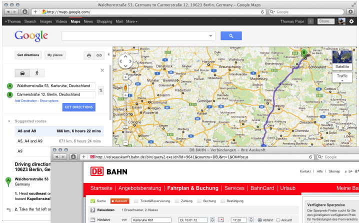
Route Planning Algorithms – New Results and Challenges

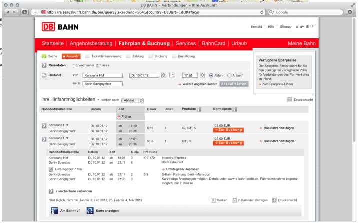
OR 2017, Berlin

Dorothea Wagner | September 8, 2017

KARLSRUHE INSTITUTE OF TECHNOLOGY – INSTITUTE OF THEORETICAL INFORMATICS – GROUP ALGORITHMICS

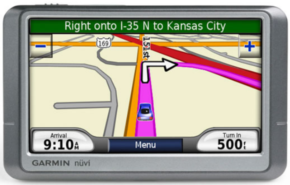
Motivation





Important applications, e.g.,

- Navigation systems for cars
- Apple Maps, Google Maps, Bing Maps, OpenStreetMap, . . .
- Timetable information



Navigation Device for the World

Worldwide network composed of car, rail, flight, . . .

Core Problem

Request:

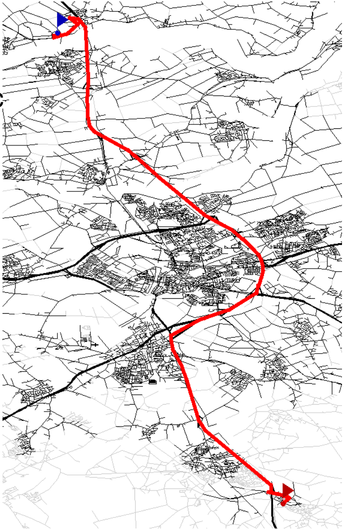
- Find the best connection in a transportation network w.r.t. some metric

Idea:

- Network as graph $G = (V, E)$
- Edge weights are according to metric
- Shortest paths in G equal best connections
- Classic problem (Dijkstra 1959)

Problems:

- Transport networks are huge
- Dijkstra too slow ($>$ 1 second)



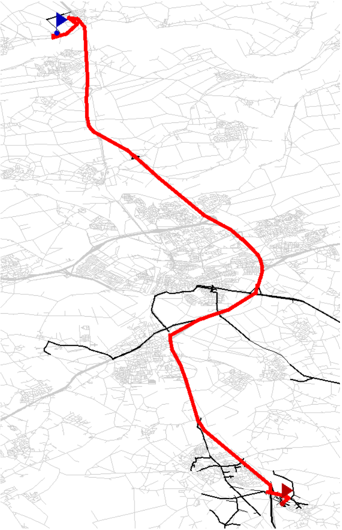
Speed-Up Techniques

Observations:

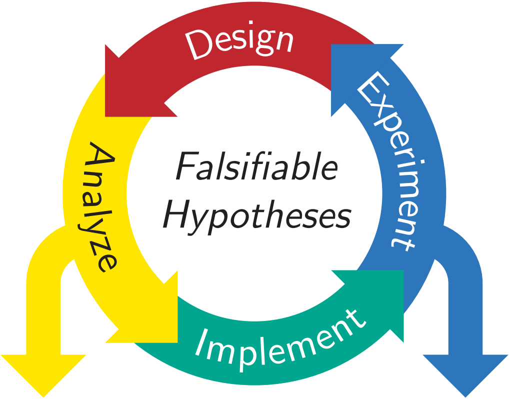
- Dijkstra visits all nodes closer than the target
- Unnecessary computations
- Many requests in a hardly changing network

Idea:

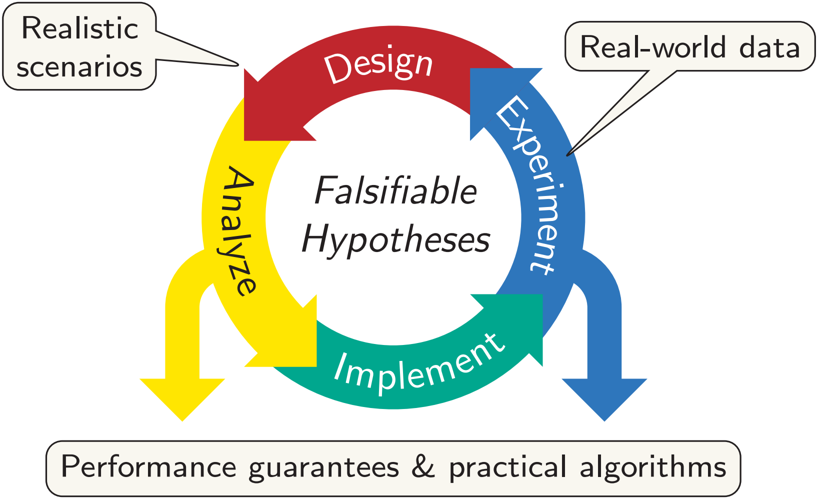
- Two-phase algorithm:
  - Offline: compute additional data during preprocessing
  - Online: speed-up query with this data
- 3 criteria: preprocessing time and space, speed-up over Dijkstra



Showpiece of Algorithm Engineering



Showpiece of Algorithm Engineering



Speed-Up Techniques

Many techniques tuned for continent-sized road networks:

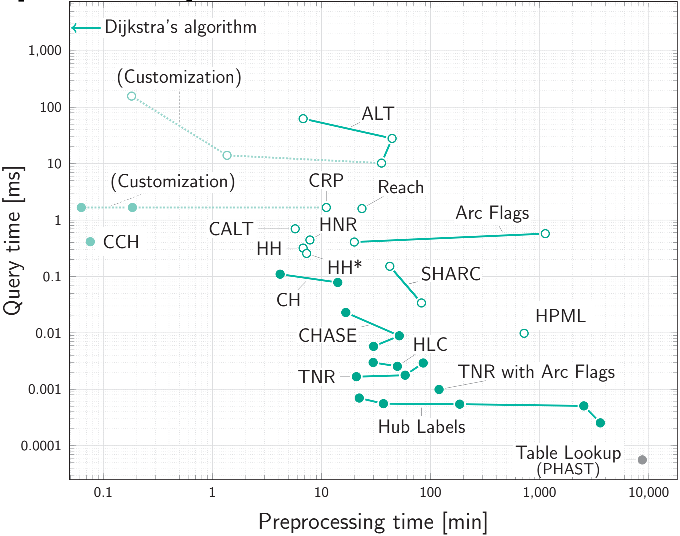
- Arc-Flags [2004,2006,2009,2013]
- Multi-Level Dijkstra [2000,2008,2009,2011]
- ALT: A*, Landmarks, Triangle Inequality [1968,2005,2012]
- Reach [2004,2007]
- Contraction Hierarchies (CH, CCH) [2008,2013,2014,2016]
- Transit Node Routing (TNR) [2007,2013]
- Hub Labeling (HL) [2003,2011,2013,2014]

Timetable information:

- Transfer Pattern [2010,2016]
- Raptor [2013]
- Connection Scan [2013,2014,2017]

Survey on “Route Planning in Transprotation Networks” [Bast et al.’16]

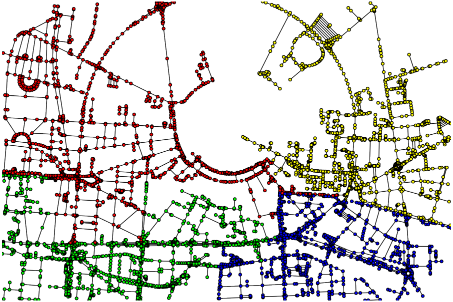
Speedup Techniques [Bast et al.'16]



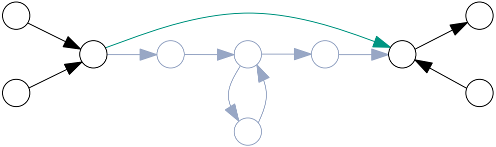
In use at Apple, Bing, Google, TomTom, ...

Some Ideas

- Partition Network

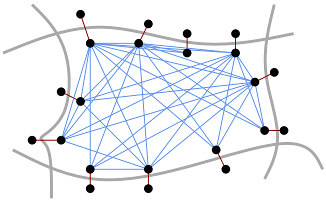


- Shortcuts



Overlays [Schulz et al.'00, Holzer et al.'08]

Observation: many (long-distance) paths share large subpaths
Idea: precompute partial solutions

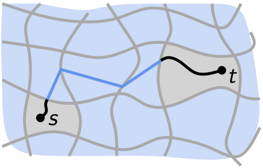


Overlay graph:

- Select important nodes (separators, path coverage, heuristic)
- Compute shortcut-edges:
  - Skip unimportant nodes
  - Conserve distances to important nodes

Queries:

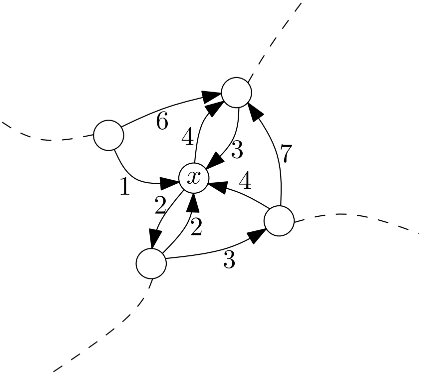
- Multi-level Dijkstra variant
- Ignore edges towards less important nodes



analogous: hierarchies with several levels of nodes of varying importances

Contraction Hierarchies [Geisberger et al.'12]

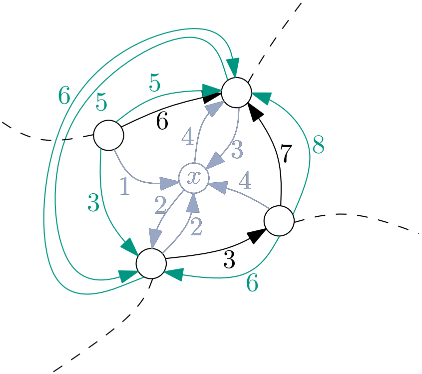
Idea: Compute shortcuts by iteratively contracting nodes



Contraction of *x*:
Remove *x*, add shortcuts among neighbors to maintain distances

Contraction Hierarchies [Geisberger et al.'12]

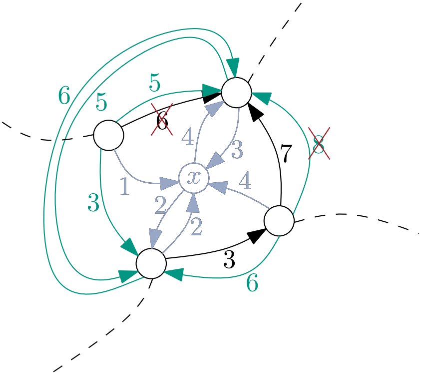
Idea: Compute shortcuts by iteratively contracting nodes



Contraction of *x*:
Remove *x*, add shortcuts among neighbors to maintain distances

Contraction Hierarchies [Geisberger et al.'12]

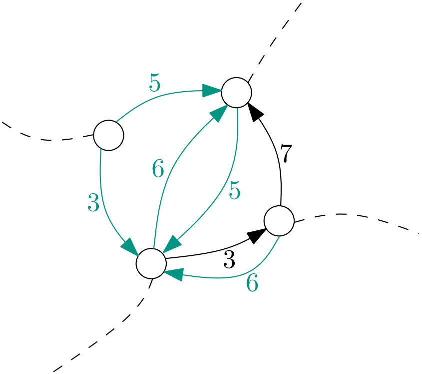
Idea: Compute shortcuts by iteratively contracting nodes



Delete longer edge in case of multi-edges

Contraction Hierarchies [Geisberger et al.'12]

Idea: Compute shortcuts by iteratively contracting nodes

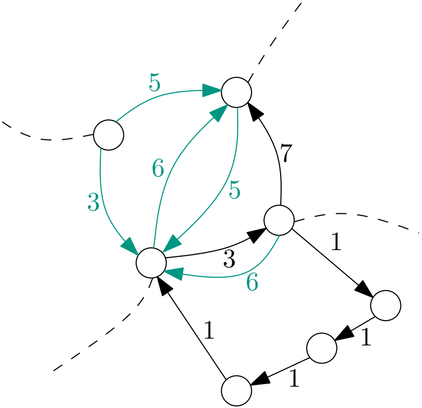


Resulting shortcuts

Institute for Theoretical Informatics
Chair Algorithmics

Contraction Hierarchies [Geisberger et al.'12]

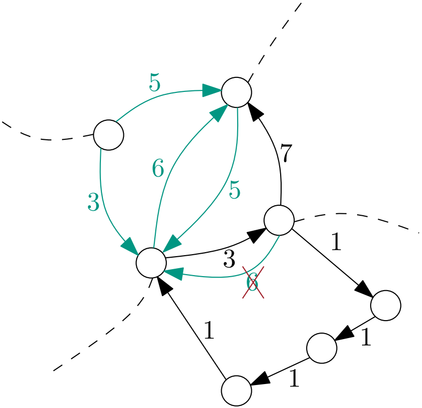
Idea: Compute shortcuts by iteratively contracting nodes



If shorter path through remaining graph exists, remove shortcut

Contraction Hierarchies [Geisberger et al.'12]

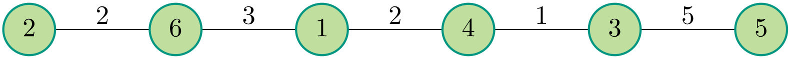
Idea: Compute shortcuts by iteratively contracting nodes



If shorter path through remaining graph exists, remove shortcut
Search for such shorter paths is called witness search

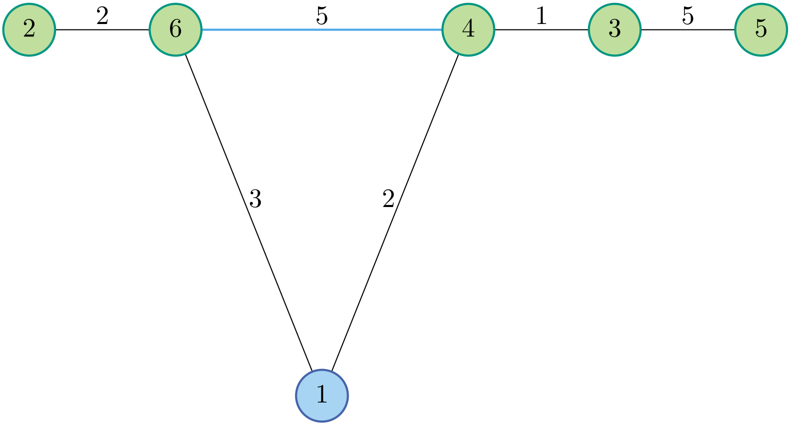
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



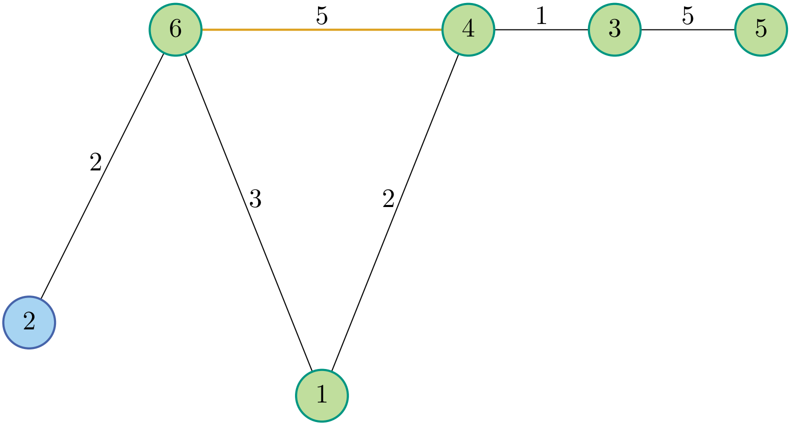
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



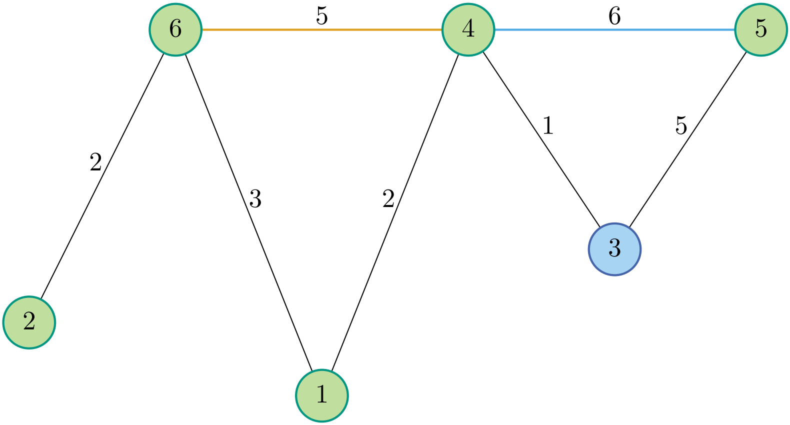
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



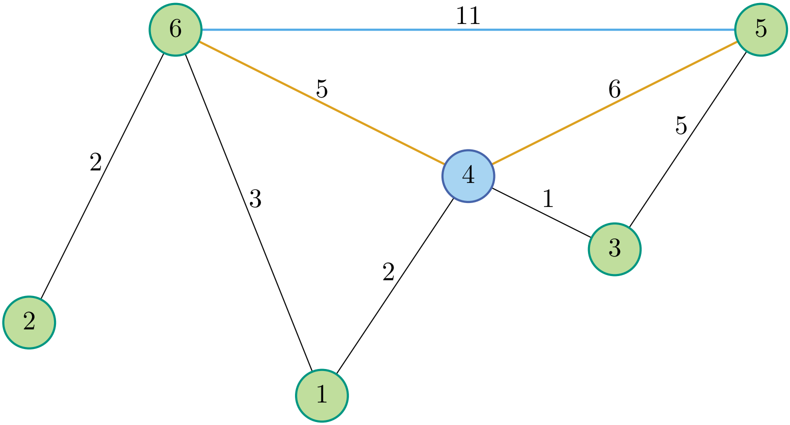
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



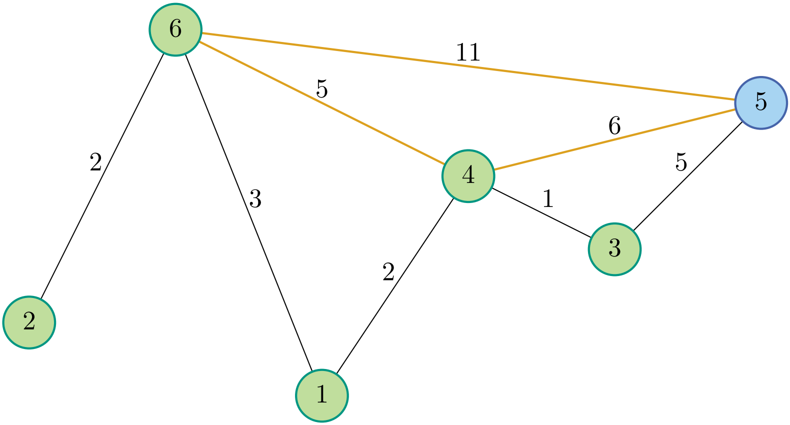
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



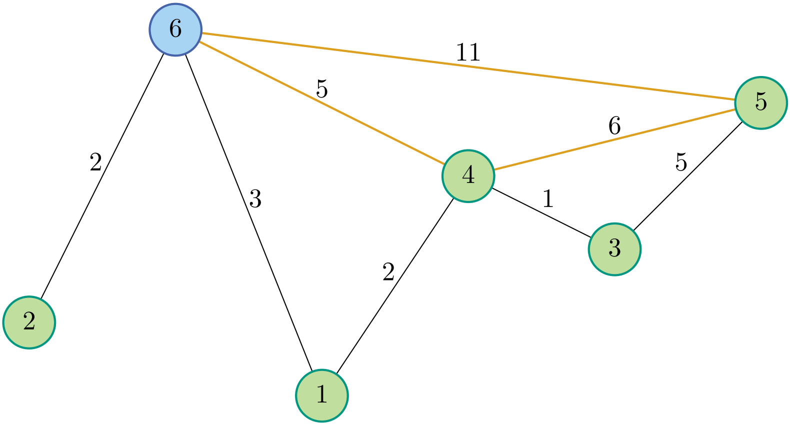
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



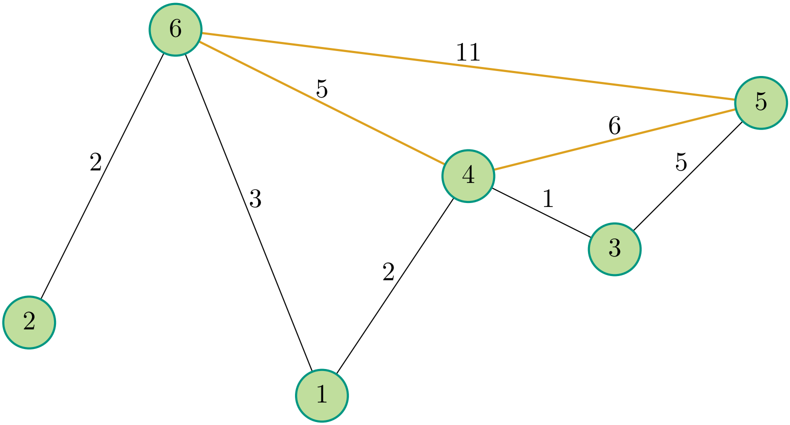
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



Contraction Hierarchies [Geisberger et al.'12]

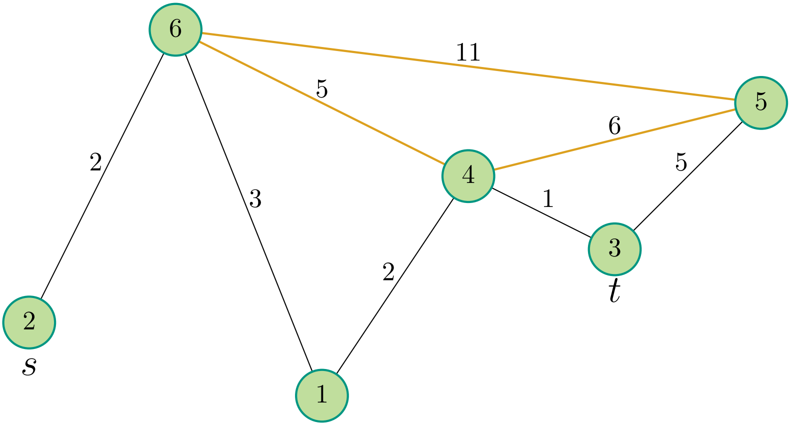
Preprocessing example: Iteratively contract nodes



Institute for Theoretical Informatics
Chair Algorithmics

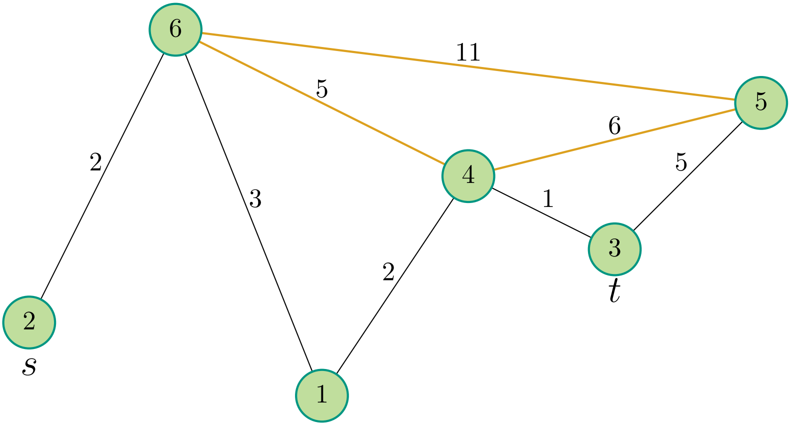
Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes



Contraction Hierarchies [Geisberger et al.'12]

Preprocessing example: Iteratively contract nodes

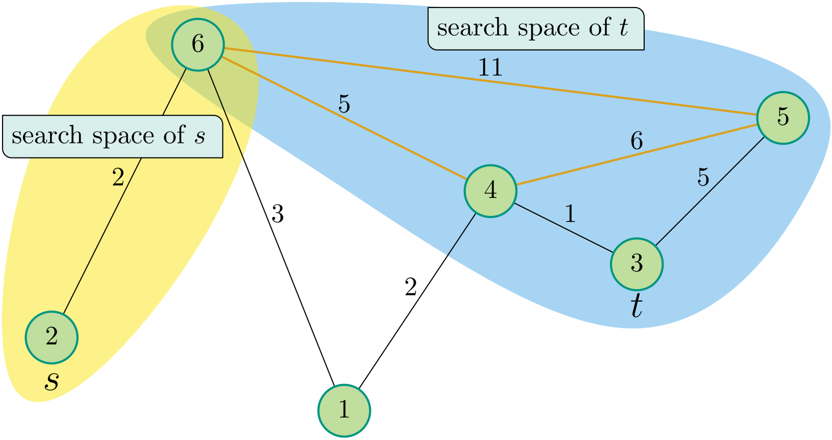


Order nodes by "importance"

Intuition: Nodes on more shortest paths are more important

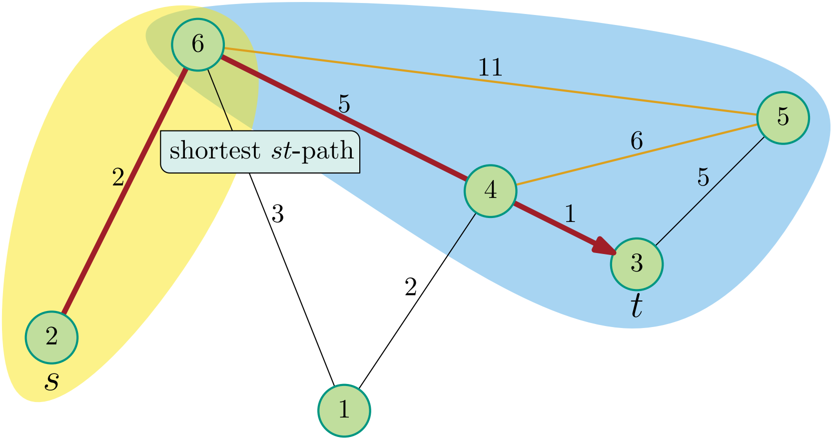
Contraction Hierarchies [Geisberger et al.'12]

Query example: Bidirectional upward search



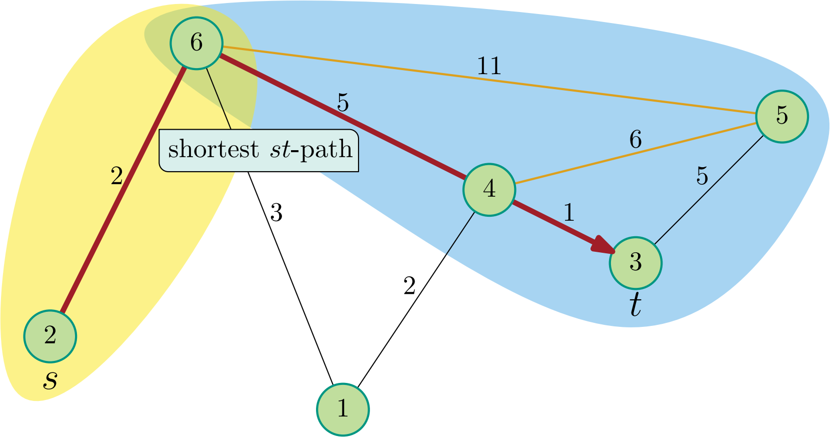
Contraction Hierarchies [Geisberger et al.'12]

Query example: Bidirectional upward search



Contraction Hierarchies [Geisberger et al.'12]

Query example: Bidirectional upward search



For every original shortest path, there is a shortest up-down path

New Challenges

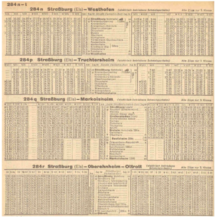
Energy Consumption of Electric Vehicles:

- Restricted battery capacity
- “Range anxiety”

User-Customizable Metrics

Timetable Information:

- Shortest paths in a timetable graph
- Timetable graphs differ from road graphs
- Incorporate unrestricted walking


284n Straßburg (Els)—Westhofen

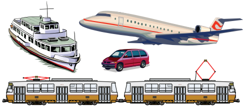
284p Straßburg (Els)—Truchtersheim

284q Straßburg (Els)—Markolsheim

284r Straßburg (Els)—Oberehnheim—Ottrott

Multimodal Route Planning:

- Change the type of transportation during the journey
- Constrained vs multicriteria shortest paths



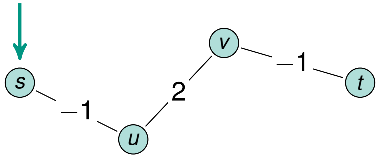
Route Planning for Electric Vehicles

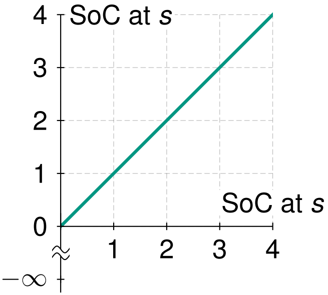
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



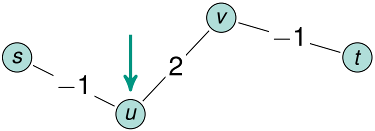


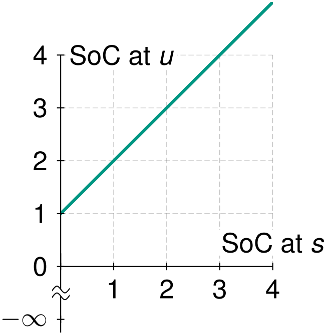
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



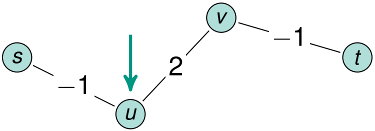


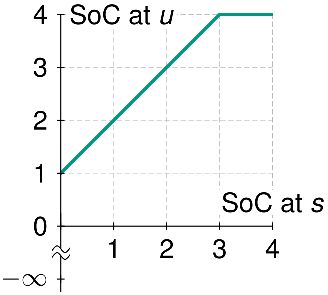
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4





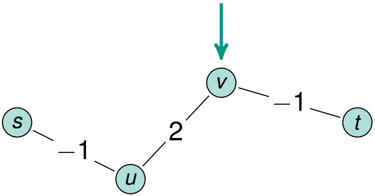
Institute for Theoretical Informatics
Chair Algorithmics

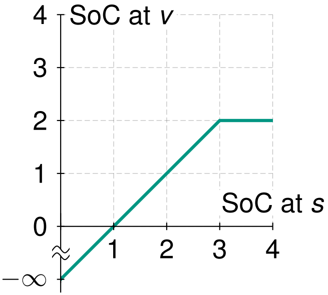
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



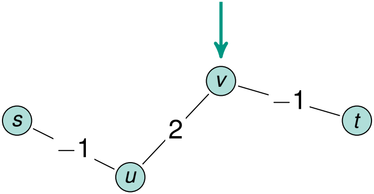


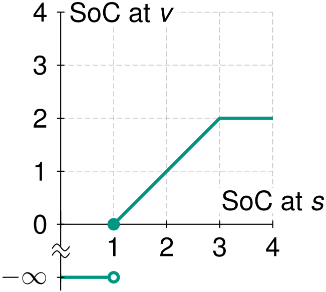
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



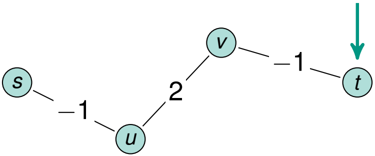


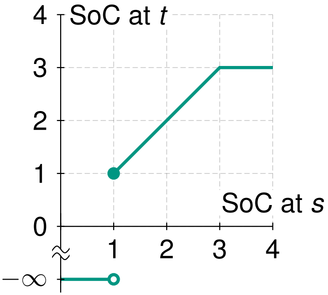
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



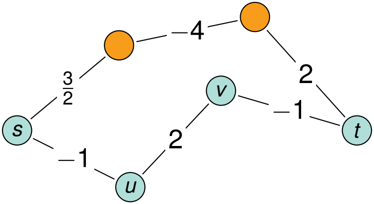


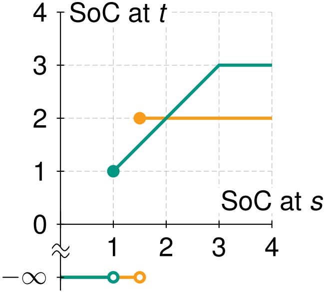
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4





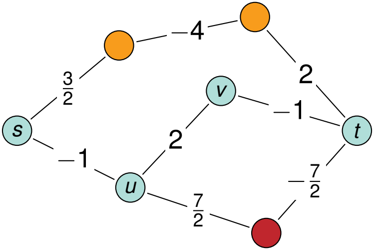
Institute for Theoretical Informatics
Chair Algorithmics

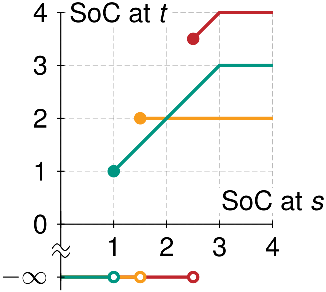
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC ("state of charge") at source to SoC at target

Example:
min. SoC 0, max. SoC 4



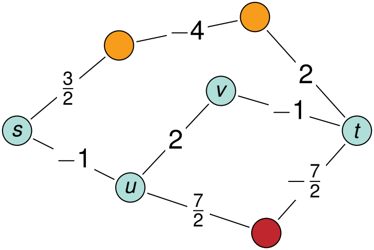


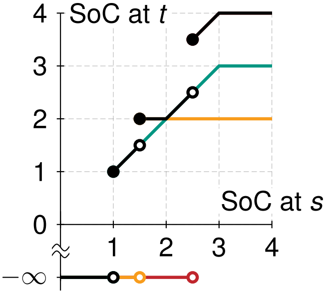
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4



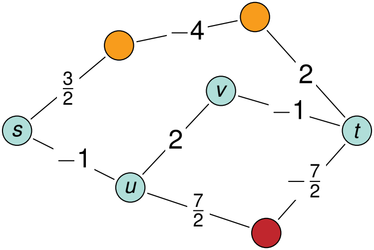


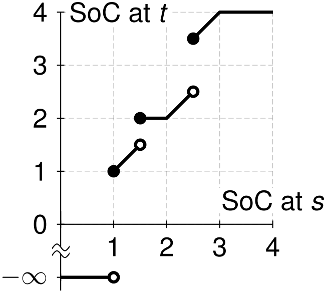
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC (“state of charge”) at source to SoC at target

Example:
min. SoC 0, max. SoC 4





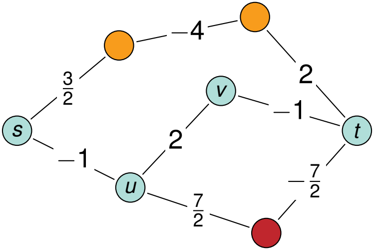
Institute for Theoretical Informatics
Chair Algorithmics

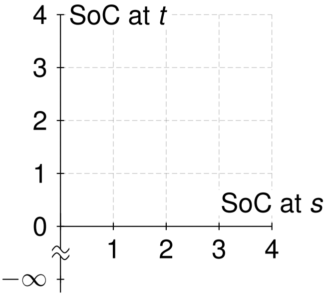
Route Planning for Electric Vehicles

- **Recuperation:** Negative edge costs (no negativen cycles)
- **Battery constraints:** Battery has a limited capacity

SoC function maps SoC ("state of charge") at source to SoC at target

Example:
min. SoC 0, max. SoC 4

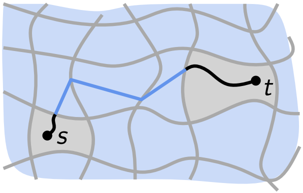


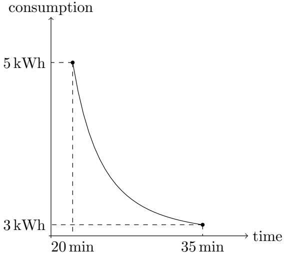


- Speedup techniques have to evaluate functions [Eisner et al.'11]

Energy-Optimal Routes [Baum et al.'13]

- Shortcuts are functions, not scalar values
- Bidirectional search more complicated (unknown state-of-charge at target)
- User-dependent consumption profiles (⇒ custom metrics)





Experiments:

- Fast queries (few milliseconds)
- Fast customization (few seconds)

But: Energy-optimal routes follow slow roads

- Energy-optimal paths: 63 % extra time
- Fastest paths: 62 % extra energy

⇒ Consider tradeoff between speed and energy consumption

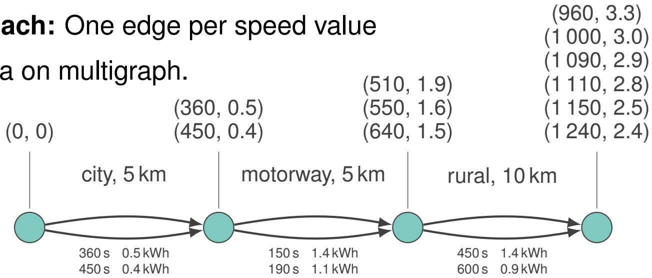
Find the fastest path such that the battery does not run out: $\mathcal{NP}$**-hard**

Institute for Theoretical Informatics
Chair Algorithmics

Constrained Shortest Paths

- Energy can be saved driving below speed limit
- Additional instructions to the driver
- **Simple approach:** One edge per speed value

$\Rightarrow$ Bicriteria Dijkstra on multigraph.



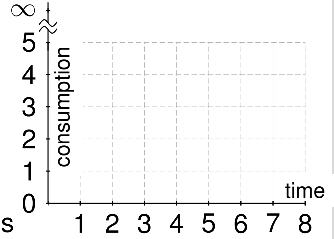
Worst case: n vertices with k parallel edges produce $\Theta(k^n)$ solutions

Simple implementation, but impractical running times

Realistic Model [Baum et al.'17]

Idea: Use continuous tradeoff functions instead of samples

- Times limits $\underline{x}$, $\overline{x}$ (speed limit, traffic flow, ...)
- More accurate model
- Less complex solution space



TFP: Tradeoff Function Propagating Algorithm

- Extends Bicriteria Dijkstra to tradeoff functions

CHAsp = CH & A* & TFP:

- Combines TFP with speedup techniques

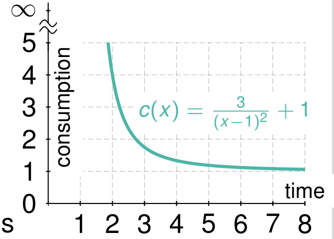
Experiments:

- Moderate preprocessing effort (Europe ∼3 h; Germany ∼30 min)
- Fast exact queries for typical ranges (<1 sec)
- Even faster heuristics (<100 ms, average error <1%)

Realistic Model [Baum et al.'17]

Idea: Use continuous tradeoff functions instead of samples

- Times limits $\underline{x}$, $\overline{x}$ (speed limit, traffic flow, ...)
- More accurate model
- Less complex solution space



TFP: Tradeoff Function Propagating Algorithm

- Extends Bicriteria Dijkstra to tradeoff functions

CHAsp = CH & A* & TFP:

- Combines TFP with speedup techniques

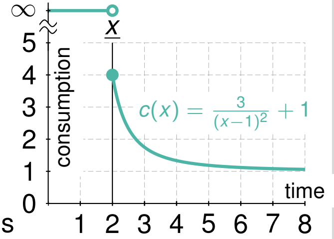
Experiments:

- Moderate preprocessing effort (Europe ∼3 h; Germany ∼30 min)
- Fast exact queries for typical ranges (<1 sec)
- Even faster heuristics (<100 ms, average error <1%)

Realistic Model [Baum et al.'17]

Idea: Use continuous tradeoff functions instead of samples

- Times limits $\underline{x}$, $\overline{x}$ (speed limit, traffic flow, ...)
- More accurate model
- Less complex solution space



TFP: Tradeoff Function Propagating Algorithm

- Extends Bicriteria Dijkstra to tradeoff functions

CHAsp = CH & A* & TFP:

- Combines TFP with speedup techniques

Experiments:

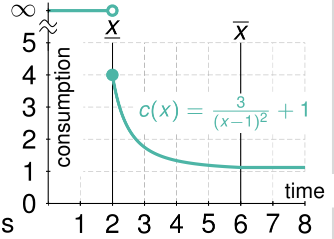
- Moderate preprocessing effort (Europe ∼3 h; Germany ∼30 min)
- Fast exact queries for typical ranges (<1 sec)
- Even faster heuristics (<100 ms, average error <1%)

Institute for Theoretical Informatics
Chair Algorithmics

Realistic Model [Baum et al.'17]

Idea: Use continuous tradeoff functions instead of samples

- Times limits $\underline{x}$, $\overline{x}$ (speed limit, traffic flow, ...)
- More accurate model
- Less complex solution space



TFP: Tradeoff Function Propagating Algorithm

- Extends Bicriteria Dijkstra to tradeoff functions

CHAsp = CH & A* & TFP:

- Combines TFP with speedup techniques

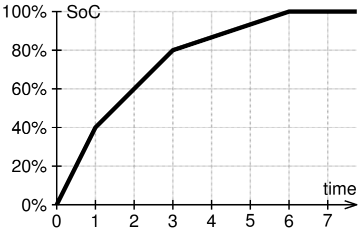
Experiments:

- Moderate preprocessing effort (Europe ~3 h; Germany ~30 min)
- Fast exact queries for typical ranges (<1 sec)
- Even faster heuristics (<100 ms, average error <1%)

Institute for Theoretical Informatics
Chair Algorithmics

Including Charging Stops [Baum et al.'15]

- Recharging allowed at some nodes (but requires charging time).
- Realistic models of charging stations:
  - Charging power varies
  - Super chargers
  - Battery swapping stations



Challenges:

1. Recuperation, battery constraints
2. Energy efficient driving vs. time consuming charging stops
   - Detour for reaching a charging station
3. Charging is not uniform
   - Interrupt charging and take another station later

Observations

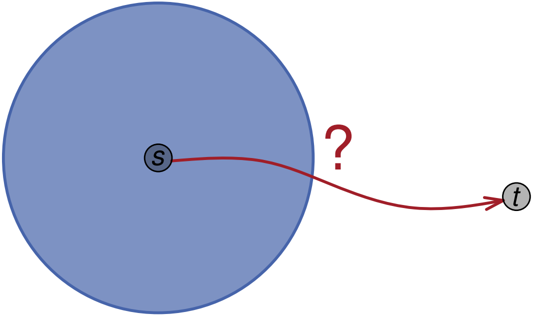
Find the fastest route from *s* to *t*:

Reachable area

Charging station

Observations

Find the fastest route from *s* to *t*:

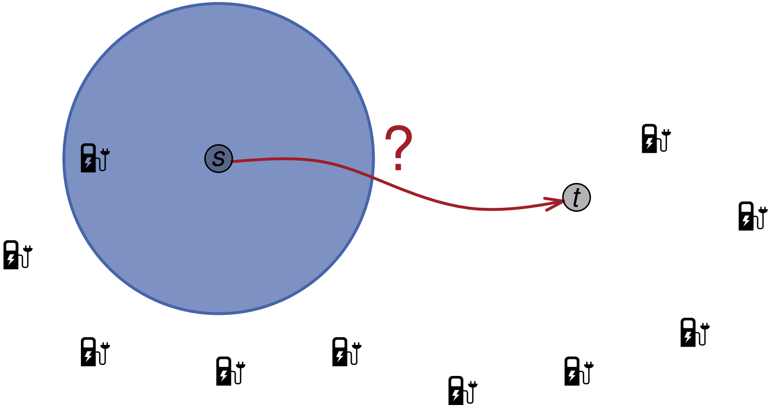


Reachable area

Charging station

Observations

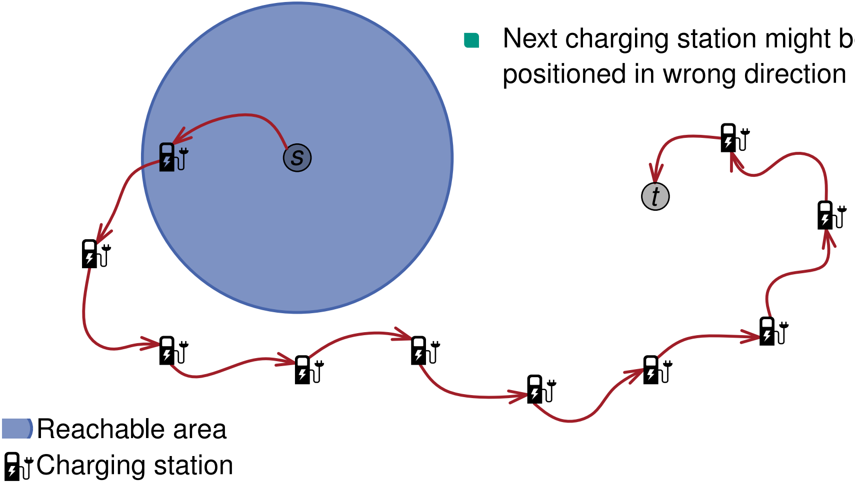
Find the fastest route from s to t:



Reachable area
Charging station

Observations

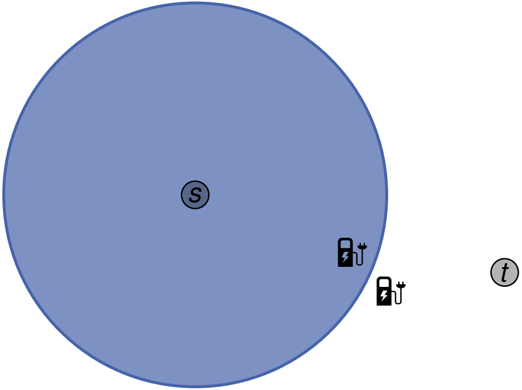
Find the fastest route from *s* to *t*:



- Next charging station might be positioned in wrong direction

Observations

Find the fastest route from *s* to *t*:



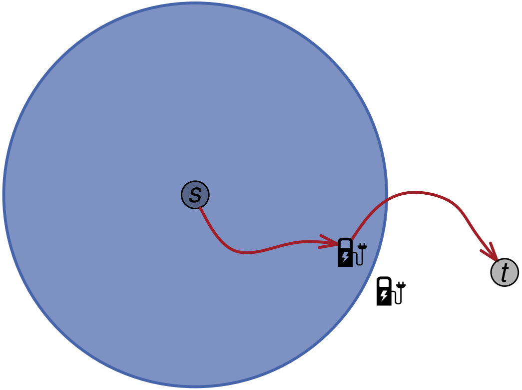
Reachable area

Charging station

Institute for Theoretical Informatics
Chair Algorithmics

Observations

Find the fastest route from *s* to *t*:

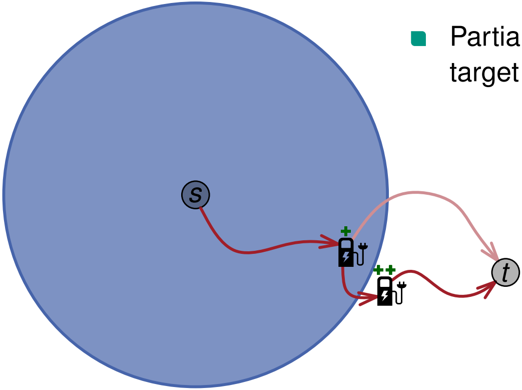


Reachable area

Charging station

Observations

Find the fastest route from *s* to *t*:

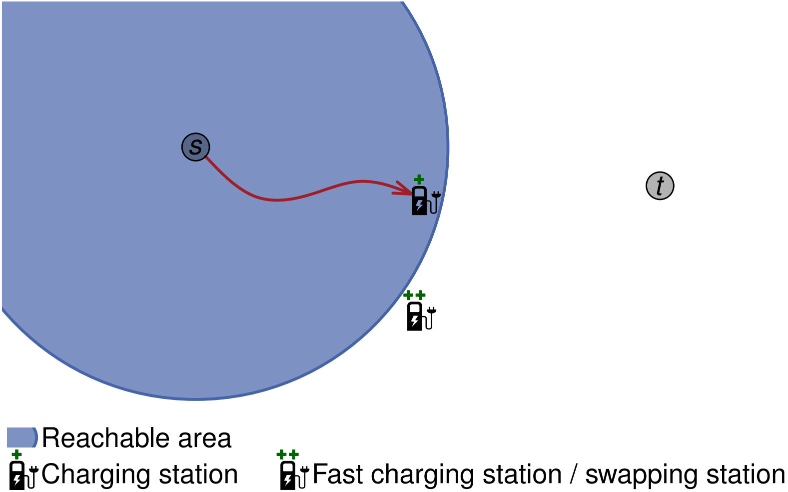


- Partial recharging, even if the target is already reachable

Reachable area
Charging station
Fast charging station / swapping station

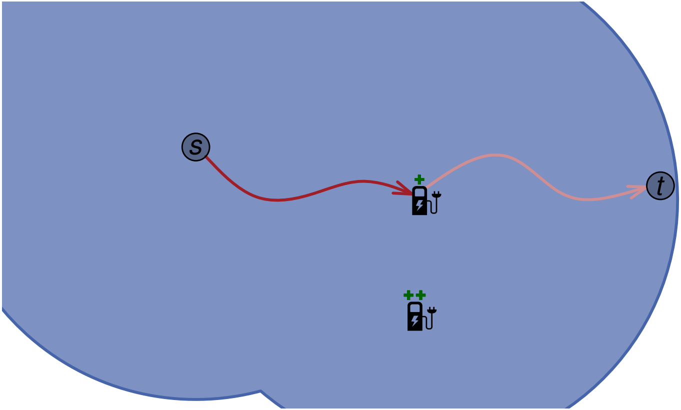
Observations

Find the fastest route from *s* to *t*:



Observations

Find the fastest route from s to t:



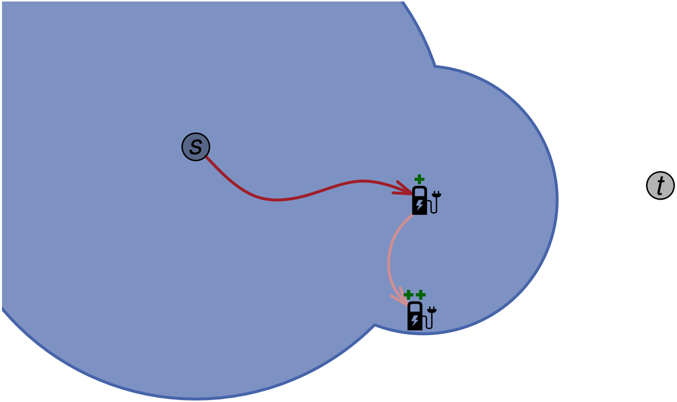
Reachable area

Charging station

Fast charging station / swapping station

Observations

Find the fastest route from *s* to *t*:



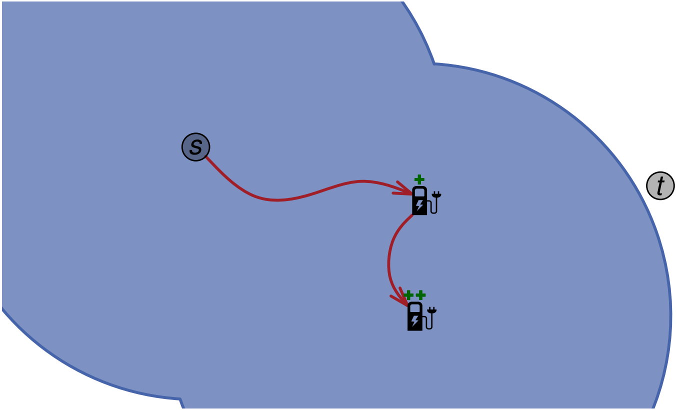
Reachable area

Charging station

Fast charging station / swapping station

Observations

Find the fastest route from s to t:



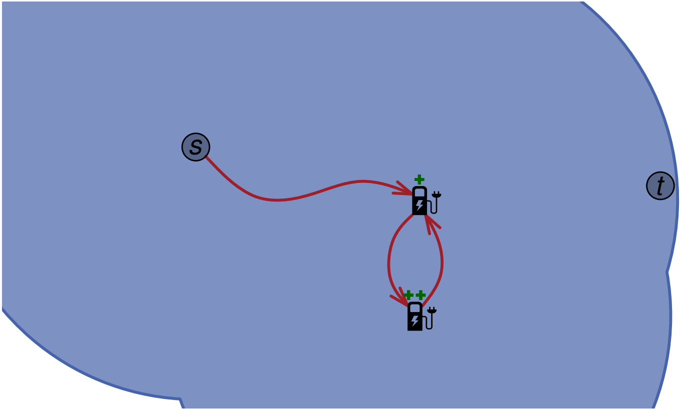
Reachable area

Charging station

Fast charging station / swapping station

Observations

Find the fastest route from s to t:



Reachable area

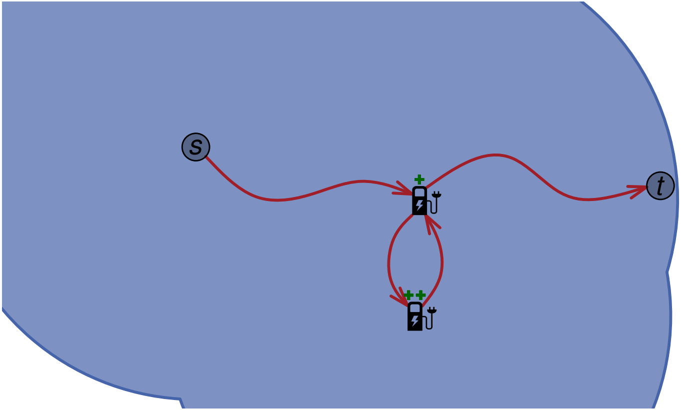
Charging station

Fast charging station / swapping station

Observations

Find the fastest route from *s* to *t*:

- Fastest route may contain cycles



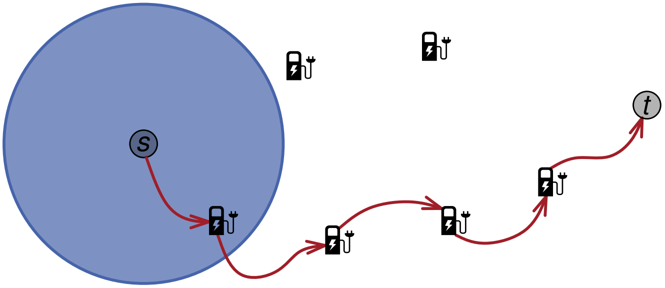
Reachable area

Charging station

Fast charging station / swapping station

Observations

Find the fastest route from s to t:



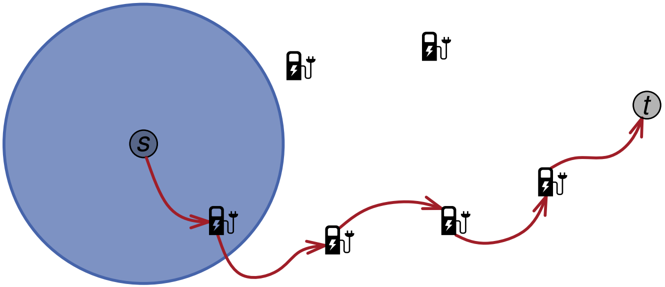
Reachable area

Charging station

Observations

Find the fastest route from *s* to *t*:

- Larger battery $\Rightarrow$ simpler problem ?



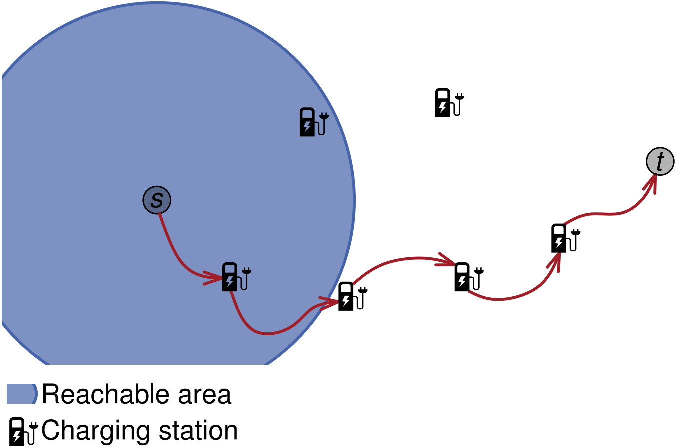
Reachable area

Charging station

Observations

Find the fastest route from s to t:

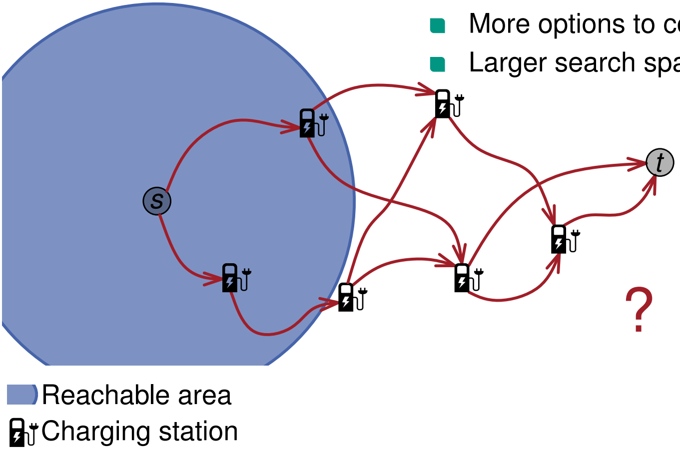
- Larger battery $\Rightarrow$ simpler problem ?



Observations

Find the fastest route from s to t:

- Larger battery $\Rightarrow$ simpler problem ?
- More options to consider
- Larger search space



Charging Function Propagation

CFP Algorithm

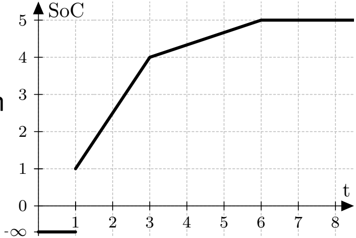
- Based on bicriteria Dijkstra
- If no charging station has been used: label = tuple (travel time, SoC)
- Per vertex: Maintain set of Pareto-optimal labels

Problem: When reaching a charging station: How long to stay?

- Depends on the remaining path to target
- Optimal state-of-charge for departure yet unknown

Solution:

- Delay this decision!
- Keep track of last passed charging station
- Labels represent charging tradeoffs



CHArge

CHArge = CH & A* & CFP:

- Combines CFP with speedup techniques
- Can handle arbitrary charging station types

Experiments:

- Moderate preprocessing times
  Europe $\sim$30 min; Germany $\sim$5 min
- Fast queries on continental-sized networks
  Europe $\sim$1 min; Germany $\sim$1 sec
- Even better results possible, using heuristics
  Europe $\sim$0.1–1 sec; Germany $\sim$20–100 ms
  often optimal solutions, mean error $\sim$1%

Range Visualization

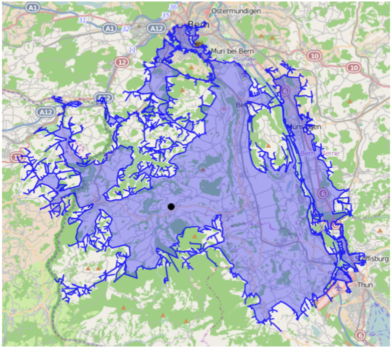
Visualize area reachable by an EV

Goals:

- Exact visualization
- Polygons with few segments
- Fast Computation

Subproblems:

1. Compute reachable subgraph [Baum et al.'15]
2. Compute polygon for visualization [Baum et al.'16]



Experiments: Polygons with $\sim$1 000 segments in $<$100 ms

Range Visualization

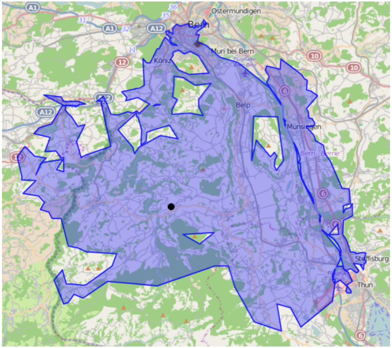
Visualize area reachable by an EV

Goals:

- Exact visualization
- Polygons with few segments
- Fast Computation

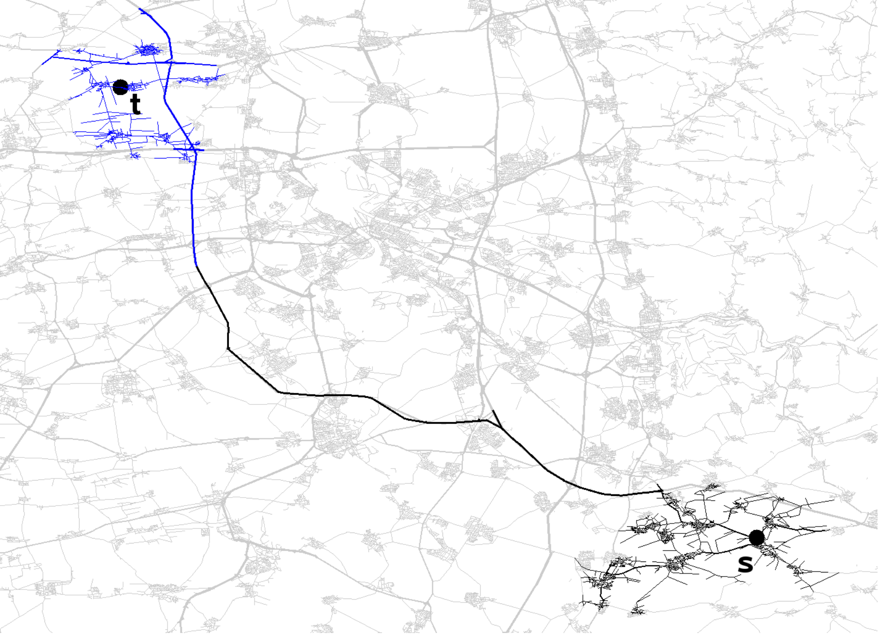
Subproblems:

1. Compute reachable subgraph [Baum et al.'15]
2. Compute polygon for visualization [Baum et al.'16]

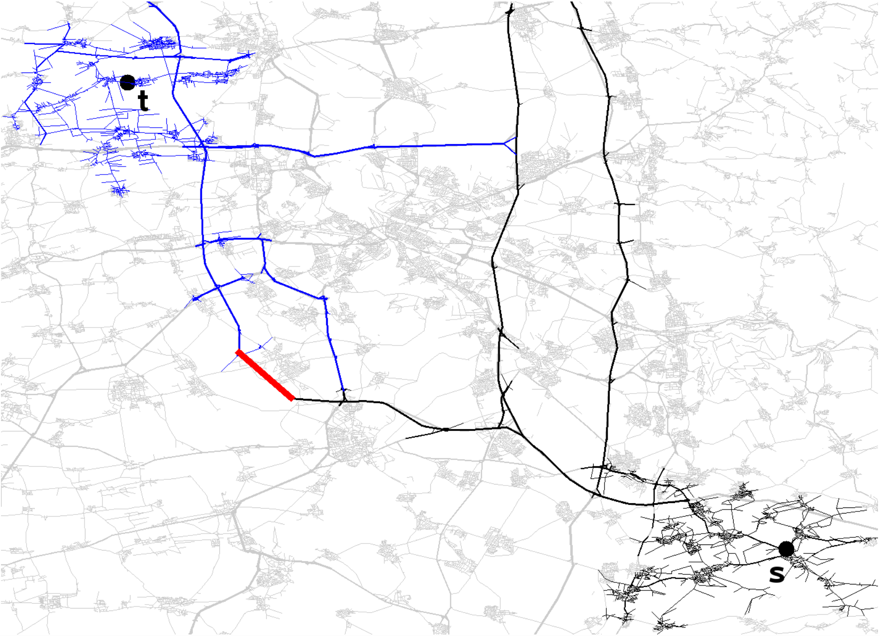


Experiments: Polygons with ∼1 000 segments in <100 ms

Customizable Route Planning



Customizable Route Planning



Real-World Metrics

- Distance
- Pedestrian
- Travel time, but don’t use toll roads
- Travel time, avoid left turns, height restrictions, . . .
- Traffic Congestion, accidents, . . .

Problem

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

Naive solution

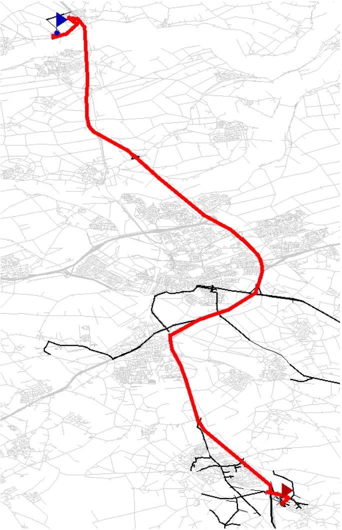
- Compute preprocessing for each metric
- Preprocessing and query time increase significantly
- Higher space overhead

$\Rightarrow$ **Metric customization**

Shortest Path Computation

Two-phase:

- Preprocessing (slow): compute additional data
- Query (fast): answer *st*-queries using data from preprocessing



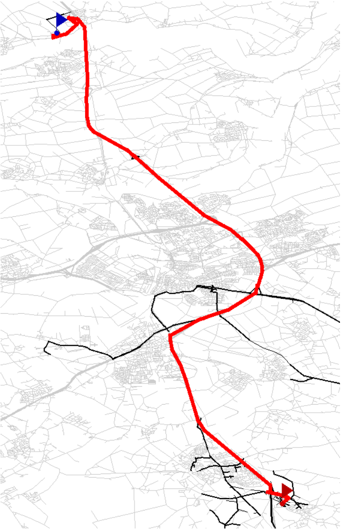
Shortest Path Computation

Two-phase:

- Preprocessing (slow): compute additional data
- Query (fast): answer *st*-queries using data from preprocessing

Three-phase:

- Preprocessing (slow): compute additional weight-independent data
- Customization (reasonably fast): introduce weights
- Query (fast): answer *st*-queries using data from preprocessing and customization



CH Revisited

Metric-dependent orders:

- Node order determines CH performance
- Many ordering algorithms exist
- Some fast, some slow, some specific to certain graph classes, ...
- **But**: Best order depends on the weights

Institute for Theoretical Informatics
Chair Algorithmics

CH Revisited

Metric-dependent orders:

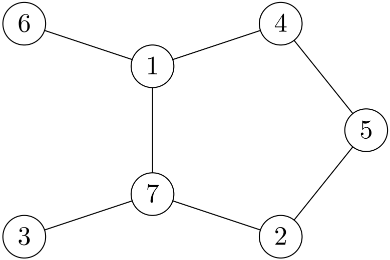
- Node order determines CH performance
- Many ordering algorithms exist
- Some fast, some slow, some specific to certain graph classes, ...
- **But**: Best order depends on the weights

Metric-independent orders:

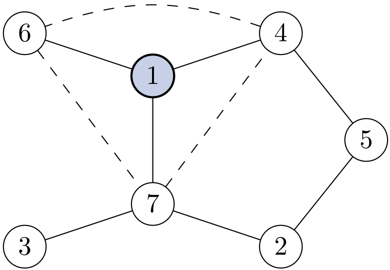
- Is there an order that is good for every weight?
  (but not necessarily best)
- Core idea of 3-phase CH

Institute for Theoretical Informatics
Chair Algorithmics

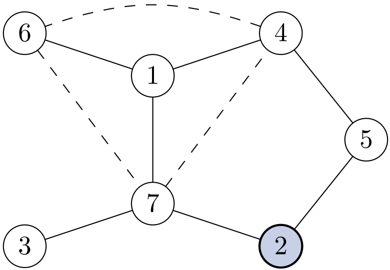
Graph Fill-In



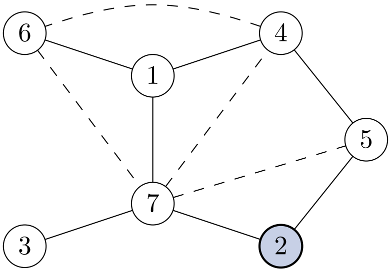
Graph Fill-In



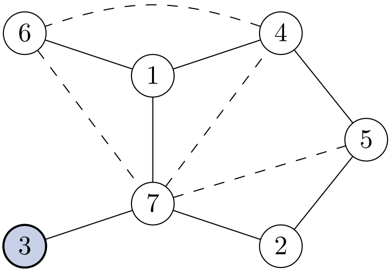
Graph Fill-In



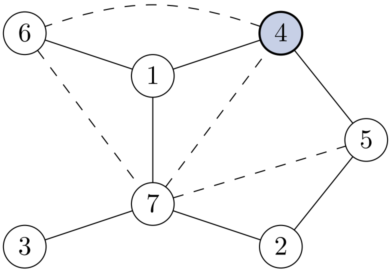
Graph Fill-In



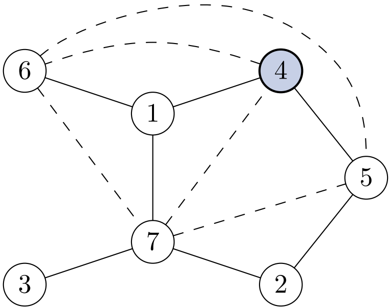
Graph Fill-In



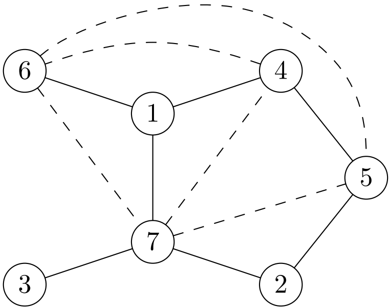
Graph Fill-In



Graph Fill-In

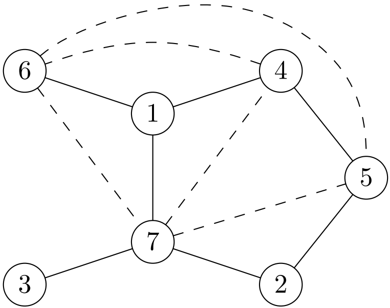


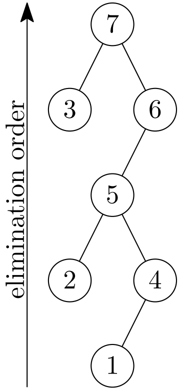
Graph Fill-In



Institute for Theoretical Informatics
Chair Algorithmics

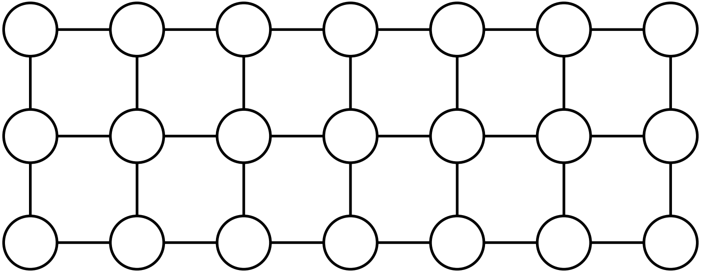
Graph Fill-In



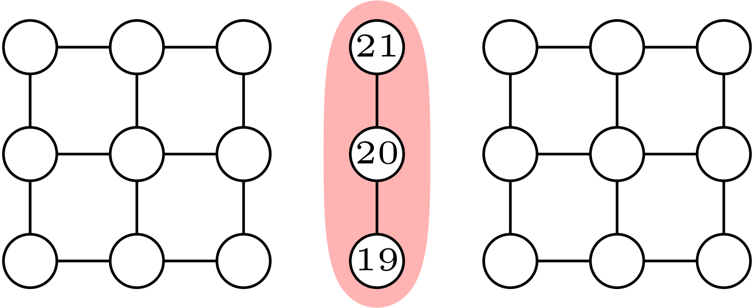


elimination tree

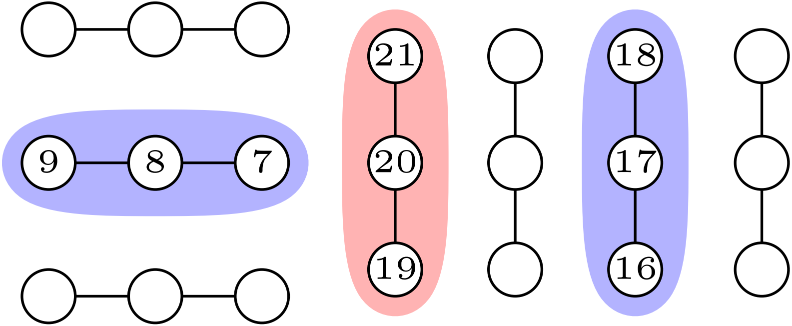
Nested Dissection (ND)



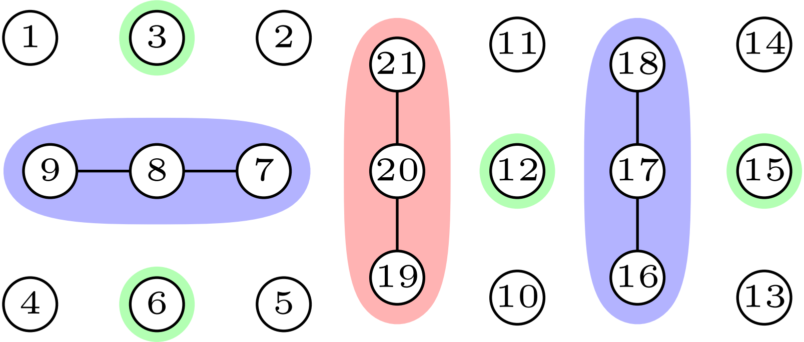
Nested Dissection (ND)



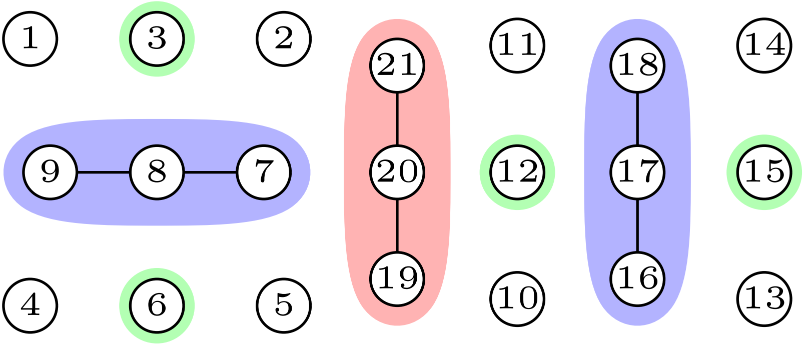
Nested Dissection (ND)



Nested Dissection (ND)



Nested Dissection (ND)



ND ordering from recursive $O(n^\beta)$ balanced separators
yields elimination tree of height $O(n^\beta)$

Consequences [Bauer et al.'13, Dibbelt et al.'16]

Theoretical guarantee:

- ND-ordering yields search space guarantee of $O(n^{\beta})$ nodes
- $O(\sqrt{n})$ rec. balanced separators yield guarantee of $O(n)$ edges
- Planar graphs have $O(\sqrt{n})$ recursive balanced separators

Institute for Theoretical Informatics
Chair Algorithmics

Consequences [Bauer et al.'13, Dibbelt et al.'16]

Theoretical guarantee:

- ND-ordering yields search space guarantee of $O(n^{\beta})$ nodes
- $O(\sqrt{n})$ rec. balanced separators yield guarantee of $O(n)$ edges
- Planar graphs have $O(\sqrt{n})$ recursive balanced separators

Practical impact:

- Contraction ordering that is weight-independent
- Minimum vs maximum contraction hierarchies
- Customizable contraction hierarchies (CCH)

Institute for Theoretical Informatics
Chair Algorithmics

CCH : Three-Phase Approach

- **Preprocessing**
  - Compute ND-order
  - Solve balanced graph bisection subproblem
  - Compute fill-in (shortcuts)
- **Customization**
  - Add weights to shortcuts
    - Enumerate lower triangles in CH
- **Query**
  - Existing CH-query works unmodified
  - **Alternative**: Elimination-tree query

Elimination-Tree-Query

While not at the root do:

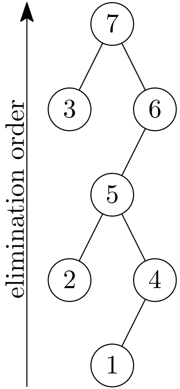
- If s comes before t in the order:
  - Relax outgoing arcs of s in its search space
  - $s \leftarrow \mathrm{parent}(s)$
- Else:
  - Relax outgoing arcs of t in its search space
  - $t \leftarrow \mathrm{parent}(t)$

Advantage:

- No queue
- Works with negative weights

But:

- Local queries are not faster than long distance queries



Experimental Evaluation

Instance:

- Standard DIMACS Europe benchmark, travel time metric
- $\approx$ 18M nodes, $\approx$42M directed edges
- 26.5% degree 1, 18.7% degree 2

Institute for Theoretical Informatics
Chair Algorithmics

Experimental Evaluation

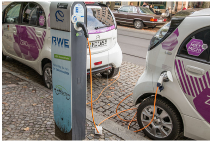
Instance:

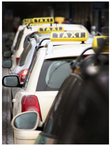
- Standard DIMACS Europe benchmark, travel time metric
- $\approx$ 18M nodes, $\approx$42M directed edges
- 26.5% degree 1, 18.7% degree 2

Results:

- Plain Dijkstra: $\approx$ 2s
- CH-preprocessing: $\approx$ 5min - 6h
- CH-query: $\approx$ 0.107ms
- CCH-customization (16 threads): $\approx$ 420ms
- CCH-query: $\approx$ 0.413ms
- CCH-query (+perfect witness search): $\approx$ 0.161ms
- CRP-customization (12 threads): $\approx$ 370ms
- CRP-query: $\approx$ 1.65ms

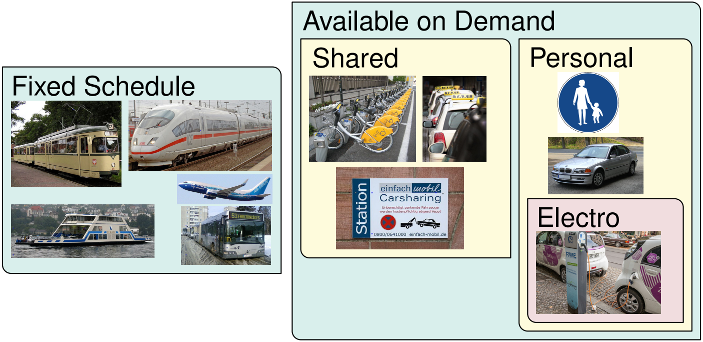
Multimodal Route Planning





- Many modes of transportation

Multimodal Route Planning



- Many modes of transportation
- Many different set of rules
- and many more modes and variations exist

Unrestricted Walking

Common Algorithms & Walking Restrictions:

Algorithm	Footpaths
RAPTOR [Delling et al. ’12/’14]	Transitively closed
CSA [Dibbelt et al. ’13/’14]	Transitively closed
Trip-Based Routing [Witt ’15]	Transitively closed
Transfer Patterns [Bast et al. ’10/’16]	Max. 400 meters
Frequency-Based [Bast, Storandt ’14]	Max. 15 minutes
Public Transit Labeling [Delling et al. ’15]	As specified by the timetable

Institute for Theoretical Informatics
Chair Algorithmics

Unrestricted Walking

Common Algorithms & Walking Restrictions:

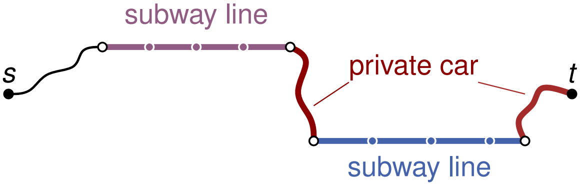
Algorithm	Footpaths
RAPTOR [Delling et al. '12/'14]	Transitively closed
CSA [Dibbelt et al. '13/'14]	Transitively closed
Trip-Based Routing [Witt '15]	Transitively closed
Transfer Patterns [Bast et al. '10/'16]	Max. 400 meters
Frequency-Based [Bast, Storandt '14]	Max. 15 minutes
Public Transit Labeling [Delling et al. '15]	As specified by the timetable

Problems:

- Transitively close graph $\Rightarrow$ limited walking
  (e.g. walking $\leq$ 15 min $\Rightarrow$ avg. degree $>$ 100)
- Unrestricted walking reduces travel times significantly [Wagner & Zündorf '17]
- Open problem: Efficient algorithms

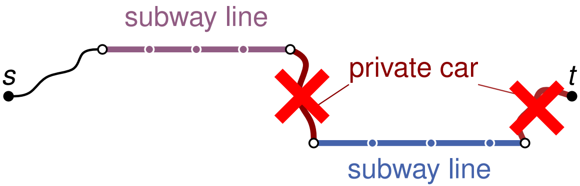
Multiple Transportation Modes

Problem: Unrestricted routes allow arbitrary transfers



Multiple Transportation Modes

Problem: Unrestricted routes allow arbitrary transfers

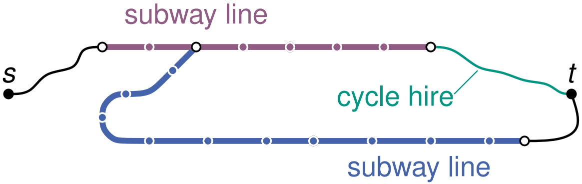


- Not all sequences of transportation modes are reasonable

Multiple Transportation Modes

[Delling et al.'09, Dibbelt et al.'12]

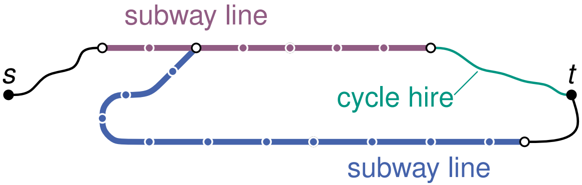
Problem: Unrestricted routes allow arbitrary transfers



- Not all sequences of transportation modes are reasonable
- Label constrained shortest paths
- Dijkstra's algorithm on product of network and finite-state automaton
- Adopt speed-up techniques

Multiple Transportation Modes

Shortcoming



Multiple Transportation Modes

Shortcoming

Multiple Transportation Modes

Shortcoming

s ? t

- Restrictions must be known in advance
- User might not know them
- Only one route is computed (no alternatives)

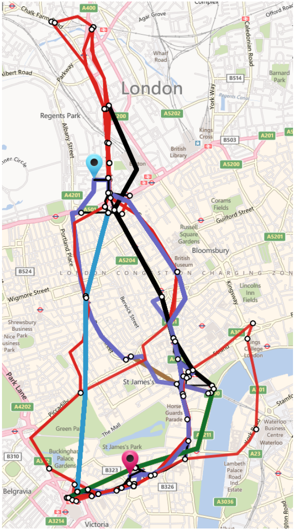
Goal: compute a *useful set* of multimodal journeys

Institute for Theoretical Informatics
Chair Algorithmics

Multiple Transportation Modes
[Delling et al.'13]

Karlsruhe Institute of Technology

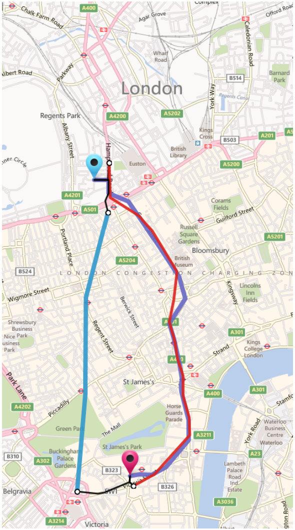
- Train, Bus, Tube, Taxi, Walking, Cycling
- Optimize w.r.t. multiple criteria:
  travel time, costs, emissions,
  # of mode changes, walking duration . . .
- Pareto solution set too large



Multiple Transportation Modes
[Delling et al.'13]

- Train, Bus, Tube, Taxi, Walking, Cycling
- Optimize w.r.t. multiple criteria:
  travel time, costs, emissions,
  # of mode changes, walking duration . . .
- Pareto solution set too large

⇒ Reduce to most relevant journeys

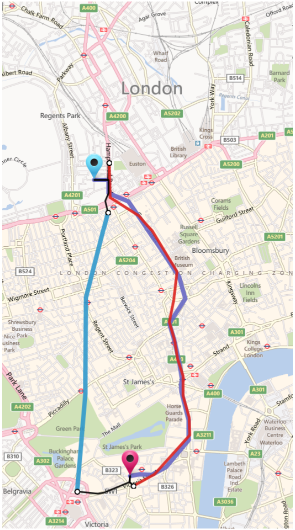


Multiple Transportation Modes

[Delling et al.'13]

- Train, Bus, Tube, Taxi, Walking, Cycling
- Optimize w.r.t. multiple criteria:
  travel time, costs, emissions,
  # of mode changes, walking duration . . .
- Pareto solution set too large

⇒ Reduce to most relevant journeys



Preliminary results

- Grade by relevance
- Fuzzy filter

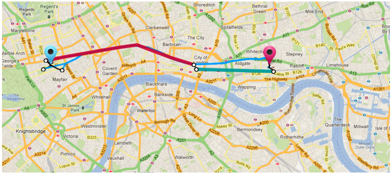
Conclusion & Outlook

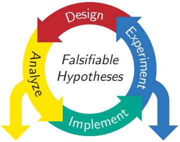
Success story for algorithm engineering

- Fast route planning on road and timetable networks
- Metric matters
- Multimodal route planning expensive

Many new challenges

- Scalability and quality in multimodal route planning
- Incorporating alternative mobility concepts
- Robustness, adjustable to unforeseen traffic situations
- Personalized route planning
- Eco-friendliness
- Autonomous driving
- Traffic control
- . . .





Thanks for your attention!