Algorithmic Aspects of Route Planning

escar – Embedded Security in Cars
Dorothea Wagner | Hamburg, November 19, 2014
Motivation

Important application, e.g.,
- Navigation systems for cars
- Google Maps, Bing Maps, ...
- Timetable information

Many commercial systems
- Use heuristic methods
- Consider “reasonable” part of the network
- Have no quality guarantees

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
- Many requests in a hardly changing network

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

Falsifiable Hypotheses

Design
Experiment
Implement
Analyze
Showpiece of Algorithm Engineering

Falsifiable Hypotheses

Design

Experiment

Analyze

Implement

Realistic scenarios

Real-world data

Performance guarantees & practical algorithms

Dorothea Wagner – Algorithmic Aspects of Route Planning
Hamburg, November 19, 2014
History I

Phase I: Theory (1959 - 1999):
- Improve theoretical worst-case running time
- By introduction of better data structures
- Bidirectional search, A*-search (goal-directed)

Phase II: Speed-up techniques (1999 - 2005):
- Two approaches: goal-directed and hierarchical approach
- Improvement on this for several inputs

Phase III: Road networks (2005 - 2008):
- Focus on continent-sized road networks
- DIMACS challenge in 2006
- Speed-up factors in range of several millions over Dijkstra
History II

Phase IV: Towards more realistic scenarios (2008-2012):
- Time-dependency, multicriteria, alternative routes, . . .
- Timetable information
- Back to theory: why do things work?

Now: New challenges (since 2012):
- Other metrics, e.g., energy consumption
- Customizability (supporting user-centric route planning)
- Multimodal
Exploiting Shortcuts

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 less important nodes
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order

![Diagram of a graph with nodes and edges labeled from 1 to 4, showing the order in which nodes are contracted.](image)
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

Idea: Solely use contraction

Approach: Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”

- Contract nodes in that order

![Diagram of Contraction Hierarchies](image-url)
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

**Idea:** Solely use contraction

**Approach:** Heuristically order nodes by “importance”
- Contract nodes in that order
Contraction Hierarchies

Idea: Solely use contraction

Approach: Heuristically order nodes by “importance”
  - Contract nodes in that order

query only looks at edges to more important nodes
(Multi-Level) Overlays

**Observation:** Many (long-distance) paths share large subpaths

**Idea:** Precompute partial solutions

**Overlay graph:**
- Partition graph
- Compute **shortcuts** between boundary nodes:
  - Conserve distances

**Queries:**
- Multi-level Dijkstra variant
- Ignore edges towards less important nodes

Analogous: Hierarchies with several levels of increasing node importance
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
- A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$

Labels must fulfill cover property:
for every $s$, $t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$

$s$–$t$ query:
Find node $v \in L(s) \cap L(t)$... that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$

Observations:
- Very simple query (can even be implemented in SQL)
- Query performance depends only on label sizes
- The "magic" lies in computing a small labeling efficiently
Hub Labeling

Preprocessing:

- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$
Hub Labeling

Preprocessing:

- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$
Hub Labeling

Preprocessing:

- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$

$s-t$ query:

- Find node $v \in L(s) \cap L(t)$
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
- A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through
  the intersection of $L(s) \cap L(t)$

$s$–$t$ query:
- Find node $v \in L(s) \cap L(t)$ ...
- ... that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s$, $t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$

$s$–$t$ query:
- Find node $v \in L(s) \cap L(t)$ . . .
- . . . that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
- A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$

$s-t$ query:
- Find node $v \in L(s) \cap L(t)$ . . .
- . . . that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through the intersection of $L(s) \cap L(t)$

$s$–$t$ query:
- Find node $v \in L(s) \cap L(t)$ . . .
- . . . that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$
Hub Labeling

Preprocessing:
- For each node $u$, compute label $L(u)$
  - A set of hub nodes $v$ and their distance $\text{dist}(u, v)$ to $u$
- Labels must fulfill cover property:
  for every $s, t$-pair, the shortest path goes through
  the intersection of $L(s) \cap L(t)$

$s$–$t$ query:
- Find node $v \in L(s) \cap L(t)$ ...
- ... that minimizes $\text{dist}(s, v) + \text{dist}(v, t)$

Observations:
- Very simple query (can even be implemented in SQL)
- Query performance depends only on label sizes
- The “magic” lies in computing a small labeling efficiently
## Experimental Evaluation

**Input:** Road network of Europe
- Approx. 18M nodes
- Approx. 42M edges

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Preprocessing</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ALT</td>
<td>0:42</td>
<td>2.2</td>
</tr>
<tr>
<td>CRP</td>
<td>1:00</td>
<td>0.5</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>0:20</td>
<td>0.3</td>
</tr>
<tr>
<td>CH</td>
<td>0:05</td>
<td>0.2</td>
</tr>
<tr>
<td>TNR</td>
<td>0:20</td>
<td>2.1</td>
</tr>
<tr>
<td>HL</td>
<td>0:37</td>
<td>18.8</td>
</tr>
</tbody>
</table>

In use at Bing, Google, Tomtom, ...
## Experimental Evaluation

**Input:** Road network of Europe
- Approx. 18M nodes
- Approx. 42M edges

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Preprocessing</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ALT</td>
<td>0:42</td>
<td>2.2</td>
</tr>
<tr>
<td>CRP</td>
<td>1:00</td>
<td>0.5</td>
</tr>
<tr>
<td>Arc-Flags</td>
<td>0:20</td>
<td>0.3</td>
</tr>
<tr>
<td>CH</td>
<td>0:05</td>
<td>0.2</td>
</tr>
<tr>
<td>TNR</td>
<td>0:20</td>
<td>2.1</td>
</tr>
<tr>
<td>HL</td>
<td>0:37</td>
<td>18.8</td>
</tr>
</tbody>
</table>

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

More realistic metrics:
- Turn costs, electro mobility
- Points of interests (nearest POIs, shortest via-POIs)
- User customizable metrics
  - e.g., height restrictions, avoid freeways, eco-friendliness, . . .
- Fast customization time per metric
- Very small space overhead

Multimodal networks:
- Change the type of transportation during the journey
- Allow only “reasonable” transfers
- Several constraints to the shortest path
- Multicriteria
Route Planning for Electric Vehicles

Electric vehicles:
- Future means of transportation
- Run on regenerative energy sources

But:
- Restricted battery capacity
- Long recharging times
- “Range anxiety”

⇒ Consider energy consumption in route planning applications

Task: Given start and destination in a road network, find the route that minimizes energy consumption.
Energy-Optimal Routes

Challenges:
- Negative edge weights (recuperation)
- Battery constraints (no over-, undercharging)

Energy consumption depends on battery state-of-charge (at the start):

\[ f_c(b) \]

\[ b \]

\[ f_c(b) \]

\[ b \]

\[ f_c(b) \]

\[ b \]

\[ f_c(b) \]

\[ b \]

\[ f_c(b) \]

\[ b \]
Energy-Optimal Routes

Challenges:

- Negative edge weights (recuperation)
- Battery constraints (no over-, undercharging)

Energy consumption depends on battery state-of-charge (at the start):
Energy-Optimal Routes

Challenges:
- Negative edge weights (recuperation)
- Battery constraints (no over-, undercharging)

Energy consumption depends on battery state-of-charge (at the start):

\[ f_c(b) \]

\[ M \]

\[ 0 \]

\[ b \]
Energy-Optimal Routes

Challenges:

- Negative edge weights (recuperation)
- Battery constraints (no over-, undercharging)

Energy consumption depends on battery state-of-charge (at the start):

![Graph with nodes and edges labeled with numbers.](image)
Energy-Optimal Routes

Requirements for speedup techniques:
- Shortcuts are functions, not scalar values
- User-dependent consumption profiles (⇒ custom metrics)

Experiments:
- Energy-optimal paths: 63 % extra time
- Fastest paths: 62 % extra energy
⇒ Energy-optimal routes: follow slow roads

Current and future work:
- Consider trade-off between travel time and energy consumption
  - Energy can be saved driving below speed limit
- Integration of charging stations into route planning
  - Different types of charging stations (super chargers, switching stations)
Custom Metrics

Problem

- Preprocessing is metric-dependent
- State-of-the-art algorithms tailored to travel time heavily exploit ‘hierarchy’ of road categories

Naive solution

- Compute preprocessing for each metric, e.g.
  - Distance
  - Pedestrian
  - Travel time, but don’t use toll roads
  - Travel time, avoid left turns, height restrictions, avoid tolls, . . .
- Preprocessing and query time increase significantly
- Higher space overhead
Customizable Contraction Hierarchies

Idea:
- CH topology is the same regardless of metric
- Quickly introduce new metric
Customizable Contraction Hierarchies

Idea:
- CH topology is the same regardless of metric
- Quickly introduce new metric

an edge in the CH
Customizable Contraction Hierarchies

Idea:
- CH topology is the same regardless of metric
- Quickly introduce new metric

establish lower triangle inequality
Customizable Contraction Hierarchies

Idea:

- CH topology is the same regardless of metric
- Quickly introduce new metric

establish lower triangle inequality
Customizable Contraction Hierarchies

Idea:
- CH topology is the same regardless of metric
- Quickly introduce new metric

do this for all lower triangles
Worldwide network composed of car, rail, flight, ...
Multimodal Routing

Up to now, research mostly on **uni-modal routing**
- Restricted to **one** transportation network
- Time-independent and time-dependent (separately)

What we **really** want is planning a journey by

- Choosing **source** and **destination**
- Desired means of transportation (car, train, flight, . . .)
- . . . in a **mixed network**
Access-Node Routing: Idea

**Assumption:** Road network only used in the beginning and end

**Observation:** Number of “relevant” entry points in the public transportation network is small

**Idea:** Compute for each road node its access-nodes and their distances
Problem: Unrestricted journeys allow arbitrary transfers
Multiple Transportation Modes

Problem: Unrestricted journeys allow arbitrary transfers

- Not all sequences of transportation modes are reasonable
Problem: Unrestricted journeys allow arbitrary transfers

- Not all sequences of transportation modes are reasonable
- Preferred mode of transport varies between users
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

Algorithms for LCSPP

- Dijkstra on the product graph with the automaton works but is slow
- Can be combined with speed-up techniques
- Automaton as input for the query: user-constrained shortest paths
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
Multicriteria Multimodal Routing

**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?
- Rate the journeys using fuzzy logic
- Journeys with a higher rating are more relevant

Dorothea Wagner – Algorithmic Aspects of Route Planning
Hamburg, November 19, 2014
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
- 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</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCR</td>
<td>29.1</td>
<td>1 438.7</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</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCR</td>
<td>29.1</td>
<td>1 438.7</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 networks
- Other metrics **besides travel time**
- Customizable
- **Multimodal routeplanning** 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
- For an overview of algorithmic techniques, see [BDG+14]
  Bast et al., *Route planning in transportation networks*. MSR-TR-2014-4

...many security aspects
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.

Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller–Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck.
Route planning in transportation networks.

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 multimodal journeys in practice.

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.  

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

Julian Dibbelt, Ben Strasser, and Dorothea Wagner.  
Customizable contraction hierarchies.  

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.  

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.

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