel time vs. departure time](image)
## Connection Scan (CSA) [DPSW13]

**Output:** earliest arrival time  
**Input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>...</td>
</tr>
</tbody>
</table>

Elementary connections ordered by departure time

<table>
<thead>
<tr>
<th>dep.stop</th>
<th>arr.stop</th>
<th>dep.time</th>
<th>arr.time</th>
<th>dep.stop</th>
<th>arr.stop</th>
<th>dep.time</th>
<th>arr.time</th>
<th>dep.stop</th>
<th>arr.stop</th>
<th>dep.time</th>
<th>arr.time</th>
<th>...</th>
</tr>
</thead>
</table>

*missing in the example: footpaths and minimum change times*
**Connection Scan (CSA)** [DPSW13]

**output:** earliest arrival time  
**input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>...</td>
</tr>
</tbody>
</table>

**elementary connections ordered by departure time**

<table>
<thead>
<tr>
<th></th>
<th>dep: 1</th>
<th>arr: 3</th>
<th>9:00</th>
<th>9:25</th>
<th>dep: 3</th>
<th>9:15</th>
<th>9:45</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>9:40</th>
<th>9:55</th>
<th>...</th>
</tr>
</thead>
</table>

*missing in the example: footpaths and minimum change times*
## Connection Scan (CSA) \[\text{[DPSW13]}\]

**Output:** earliest arrival time  
**Input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>\cdots</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>(\infty)</td>
<td>8:00</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>\cdots</td>
</tr>
</tbody>
</table>

### Elementary Connections Ordered by Departure Time

<table>
<thead>
<tr>
<th></th>
<th>dep: 1</th>
<th>arr: 3</th>
<th>9:00</th>
<th>9:25</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>9:15</th>
<th>9:45</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>9:40</th>
<th>9:55</th>
<th>\cdots</th>
</tr>
</thead>
</table>

*missing in the example: footpaths and minimum change times*
**Connection Scan (CSA)** [DPSW13]

**Output:** earliest arrival time

**Input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
<td>+∞</td>
</tr>
</tbody>
</table>

**Elementary connections ordered by departure time**

<table>
<thead>
<tr>
<th>dep:</th>
<th>arr:</th>
<th>dep:</th>
<th>arr:</th>
<th>dep:</th>
<th>arr:</th>
<th>dep:</th>
<th>arr:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>3:</td>
<td>3:</td>
<td>4:</td>
<td>3:</td>
<td>4:</td>
<td>3:</td>
<td>4:</td>
</tr>
</tbody>
</table>

*missing in the example: footpaths and minimum change times*
Connection Scan (CSA) \([\text{DPSW13}]\)

**output:** earliest arrival time  
**input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>(\infty)</td>
<td>8:00</td>
<td>(\infty)</td>
<td>9:25</td>
<td>(\infty)</td>
<td>(\infty)</td>
</tr>
</tbody>
</table>

*missing in the example: footpaths and minimum change times*

Dorothea Wagner – Algorithmic Challenges in Multi-Modal Route Planning  
May 19, 2014
**Connection Scan (CSA)** [DPSW13]

**output:** earliest arrival time  
**input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>+∞</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

*missing in the example: footpaths and minimum change times*
**Connection Scan (CSA) [DPSW13]**

**output:** earliest arrival time  
**input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>+∞</td>
</tr>
</tbody>
</table>

**elementary connections ordered by departure time**

<table>
<thead>
<tr>
<th></th>
<th>dep: 1</th>
<th>arr: 3</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>dep: 3</th>
<th>arr: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>dep</td>
<td>9:00</td>
<td>9:25</td>
<td>9:15</td>
<td>9:45</td>
<td>9:40</td>
<td>9:55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*missing in the example: footpaths and minimum change times*
**output:** earliest arrival time  
**input:** timetable, source stop, source time, target stop

<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>9:55</td>
</tr>
</tbody>
</table>

**elementary connections ordered by departure time**

<table>
<thead>
<tr>
<th>dep: 1</th>
<th>arr: 3</th>
<th>9:00</th>
<th>9:25</th>
<th>dep: 3</th>
<th>9:15</th>
<th>9:45</th>
<th>dep: 3</th>
<th>arr: 4</th>
<th>9:45</th>
<th>9:55</th>
</tr>
</thead>
</table>

*missing in the example: footpaths and minimum change times*
Connection Scan (CSA) [DPSW13]

**output:** earliest arrival time

**input:** timetable, source stop, source time, target stop

```
<table>
<thead>
<tr>
<th>stop ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest arrival time</td>
<td>+∞</td>
<td>8:00</td>
<td>+∞</td>
<td>9:25</td>
<td>9:55</td>
<td>...</td>
</tr>
</tbody>
</table>
```

*missing in the example: footpaths and minimum change times

faster than Dijkstra, better use of modern processor architectures
## Experimental Evaluation

**input: timetable**

- London: 5 M connections, 21 k stops
- Deutschland: 46 M connections, 252 k stops

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time [ms]</th>
<th>speed-up.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>London</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TE Dijkstra</td>
<td>44.8</td>
<td>—</td>
</tr>
<tr>
<td>TD Dijkstra</td>
<td>10.9</td>
<td>4.1</td>
</tr>
<tr>
<td>CSA</td>
<td>1.8</td>
<td>24.9</td>
</tr>
<tr>
<td><strong>DE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TE Dijkstra</td>
<td>2960.2</td>
<td>—</td>
</tr>
<tr>
<td>CSA</td>
<td>298.6</td>
<td>9.9</td>
</tr>
<tr>
<td>CSAccel</td>
<td>8.7*</td>
<td>340.2</td>
</tr>
</tbody>
</table>

Intel Xeon E5-2670, 2.6 GHz, 64 GiB DDR3-1600 RAM, 20 MiB L2 cache

*preprocessing: 30 min, 256.4 MiB
Vision: Navi for the World

Worldwide network composed of car, rail, flight, ...
Multiple Transportation Modes

**Problem:** unrestricted journeys allow arbitrary transfers
Multiple Transportation Modes

**Problem:** unrestricted journeys allow arbitrary transfers

![Diagram of transportation modes](image)
**Multiple Transportation Modes**

**problem:** unrestricted journeys allow arbitrary transfers

![Diagram showing the problem with private car and subway line connections between source s and destination t with restrictions indicated by X marks.](image)
Multiple Transportation Modes

**problem:** unrestricted journeys allow arbitrary transfers

![Diagram showing multiple transportation modes: subway line, private car, and unrestricted journeys.](image-url)
**problem:** unrestricted journeys allow arbitrary transfers
**problem:** unrestricted journeys allow arbitrary transfers

- not all sequences of transportation modes are reasonable
- preferred mode of transport varies between users

Diagram:
- subway line
- cycle hire

Source: Dorothea Wagner – Algorithmic Challenges in Multi-Modal Route Planning

May 19, 2014
„Label Constrained Shortest Path Problem“ (LCSPP)

- define alphabet of transportation mode
- finite-state automaton describes sequences of vehicles
- every path must fulfill the requirements imposed by the automaton
Solution

„Label Constrained Shortest Path Problem“ (LCSPP)

- define alphabet of transportation mode
- finite-state automaton describes sequences of vehicles
- every path must fulfill the requirements imposed by the automaton

![Finite-state automaton diagram]

algorithms for LCSPP

- Dijkstra on the product graph with the automaton works but is slow [BJM00]
- speed-up techniques: ANR [DPW09], SDALT [KLPC11]
- automaton as input during the query: UCCH [DPW12]
User-constrained CH (UCCH) [DPW12]

multi-modal CH:
- contraction introduces shortcuts with label sequences
- witness search depends on constraints
  requires a-priori knowledge of the constraint automata

**idea:** do not contract nodes with incident link-edges.

- contraction and witness search are limited to each modality
  ⇒ preprocessing independent of mode sequence constraints
Example: UCCH Preprocessing
preprocessing
- linked nodes are not contracted thus contained in the core
- shortcuts between core nodes preserve distances
  allows using the road network between rail stations

query
- CH search on the component
- label constrained search on the core
- engineering yields further improvement
Experimental Evaluation

networks:
road: europe & north america (50 M nodes, 125 M edges)
train: europe (31 k stops, 1.6 M connections)
flight: Star Alliance (1 172 airports, 28 k connections)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>road &amp; flight</th>
<th>Time [h:m]</th>
<th>Space [MiB]</th>
<th>Query</th>
<th>Time [ms]</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>33 862</td>
<td>1</td>
</tr>
<tr>
<td>ANR [DPW09]</td>
<td>3:04</td>
<td>14 050</td>
<td>1.07</td>
<td>31 551</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCCH [DPW12]</td>
<td>1:18</td>
<td>542</td>
<td>50 540</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>all three</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>35 261</td>
<td>1</td>
</tr>
<tr>
<td>ANR [DPW09]</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>UCCH [DPW12]</td>
<td>1:27</td>
<td>558</td>
<td>70.52</td>
<td>500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Intel Xeon E5430, 2.66 GHz, 32 GiB RAM, 12 MiB L2 cache
Solution?

Problems of LCSPP

- Subway line
- Cycle hire

Restrictions must be known in advance, but the user might not know them. Only a single (best?) journey is computed (no alternatives).

Goal: Compute a useful set of multimodal journeys.
Solution?

Problems of LCSPP

\[ s \quad ? \quad t \]
Solution?

Problems of LCSPP

- restrictions must be known in advance
- user might not know them
- only a single (best?) journey is computed (no alternatives)

**goal:** compute a *useful set* of multimodal journeys
New Approach [?]  

idea: compute multicriteria, multimodal Pareto sets

- optimize arrival time plus
- various (per mode of transport) „convenience criteria“
  for example # transfers (trains), walking time, taxi costs, etc.
**New Approach**

**idea:** compute multicriteria, multimodal Pareto sets

- optimize arrival time plus
- various (per mode of transport) “convenience criteria“
  for example # transfers (trains), walking time, taxi costs, etc.

**known problem:** Pareto set sizes explode in the number of criteria
Relevant Journeys

- 10 min of walking to arrival 10 sec earlier?
- 1 hour of bus drive to walk 10 sec less?
Relevant Journeys

- 10 min of walking to arrival 10 sec earlier?
- 1 hour of bus drive to walk 10 sec less?
- rate the journeys using fuzzy logic [FA04]
- journeys with a higher rating are more relevant
Relevant Journeys

- 10 min of walking to arrival 10 sec earlier?
- 1 hour of bus drive to walk 10 sec less?
- rate the journeys using fuzzy logic [FA04]
- journeys with a higher rating are more relevant
Reducing the Amount of Work

Problem: queries are slow (> 1 s)

many irrelevant journeys ⇒ can we avoid computing them?

Filter already during the algorithm

- **MCR-hf**: fuzzy filter
- **MCR-hb**: Pareto filter, but discrete criteria

Restricted walking (arbitrary heuristic)

- **MCR-\(tx-ry\)**: max \(x\) minutes of walking between vehicles and max. \(y\) at source/target

Reduce the dimension/number of criteria

- **MR-x**: increase for every \(x\) minutes of walking the #transfers by +1
Experimental Evaluation

London, multimodal:
- roads: 260 k nodes, 1.4 M edges
- subway, bus, tram, . . .
  21 k stops, 5 M connections
- 564 cycle hire station

criteria: arrival time, # transfers, walking time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Sol.</th>
<th>Time [ms]</th>
<th>Quality-6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Avg.</td>
</tr>
<tr>
<td>MCR</td>
<td>29.1</td>
<td>1438.7</td>
<td>100 %</td>
</tr>
<tr>
<td>MCR-hf</td>
<td>10.9</td>
<td>699.4</td>
<td>89 %</td>
</tr>
<tr>
<td>MCR-hb</td>
<td>9.0</td>
<td>456.7</td>
<td>91 %</td>
</tr>
<tr>
<td>MCR-t10-r15</td>
<td>13.2</td>
<td>885.0</td>
<td>30 %</td>
</tr>
<tr>
<td>MR-10</td>
<td>4.3</td>
<td>39.4</td>
<td>45 %</td>
</tr>
</tbody>
</table>

Intel Xeon E5-2670, 2.6 GHz, 64 GiB DDR3-1600 RAM, 20 MiB L2 cache
Experimental Evaluation

London, multimodal:
- roads: 260 k nodes, 1.4 M edges
- subway, bus, tram, . . .
  21 k stops, 5 M connections
- 564 cycle hire station

**criteria:** arrival time, # transfers, walking time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Sol.</th>
<th>Time [ms]</th>
<th>Quality-6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg.</td>
<td>Sd.</td>
</tr>
<tr>
<td>MCR</td>
<td>29.1</td>
<td>1438.7</td>
<td>100 %</td>
</tr>
<tr>
<td>MCR-hf</td>
<td>10.9</td>
<td>699.4</td>
<td>89 %</td>
</tr>
<tr>
<td>MCR-hb</td>
<td>9.0</td>
<td>456.7</td>
<td>91 %</td>
</tr>
<tr>
<td>MCR-t10-r15</td>
<td>13.2</td>
<td>885.0</td>
<td>30 %</td>
</tr>
<tr>
<td>MR-10</td>
<td>4.3</td>
<td>39.4</td>
<td>45 %</td>
</tr>
</tbody>
</table>

Intel Xeon E5-2670, 2.6 GHz, 64 GiB DDR3-1600 RAM, 20 MiB L2 cache
Conclusion

Summary

- Algorithm Engineering: combination of theory and practice
- (very) fast route planning on road and timetable networks
- multimodal route planning is more expensive
  - fast methods when only optimizing travel time
  - network offers many interesting trade-offs between criteria
  - multicriteria optimization useful, to allow the user to chose his journey
  - fuzzy filtering is a practical method to rate the journey relevance

Outlook

- Is the quality-formalization of multimodal journeys done?
- scalability: multimodal multicriteria for worldwide routing?
- additional questions: delay-robustness, park & ride, . . .?
Thank you for your attention!
Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck.
Hierarchical hub labelings for shortest paths.

Julian Arz, Dennis Luxen, and Peter Sanders.
Transit node routing reconsidered.

Reinhard Bauer, Tobias Columbus, Ignaz Rutter, and Dorothea Wagner.
Search-space size in contraction hierarchies.

Chris Barrett, Riko Jacob, and Madhav V. Marathe.
Formal-language-constrained path problems.

Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F. Werneck.
Computing and evaluating multimodal journeys.
Technical Report 2012-20, Faculty of Informatics, Karlsruhe Institute of Technology, 2012.

Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck.
Customizable route planning.
Edsger W. Dijkstra.
A note on two problems in connexion with graphs.

Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner.
Intriguingly simple and fast transit routing.

Daniel Delling, Thomas Pajor, and Dorothea Wagner.
Accelerating multi-modal route planning by access-nodes.

Julian Dibbelt, Thomas Pajor, and Dorothea Wagner.
User-constrained multi-modal route planning.

Julian Dibbelt, Ben Strasser, and Dorothea Wagner.
Customizable contraction hierarchies.
Technical report, ITI Wagner, Department of Informatics, Karlsruhe Institute of Technology (KIT), 2014.

Marco Farina and Paolo Amato.
A fuzzy definition of “optimality” for many-criteria optimization problems.
Andrew V. Goldberg and Chris Harrelson.
Computing the shortest path: A* search meets graph theory.

Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck.
Better landmarks within reach.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling.
Contraction hierarchies: Faster and simpler hierarchical routing in road networks.

Andrew V. Goldberg and Renato F. Werneck.
Computing point-to-point shortest paths from external memory.

Martin Holzer, Frank Schulz, and Dorothea Wagner.
Engineering multilevel overlay graphs for shortest-path queries.

Dominik Kirchler, Leo Liberti, Thomas Pajor, and Roberto Wolfler Calvo.
UniALT for regular language constraint shortest paths on a multi-modal transportation network.
Ulrich Lauther.
An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background.

Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis.
Efficient models for timetable information in public transportation systems.

Peter Sanders and Dominik Schultes.
Highway hierarchies hasten exact shortest path queries.

Ben Strasser and Dorothea Wagner.
Connection scan accelerated.

Frank Schulz, Dorothea Wagner, and Karsten Weihe.
Dijkstra's algorithm on-line: An empirical case study from public railroad transport.

Frank Schulz, Dorothea Wagner, and Karsten Weihe.
Dijkstra’s algorithm on-line: An empirical case study from public railroad transport.