Karlsruhe Institute of Technology

Candidate Sets for Alternative Routes in Road Networks

Dennis Luxen, Dennis Schieferdecker – *{luxen,schieferdecker}@kit.edu*
`http://algo2.iti.kit.edu/AlgorithmenII.php`

Institute of Theoretical Informatics - Algorithmics II

Motivation

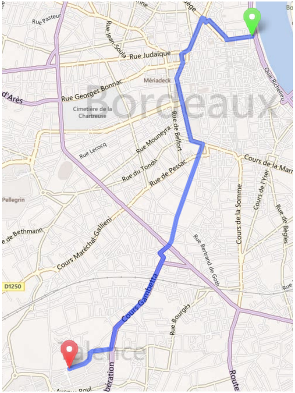
Advanced Route Planning

PAST: finding shortest paths

- "race" in algorithm engineering
  (new algorithms, engineering existing ones)
- sub-microsecond query times, preprocessing in minutes

PRESENT: more complex tasks

- time-dependent / stochastic routing
- dynamic / multi-modal settings
- alternative routes

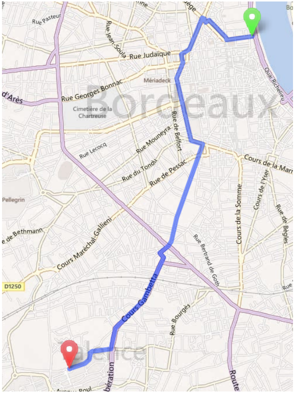


Motivation

Advanced Route Planning

PAST: finding shortest paths

- "race" in algorithm engineering
  (new algorithms, engineering existing ones)
- sub-microsecond query times, preprocessing in minutes



PRESENT: more complex tasks

- time-dependent / stochastic routing
- dynamic / multi-modal settings
- **alternative routes**

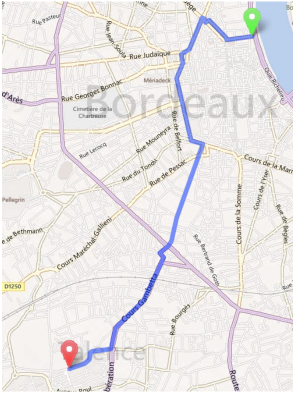
Motivation

Advanced Route Planning

Why even consider alternative routes?

- attractive from a business point of view
  - easy way to provide options
    (users have diverse preferences, let them decide)
  - overcome shortcomings in model and data
    (shortest path might not be best in reality!)

- interesting from a scientific viewpoint
  - building block (traffic models, stochastic routing)
  - NP-hard aspects (alternative graphs)

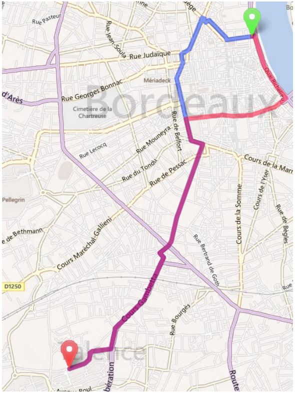


Motivation
Advanced Route Planning

Why even consider alternative routes?

- attractive from a business point of view
  - easy way to provide options
    (users have diverse preferences, let them decide)
  - overcome shortcomings in model and data
    (shortest path might not be best in reality!)

- interesting from a scientific viewpoint
  - building block (traffic models, stochastic routing)
  - NP-hard aspects (alternative graphs)

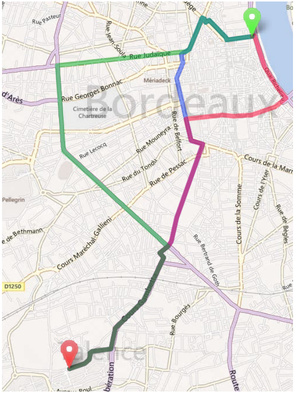


Motivation

Advanced Route Planning

Why even consider alternative routes?

- attractive from a business point of view
  - easy way to provide options
    (users have diverse preferences, let them decide)
  - overcome shortcomings in model and data
    (shortest path might not be best in reality!)
- interesting from a scientific viewpoint
  - building block (traffic models, stochastic routing)
  - NP-hard aspects (alternative graphs)



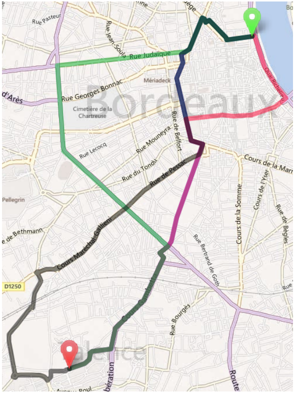
Motivation

Advanced Route Planning

Why even consider alternative routes?

- attractive from a business point of view
  - easy way to provide options
    (users have diverse preferences, let them decide)
  - overcome shortcomings in model and data
    (shortest path might not be best in reality!)

- interesting from a scientific viewpoint
  - building block (traffic models, stochastic routing)
  - NP-hard aspects (alternative graphs)



Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly

Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly

Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

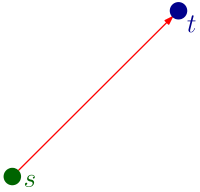
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

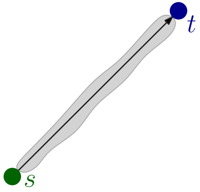
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

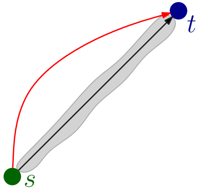
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

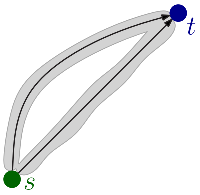
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

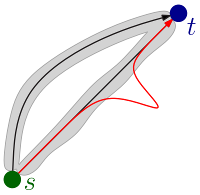
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

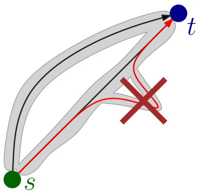
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

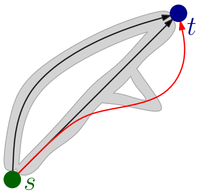
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

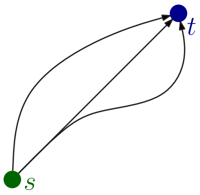
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

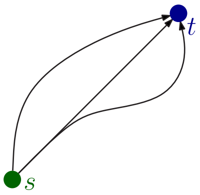
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters

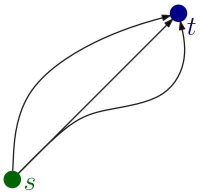
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

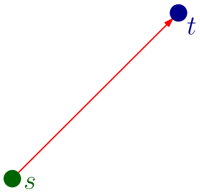
- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters



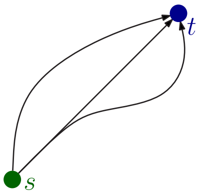
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

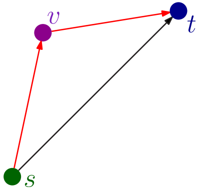
- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters



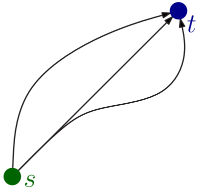
Alternative Routes

What has been done before?

Alternative Route Graphs in Road Networks

[Bader et al. 11]

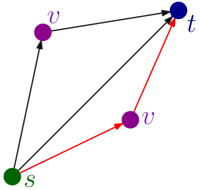
- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters



Alternative Routes

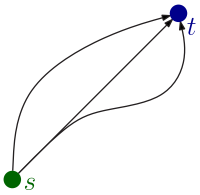
What has been done before?

Karlsruhe Institute of Technology

Alternative Route Graphs in Road Networks

[Bader et al. 11]

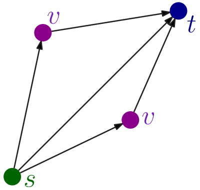
- penalty method
- constructing alternative graphs
  (open how to extract single alternatives)
- slow, difficult to tune properly



Alternative Routes in Road Networks

[Abraham et al. 10a]

- via nodes (Choice Routing)
- finding (single) good alternatives
- fast, intuitive parameters



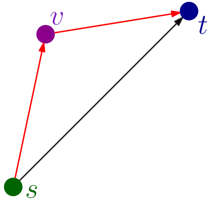
Modeling

[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \to \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$



Modeling

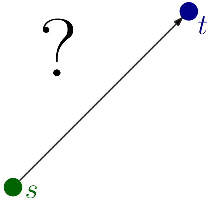
[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \rightarrow \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$

criteria for a good alternative ?



Modeling

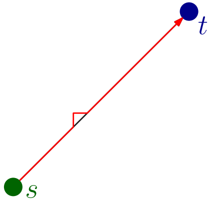
[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \to \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$

criteria for a good alternative ?



Modeling

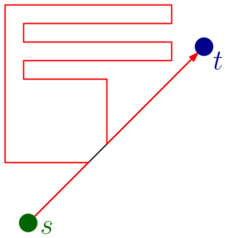
[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \to \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$

criteria for a good alternative ?



Modeling

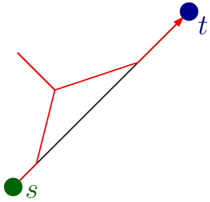
[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \rightarrow \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$

criteria for a good alternative ?



Modeling

[Abraham et al. 10a]

How do you define a good alternative?

basic model

- *road network*: graph $G(V, E)$, edge weights $w : E \rightarrow \mathbb{R}_0^+$
- *alternative*: concatenation of two shortest paths $\langle s..v \rangle$, $\langle v..t \rangle$

criteria for a good alternative

- not too much longer (stretch ϵ)
- sufficiently different (overlap γ)
- reasonable (α-locally optimal)

$\rightarrow$ define *quantitative quality measure* $f(\alpha, \gamma, \epsilon)$ for alternatives
(only accept alternatives as viable, if single criteria are good enough)

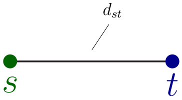
Alternatives Routes

[Abraham et al. 10a]

How do you find them?

basic approach (X-BDV)

- based on bidirectional search (Dijkstra's algorithm)
  → grow search spaces from *s* and *t*
- meeting nodes in search spaces are *candidate via nodes*
  → rank and check if alternative is viable



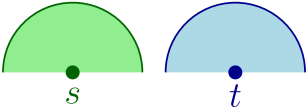
Alternatives Routes

[Abraham et al. 10a]

How do you find them?

basic approach (X-BDV)

- based on bidirectional search (Dijkstra's algorithm)
  $\longrightarrow$ grow search spaces from *s* and *t*
- meeting nodes in search spaces are *candidate via nodes*
  $\longrightarrow$ rank and check if alternative is viable



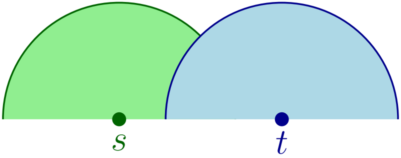
Alternatives Routes

How do you find them?

[Abraham et al. 10a]

basic approach (X-BDV)

- based on bidirectional search (Dijkstra's algorithm)
  $\longrightarrow$ grow search spaces from *s* and *t*
- meeting nodes in search spaces are *candidate via nodes*
  $\longrightarrow$ rank and check if alternative is viable



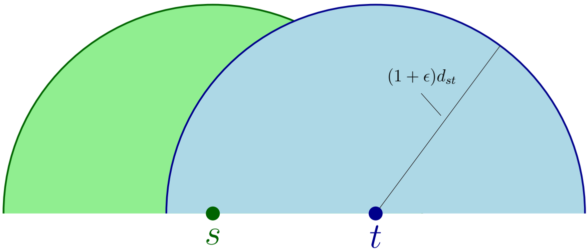
Alternatives Routes

[Abraham et al. 10a]

How do you find them?

basic approach (X-BDV)

- based on bidirectional search (Dijkstra's algorithm)
  $\longrightarrow$ grow search spaces from *s* and *t*
- meeting nodes in search spaces are *candidate via nodes*
  $\longrightarrow$ rank and check if alternative is viable



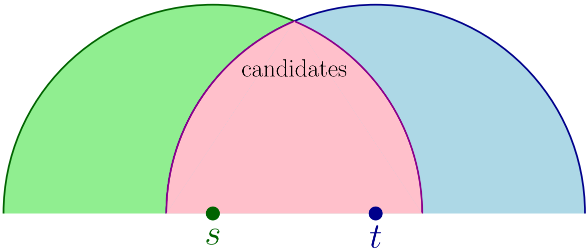
Alternatives Routes

How do you find them?

[Abraham et al. 10a]

basic approach (X-BDV)

- based on bidirectional search (Dijkstra's algorithm)
  $\longrightarrow$ grow search spaces from *s* and *t*
- meeting nodes in search spaces are *candidate via nodes*
  $\longrightarrow$ rank and check if alternative is viable



Alternatives Routes

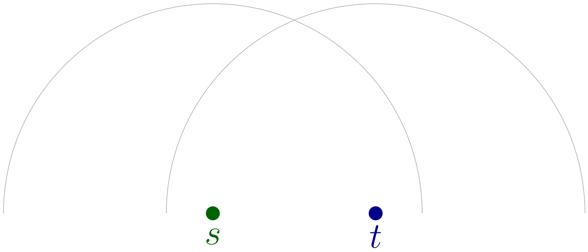
How do you find them, quicker?

[Abraham et al. 10a]

Karlsruhe Institute of Technology

using speed-up techniques (X-CHV)

- much faster than Dijkstra's algorithm (Contraction Hierarchies)
  $\longrightarrow$ reduces search spaces significantly
  $\longrightarrow$ less opportunities to find alternatives



Alternatives Routes

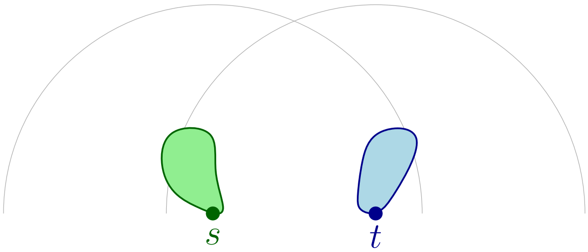
How do you find them, quicker?

[Abraham et al. 10a]

Karlsruhe Institute of Technology

using speed-up techniques (X-CHV)

- much faster than Dijkstra's algorithm (Contraction Hierarchies)
  $\longrightarrow$ reduces search spaces significantly
  $\longrightarrow$ less opportunities to find alternatives



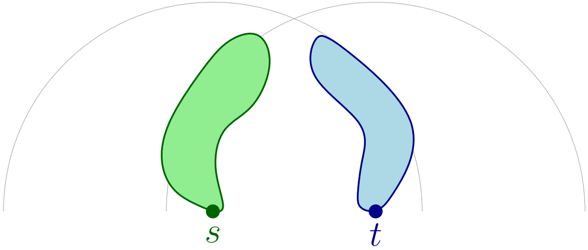
Alternatives Routes

How do you find them, quicker?

[Abraham et al. 10a]

using speed-up techniques (X-CHV)

- much faster than Dijkstra's algorithm (Contraction Hierarchies)
  $\longrightarrow$ reduces search spaces significantly
  $\longrightarrow$ less opportunities to find alternatives

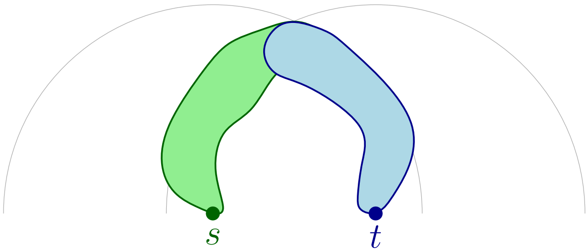


Alternatives Routes

How do you find them, quicker?

using speed-up techniques (X-CHV)

- much faster than Dijkstra's algorithm (Contraction Hierarchies)
  $\longrightarrow$ reduces search spaces significantly
  $\longrightarrow$ less opportunities to find alternatives



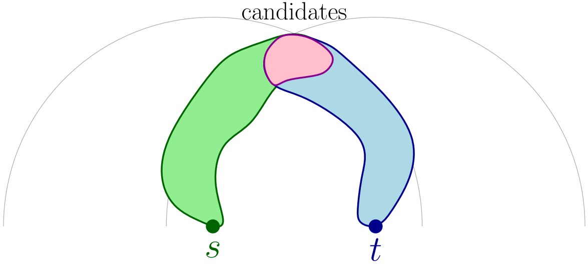
Alternatives Routes

How do you find them, quicker?

[Abraham et al. 10a]

using speed-up techniques (X-CHV)

- much faster than Dijkstra's algorithm (Contraction Hierarchies)
  $\rightarrow$ reduces search spaces significantly
  $\rightarrow$ less opportunities to find alternatives



Optimization Potential

What can we improve?

basic engineering (X-CHASEV)

- apply faster speed-up techniques (CHASE)
- precompute and store frequently used data (preunpacked shortcuts)

Optimization Potential

What can we improve?

basic engineering (X-CHASEV)

- apply faster speed-up techniques (CHASE)
- precompute and store frequently used data (preunpacked shortcuts)

initial idea

- computation of via nodes is costly
  - full search space exploration
  - evaluation of candidates
- store all via nodes?

  $\rightarrow$ quadratic overhead in number of nodes...

Optimization Potential

What can we improve?

basic engineering (X-CHASEV)

- apply faster speed-up techniques (CHASE)
- precompute and store frequently used data (preunpacked shortcuts)

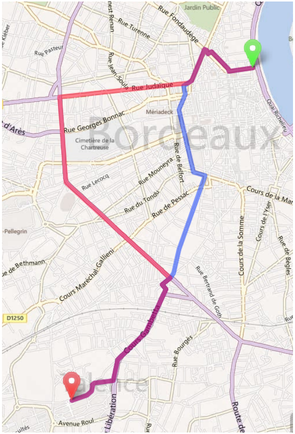
initial idea

- computation of via nodes is costly
  - full search space exploration
  - evaluation of candidates
- store all via nodes?

  $\longrightarrow$ quadratic overhead in number of nodes. . . you don't want to do that!

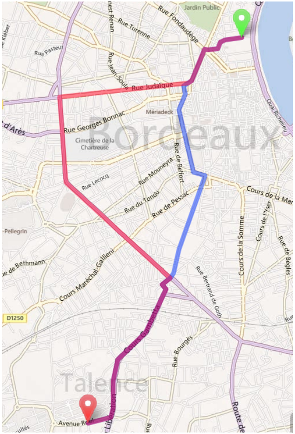
Optimization Potential

Can we still find a more substantial improvement?



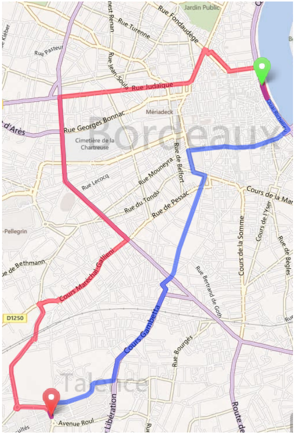
Optimization Potential

Can we still find a more substantial improvement?



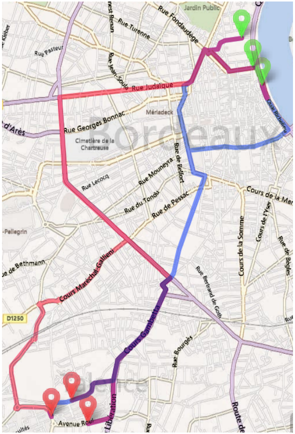
Optimization Potential

Can we still find a more substantial improvement?



Optimization Potential

Can we still find a more substantial improvement?

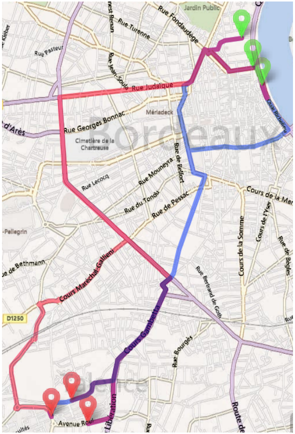


Optimization Potential

Can we still find a more substantial improvement?

observation

- alternatives between regions share a lot



Optimization Potential

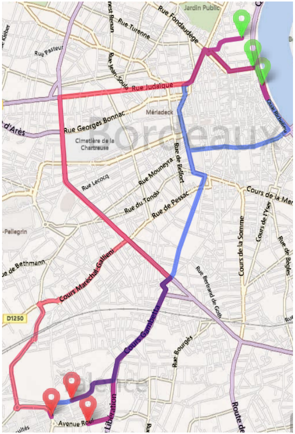
Can we still find a more substantial improvement?

observation

- alternatives between regions share a lot

- well-known fact for shortest paths

  → shortest paths entering/leaving a region are covered by small number of nodes [Abraham et al. 10b]



Optimization Potential

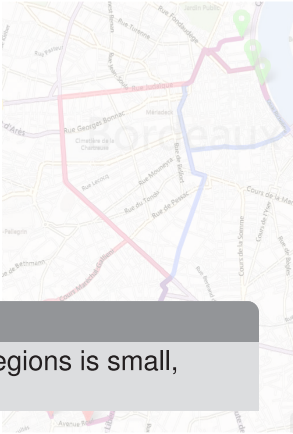
Can we still find a more substantial improvement?

Karlsruhe Institute of Technology

observation

- alternatives between regions share a lot

- well-known fact for shortest paths
  - → shortest paths entering/leaving a region are covered by small number of nodes [Abraham et al. 10b]



combined assumption

If the number of shortest paths between two regions is small, so is the number of viable alternatives.

→ they can be covered by few nodes

Optimization Potential

Can we still find a more substantial improvement? Yes we can!

Karlsruhe Institute of Technology

How to profit from this assumption?

- graph partitioning
  $\longrightarrow$ group nodes with similar shortest path characteristics
  (and alternative route characteristics)

- for each pair of regions store a *via node candidate set*
  $\longrightarrow$ nodes that cover (good) alternatives between this region pair
  $\longrightarrow$ only evaluate these candidates
  (search space exploration no longer needed)

Optimization Potential

Can we still find a more substantial improvement? Yes we can!

Karlsruhe Institute of Technology

How to profit from this assumption?

- graph partitioning
  $\longrightarrow$ group nodes with similar shortest path characteristics
  (and alternative route characteristics)

- for each pair of regions store a *via node candidate set*
  $\longrightarrow$ nodes that cover (good) alternatives between this region pair
  $\longrightarrow$ only evaluate these candidates
  (search space exploration no longer needed)

$\longrightarrow$ **single-level approach**
(preprocessing, query)

Query

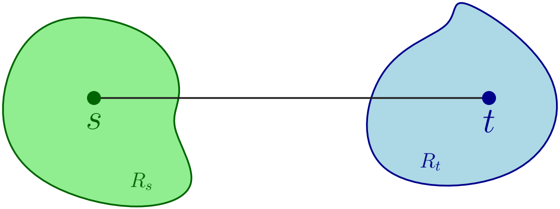
How to find an alternative between s and t?

- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)

Query

How to find an alternative between s and t?

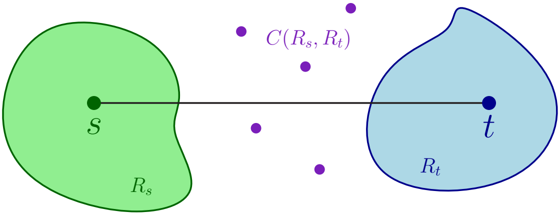
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

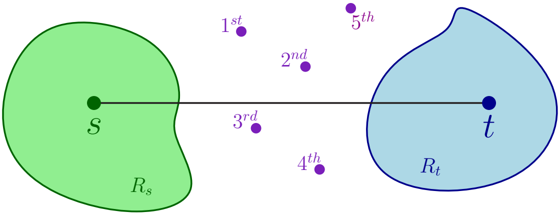
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

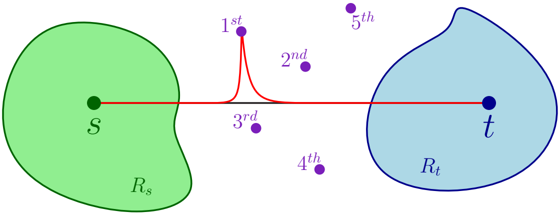
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

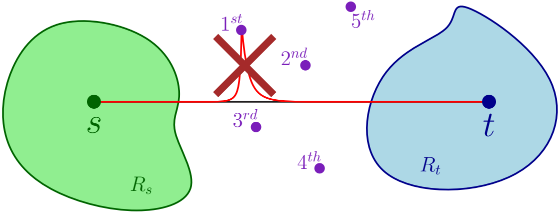
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t\rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

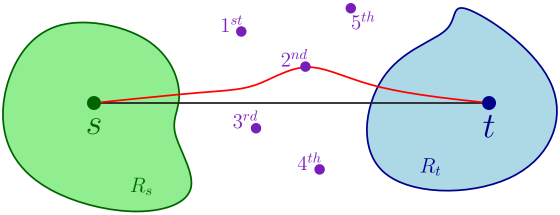
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

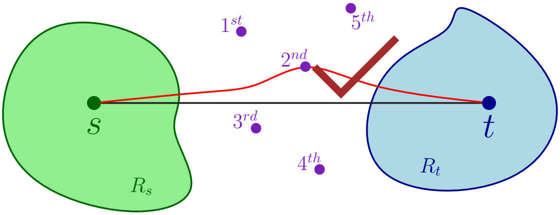
- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t\rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)



Query

How to find an alternative between s and t?

- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)

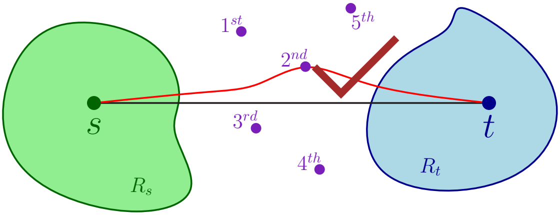


Query

How to find an alternative between s and t?

- look up respective via node candidate set $C(R_s, R_t)$
- check if path $\langle s..v..t \rangle$ is a viable alternative, $v \in C(R_s, R_t)$
  $\longrightarrow$ stop as soon as one is found (greedy)
  $\longrightarrow$ iterate over all candidates to find best alternative (full processing)
  (optimization: sort via node candidates in order of importance)

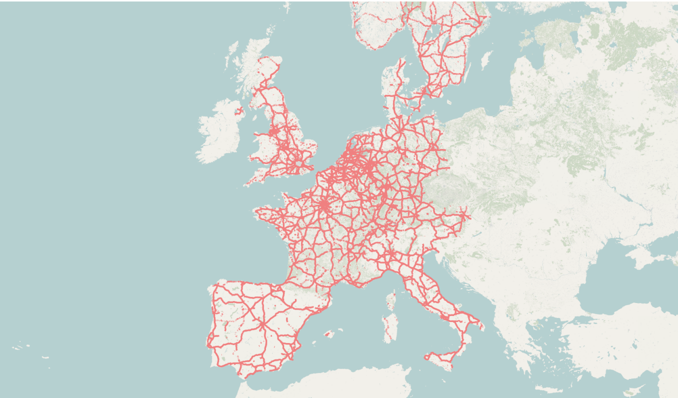
$\longrightarrow$ checking candidates much faster than search space exploration



Preprocessing

How to compute via node candidate sets $C(R_i, R_j)$

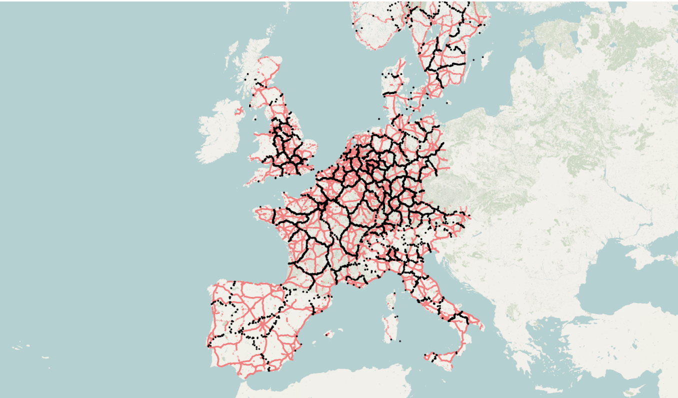
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

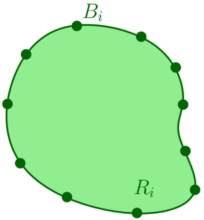
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])

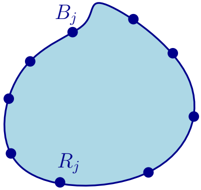


Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)

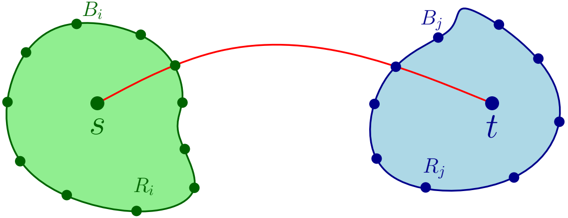




Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

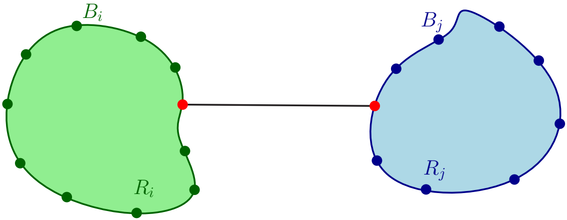
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

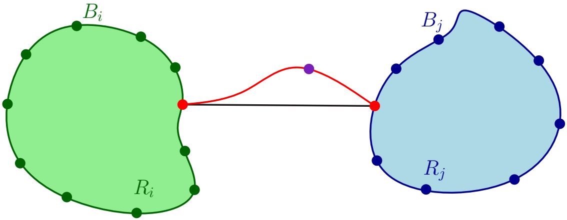
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

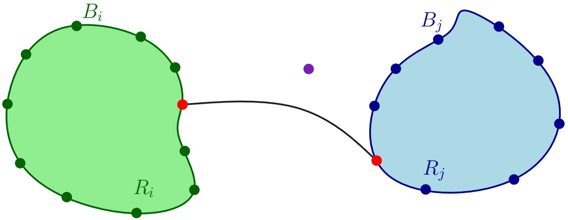
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

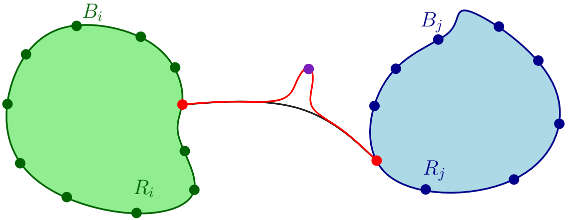
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

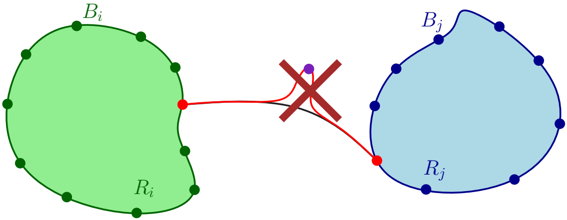
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

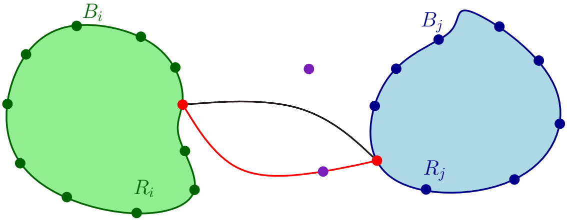
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\longrightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

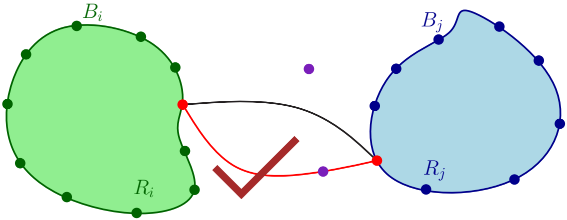
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

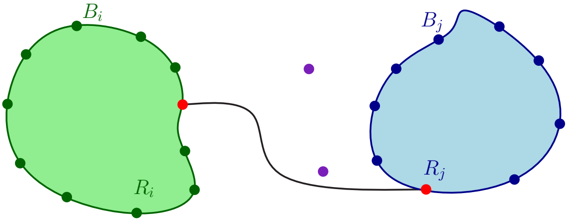
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\longrightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

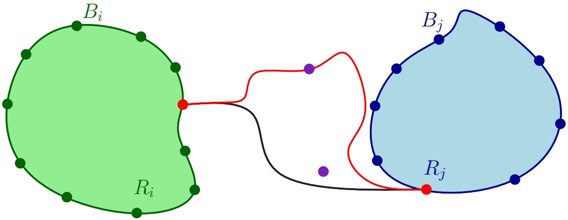
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

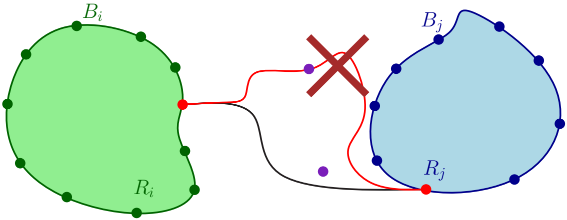
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C}(\mathbf{R_i}, \mathbf{R_j})$

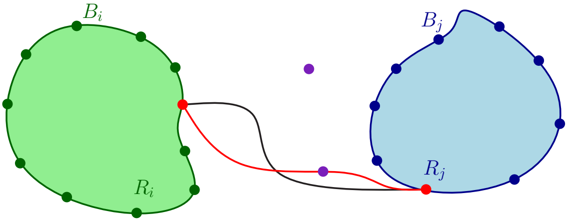
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C(R_i, R_j)}$

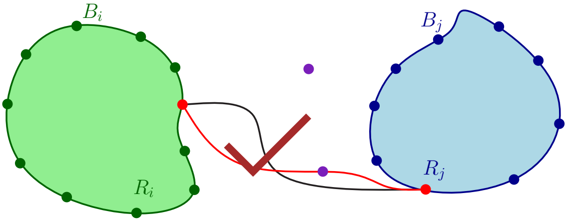
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  → f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $\mathbf{C(R_i, R_j)}$

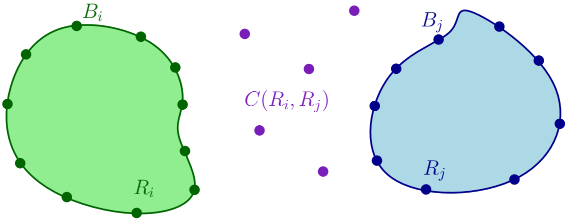
- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one



Preprocessing

How to compute via node candidate sets $C(R_i, R_j)$

- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one

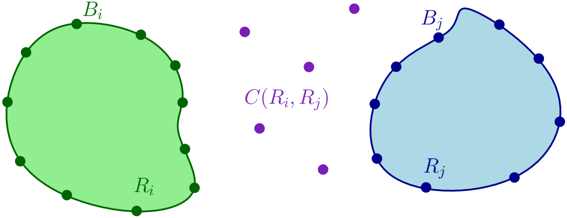


Preprocessing

How to compute via node candidate sets $\mathbf{C(R_i, R_j)}$

- graph partitioning (128 regions, Buffoon [Sanders, Schulz 11])
- alternatives between region pairs R_i, R_j cross border nodes B_i, B_j
  $\rightarrow$ f.e. region pair find alternatives between all border nodes (store via nodes)
- *bootstrapping*
  - check all known via nodes first
  - if none is viable, compute a new one

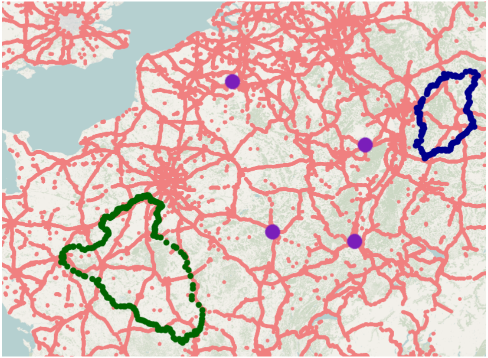
$\rightarrow$ quadratic overhead, but in number of regions ($\sim$ linear in number of nodes)



One problem remains...

What can we do about it?

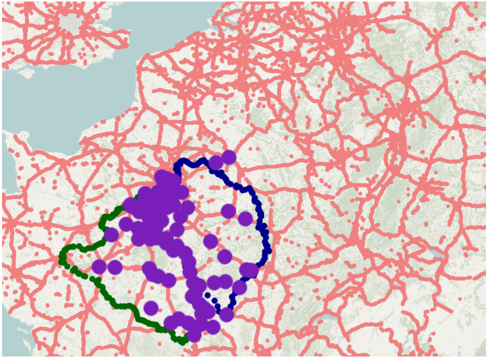
- via sets of neighboring regions are *very* large
  $\longrightarrow$ longer query times, higher space consumption



One problem remains...

What can we do about it?

- via sets of neighboring regions are *very* large
  $\longrightarrow$ longer query times, higher space consumption



One problem remains...

What can we do about it?

- via sets of neighboring regions are *very* large
  $\longrightarrow$ longer query times, higher space consumption

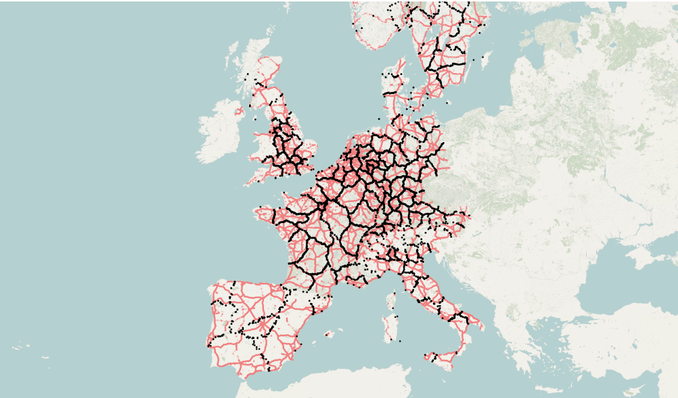
possible solutions

- fallback to baseline algorithm (X-CHASEV)
  (for neighboring regions / within one region)
- multi-level approach
  (additional overhead)

Multi-level approach

How does it work?

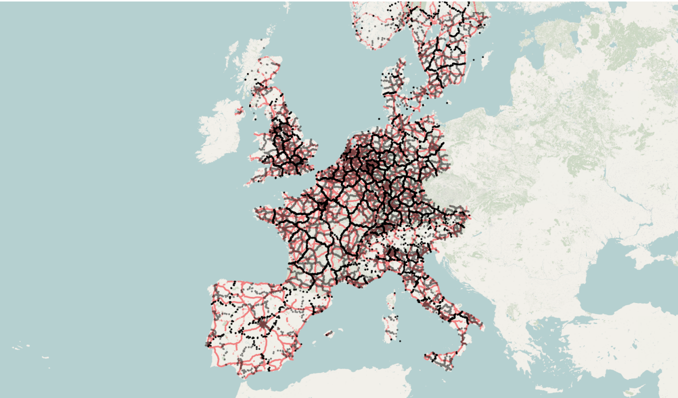
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions



Multi-level approach

How does it work?

- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions



Multi-level approach

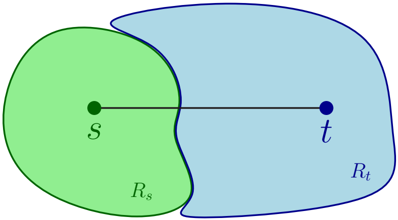
How does it work?

- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed

Multi-level approach

How does it work?

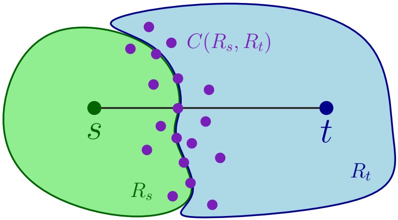
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed



Multi-level approach

How does it work?

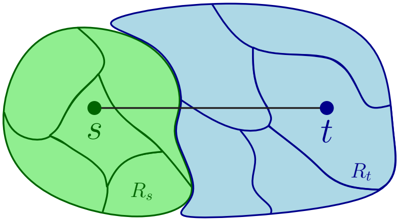
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed



Multi-level approach

How does it work?

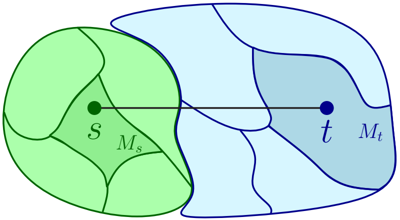
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed



Multi-level approach

How does it work?

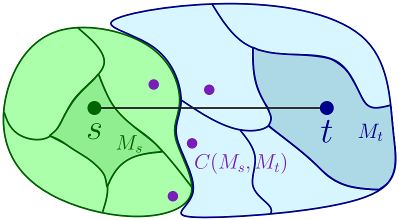
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed



Multi-level approach

How does it work?

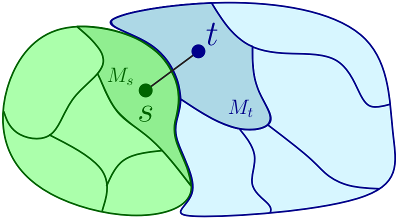
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed



Multi-level approach

How does it work?

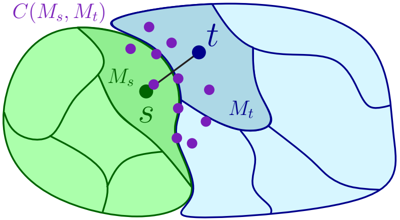
- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed
  - apply baseline algorithm for neighboring finer regions (X-CHASEV)



Multi-level approach

How does it work?

- additional finer graph partitioning (1024 regions, Buffoon)
  - finer regions respect original course regions
- when via node candidate sets get too large, switch to finer regions
  - affected region pairs marked by flag
    $\longrightarrow$ little overhead in query
    $\longrightarrow$ additional preprocessing only done when needed
  - apply baseline algorithm for neighboring finer regions (X-CHASEV)



Results

Preprocessing performance

				candidate sets					
				p=1		p=2		p=3	
relaxed	preprocessing	time [h]	size [kiB]	empty [%]	avg. size	empty [%]	avg. size	empty [%]	avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  $\longrightarrow$ small memory overhead
  $\longrightarrow$ less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  $\longrightarrow$ linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Preprocessing performance

relaxed	preprocessing	time [h]	size [kiB]	candidate sets p=1 empty [%]	p=1 avg. size	p=2 empty [%]	p=2 avg. size	p=3 empty [%]	p=3 avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  → small memory overhead
  → less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  → linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Preprocessing performance

				candidate sets					
				p=1		p=2		p=3	
relaxed	preprocessing	time [h]	size [kiB]	empty [%]	avg. size	empty [%]	avg. size	empty [%]	avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  → small memory overhead
  → less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  → linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Preprocessing performance

				candidate sets					
				p=1		p=2		p=3	
relaxed	preprocessing	time [h]	size [kiB]	empty [%]	avg. size	empty [%]	avg. size	empty [%]	avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  $\longrightarrow$ small memory overhead
  $\longrightarrow$ less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  $\longrightarrow$ linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Preprocessing performance

				candidate sets					
				p=1		p=2		p=3	
relaxed	preprocessing	time [h]	size [kiB]	empty [%]	avg. size	empty [%]	avg. size	empty [%]	avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  → small memory overhead
  → less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  → linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Preprocessing performance

relaxed	preprocessing	time [h]	size [kiB]	candidate sets p=1 empty [%]	p=1 avg. size	p=2 empty [%]	p=2 avg. size	p=3 empty [%]	p=3 avg. size
–	single-level	1.1	859	2.6	4.4	12.7	5.1	30.5	4.4
–	multi-level	1.7	3 669	6.2	6.1	17.4	5.9	36.9	4.2
✓	single-level	2.3	1 742	1.4	6.7	3.0	10.2	10.8	11.5
✓	multi-level	4.3	8 909	1.1	12.2	4.9	15.0	11.6	14.2

(4 AMD Opteron 6168 (1.90 GHz), 256 GiB main memory, 48 cores)

- via node candidate sets are sparse (even better on connected regions)
  $\longrightarrow$ small memory overhead
  $\longrightarrow$ less than 10 MByte (good for caching)
- preprocessing easily parallelizable
  $\longrightarrow$ linear speed-up until memory bandwidth is reached

(relaxed variant of baseline algorithm with higher success rates but additional overhead)

Results

Query performance

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	1.2	75.5	-	1.7	40.2	-	2.3	14.2	-
X-CHASEV	0.5	75.5	-	0.7	40.2	-	1.0	14.2	-
single-level	0.1	80.7	1.9	0.3	50.8	2.8	0.4	24.8	3.8
multi-level	0.1	81.2	2.0	0.3	51.2	2.9	0.4	25.0	3.8

(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

- alternatives in sub-milliseconds
  $\rightarrow$ up to one order of magnitude faster than X-CHV
- higher success rates
  $\rightarrow$ X-BDV as *"gold standard"*
  $\rightarrow$ relative gap to X-BDV reduced by more than 25%

(relaxed variant with higher speed-ups and similar relative improvement in success rates)

Results

Query performance

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	1.2	75.5	-	1.7	40.2	-	2.3	14.2	-
X-CHASEV	0.5	75.5	-	0.7	40.2	-	1.0	14.2	-
single-level	0.1	80.7	1.9	0.3	50.8	2.8	0.4	24.8	3.8
multi-level	0.1	81.2	2.0	0.3	51.2	2.9	0.4	25.0	3.8

(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

- alternatives in sub-milliseconds
  $\longrightarrow$ up to one order of magnitude faster than X-CHV
- higher success rates
  $\longrightarrow$ X-BDV as *"gold standard"*
  $\longrightarrow$ relative gap to X-BDV reduced by more than 25%

(relaxed variant with higher speed-ups and similar relative improvement in success rates)

Results

Query performance

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	1.2	75.5	-	1.7	40.2	-	2.3	14.2	-
X-CHASEV	0.5	75.5	-	0.7	40.2	-	1.0	14.2	-
single-level	0.1	80.7	1.9	0.3	50.8	2.8	0.4	24.8	3.8
multi-level	0.1	81.2	2.0	0.3	51.2	2.9	0.4	25.0	3.8

(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

- alternatives in sub-milliseconds
  $\longrightarrow$ up to one order of magnitude faster than X-CHV
- higher success rates
  $\longrightarrow$ X-BDV as *"gold standard"*
  $\longrightarrow$ relative gap to X-BDV reduced by more than 25%

(relaxed variant with higher speed-ups and similar relative improvement in success rates)

Results

Query performance

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	1.2	75.5	-	1.7	40.2	-	2.3	14.2	-
X-CHASEV	0.5	75.5	-	0.7	40.2	-	1.0	14.2	-
single-level	0.1	80.7	1.9	0.3	50.8	2.8	0.4	24.8	3.8
multi-level	0.1	81.2	2.0	0.3	51.2	2.9	0.4	25.0	3.8

(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

- alternatives in sub-milliseconds
  → up to one order of magnitude faster than X-CHV
- higher success rates
  → X-BDV as *"gold standard"*
  → relative gap to X-BDV reduced by more than 25%

(relaxed variant with higher speed-ups and similar relative improvement in success rates)

Results

Query performance

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	1.2	75.5	-	1.7	40.2	-	2.3	14.2	-
X-CHASEV	0.5	75.5	-	0.7	40.2	-	1.0	14.2	-
single-level	0.1	80.7	1.9	0.3	50.8	2.8	0.4	24.8	3.8
multi-level	0.1	81.2	2.0	0.3	51.2	2.9	0.4	25.0	3.8

(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

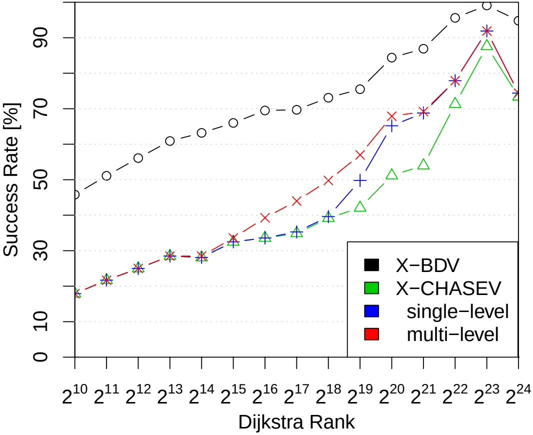
- alternatives in sub-milliseconds
  → up to one order of magnitude faster than X-CHV
- higher success rates
  → X-BDV as *"gold standard"*
  → relative gap to X-BDV reduced by more than 25%

(relaxed variant with higher speed-ups and similar relative improvement in success rates)

Results

Local queries (first alternative)

- success rates compared to shortest path lengths
- highest improvement for mid-range queries
  (avg. region sizes: 2^{15}, 2^{18})



(relaxed variant ≈ 5% below X-BDV)

Online Setting

Application I

Properties

- learn via node candidate sets from stream of queries
- applicable to legacy system (that implements some baseline algorithm)
- only partitioning required in advance

Online Setting

Application I

Karlsruhe Institute of Technology

Properties

- learn via node candidate sets from stream of queries
- applicable to legacy system (that implements some baseline algorithm)
- only partitioning required in advance

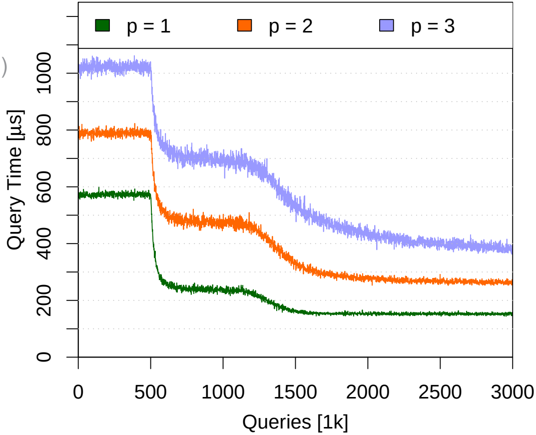
Algorithm

- start with empty via node candidate sets
- apply our *single-level approach*
  (check if existing candidate yields viable alternative)
- if no alternative is found
  - apply baseline algorithm (X-CHASEV)
  - if alternative is found: store its via node
- stop using baseline algorithm after some threshold is reached
  (except for neighboring regions / within one region)

Online Setting

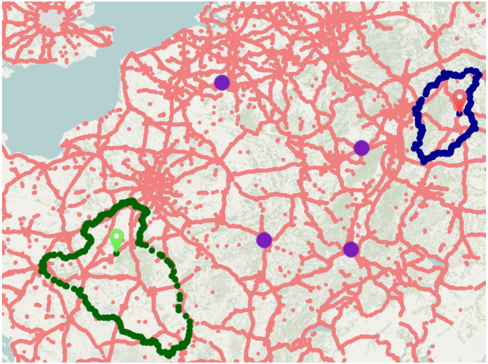
Simulation results

- baseline algorithm for first 500k queries (no learning)
- rapid fall in query times as our algorithm is applied (learning phase ≈ 100k queries)
- second decline when thresholds are reached (50 queries for each region pair)



Alternative Graphs
Application II

Properties
- fast to compute, little overhead
- two variants (with and without additional preprocessing)



Alternative Graphs

Application II

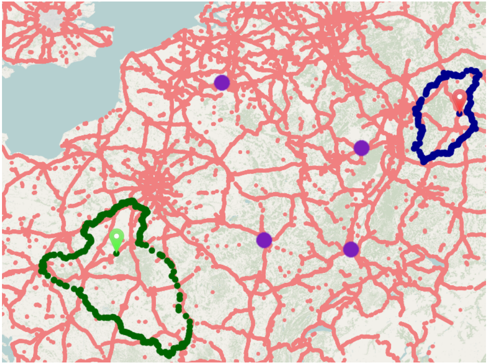
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

  $\rightarrow$ computes superset



Alternative Graphs

Application II

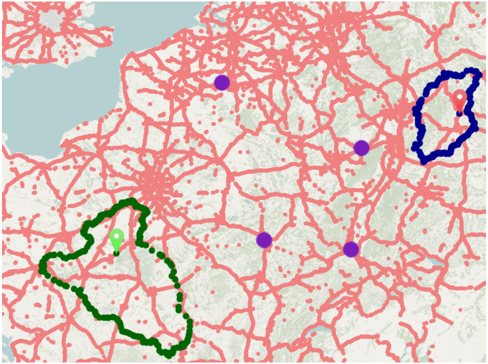
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

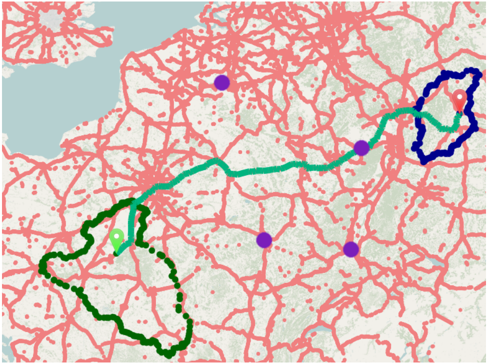
$\longrightarrow$ computes superset



Alternative Graphs
Application II

Properties
- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction
- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)
  - $\rightarrow$ computes superset



Alternative Graphs

Application II

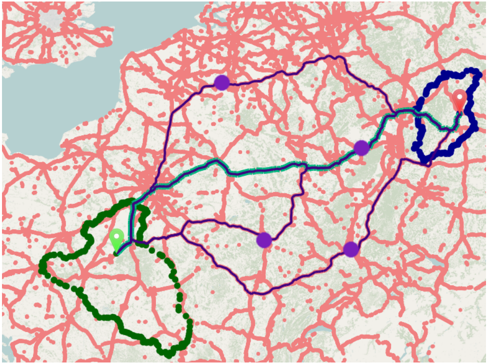
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

  $\longrightarrow$ computes superset



Alternative Graphs

Application II

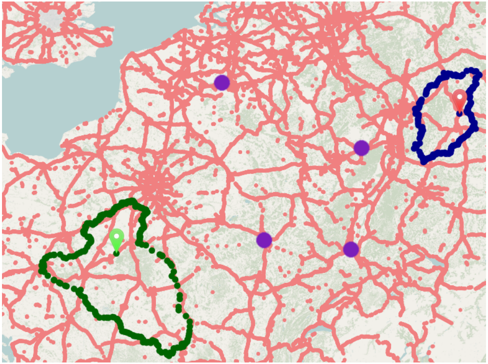
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

$\longrightarrow$ computes superset



Alternative Graphs

Application II

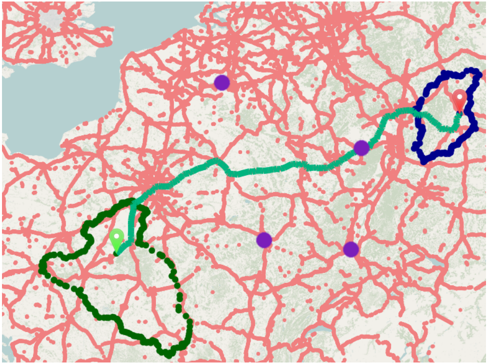
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

  $\longrightarrow$ computes superset



Alternative Graphs

Application II

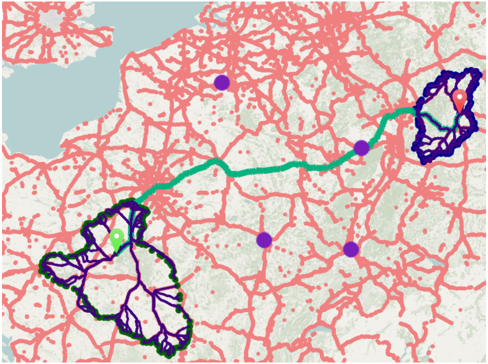
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

$\longrightarrow$ computes superset



Alternative Graphs

Application II

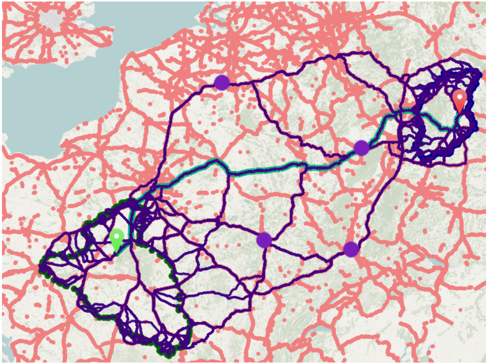
Properties

- fast to compute, little overhead
- two variants (with and without additional preprocessing)

Construction

- combination of shortest paths
- *base method*
  - s to t
  - s to $C(R_s, R_t)$ to t
- *enhanced method*
  - s to t
  - s to B_s / B_t to t
  - B_s to $C(R_s, R_t)$ to B_t (prepro.)

$\longrightarrow$ computes superset



Conclusion

Karlsruhe Institute of Technology

Summary

- improvement in query times (one order of magnitude)
- lowered quality gap to X-BDV (25% and more)
- negligible memory footprint (less than 10 MByte)
- applications (online setting, alternative graphs)

Outlook

- theoretical foundations (using highway dimension)
- alternatives with more via nodes (similar to transit node routing)

Thank you for your attention!

Time for questions!

References

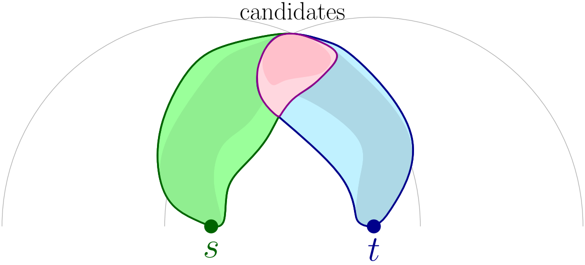
- [Abraham et al. 10a]
  Alternative Routes in Road Networks
- [Abraham et al. 10b]
  Highway Dimension, Shortest Paths and Provably Efficient Algorithms
- [Bader et al. 11]
  Alternative Route Graphs in Road Networks

Alternatives Routes

How do you find them, quicker and more often?

relaxed Contraction Hierarchies

- artificially increase search space (allow descent up to *x* levels)
  → improves success rate
- adjustable trade-off: speed vs. success rate



Results

Query performance (relaxed CH)

algorithm	p=1 time [ms]	p=1 success rate[%]	p=1 avg. tested	p=2 time [ms]	p=2 success rate[%]	p=2 avg. tested	p=3 time [ms]	p=3 success rate[%]	p=3 avg. tested
X-BDV	11.5s	94.5	-	12.2s	80.6	-	13.3s	59.5	-
X-CHV	3.4	88.5	-	4.3	64.7	-	5.3	38.0	-
X-CHASEV	2.7	88.5	-	3.2	64.7	-	3.8	38.0	-
single-level	0.2	90.0	2.2	0.4	70.2	3.8	0.6	44.0	5.6
multi-level	0.1	90.0	2.3	0.3	70.4	4.0	0.5	44.2	5.8

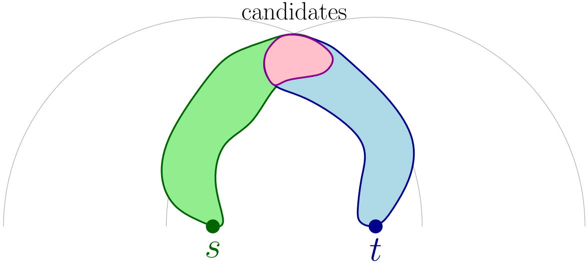
(Intel Core i7-920 (2.66 GHz), 12 GiB main memory, single core)

- alternatives in sub-milliseconds
  → *more than* one order of magnitude faster than X-CHV
- higher success rates
  → relative gap to X-BDV reduced by more than 25%

Results
Query performance

Why does *single-level*/*multi-level* improve success rates?

- via node candidates derived from alternatives between border nodes
  $\longrightarrow$ combination of search spaces of all border nodes
  $\longrightarrow$ larger overlap
  $\longrightarrow$ more chances to encounter viable via nodes

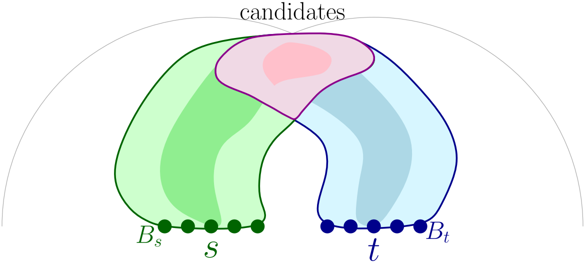


Results

Query performance

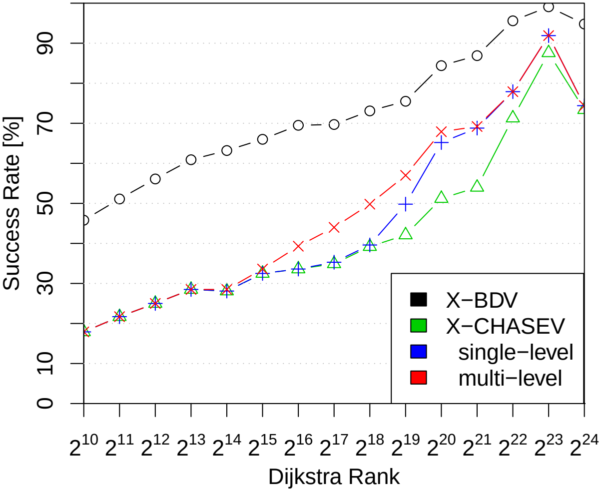
Why does *single-level*/*multi-level* improve success rates?

- via node candidates derived from alternatives between border nodes
  $\longrightarrow$ combination of search spaces of all border nodes
  $\longrightarrow$ larger overlap
  $\longrightarrow$ more chances to encounter viable via nodes



Results

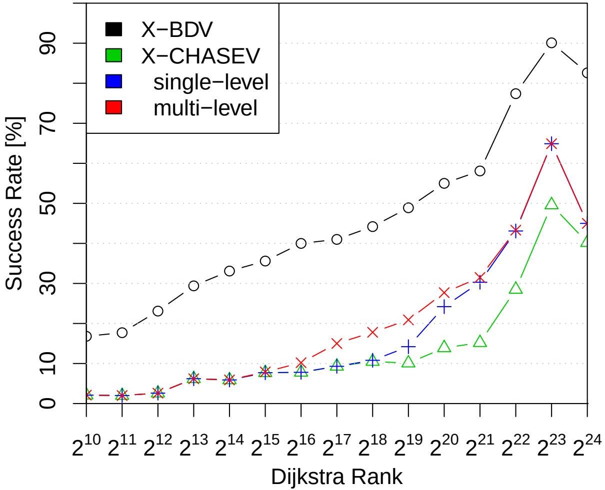
Local queries (basic CH)
first alternative



Results

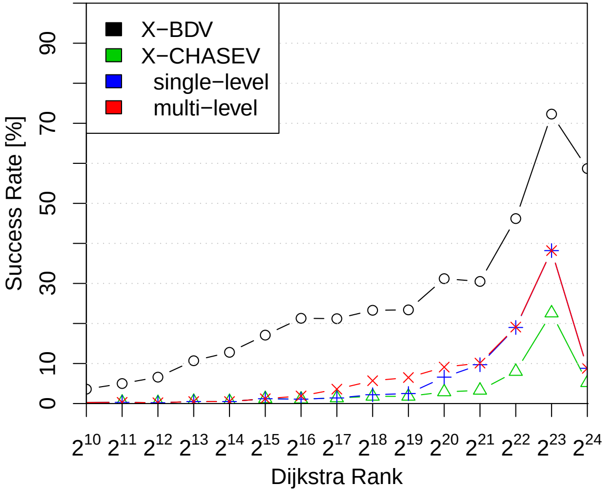
Local queries (basic CH)

second alternative



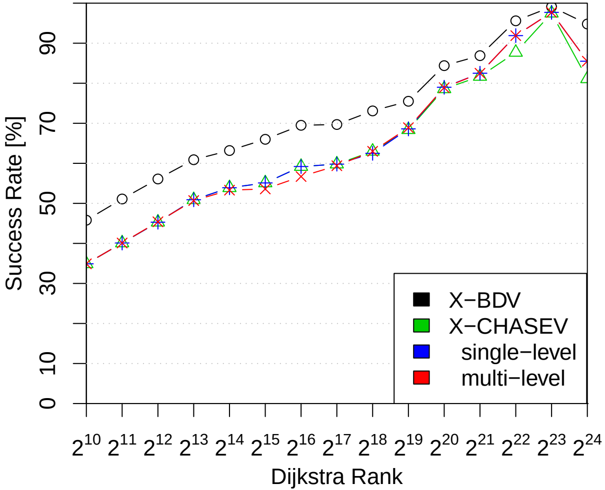
Results

Local queries (basic CH)
third alternative



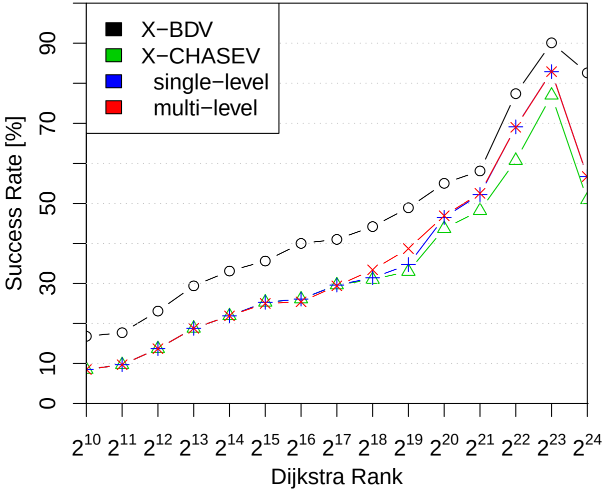
Results

Local queries (relaxed CH)
first alternative



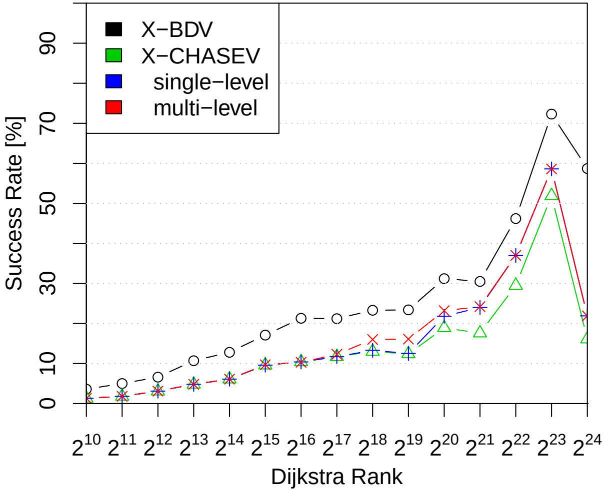
Results

Local queries (relaxed CH) second alternative



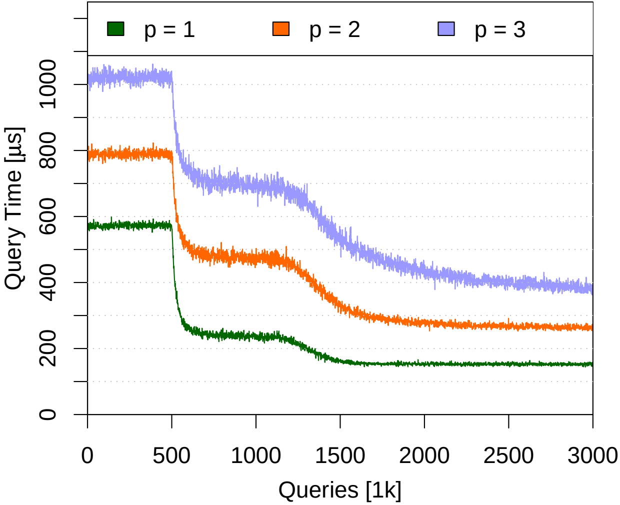
Results

Local queries (relaxed CH)
third alternative



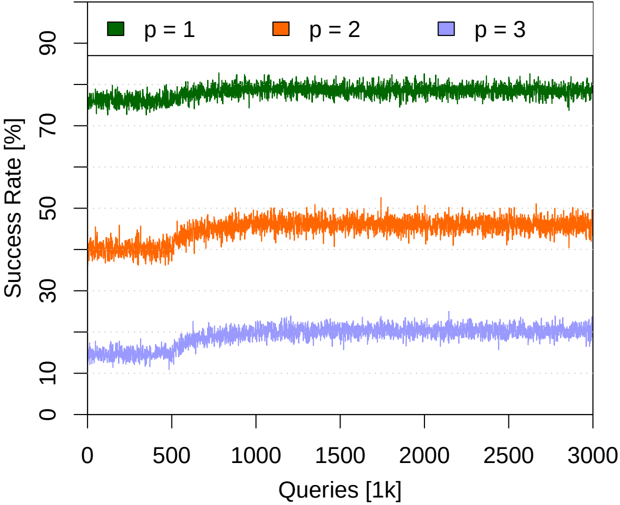
Online Setting

Simulation results (basic CH)



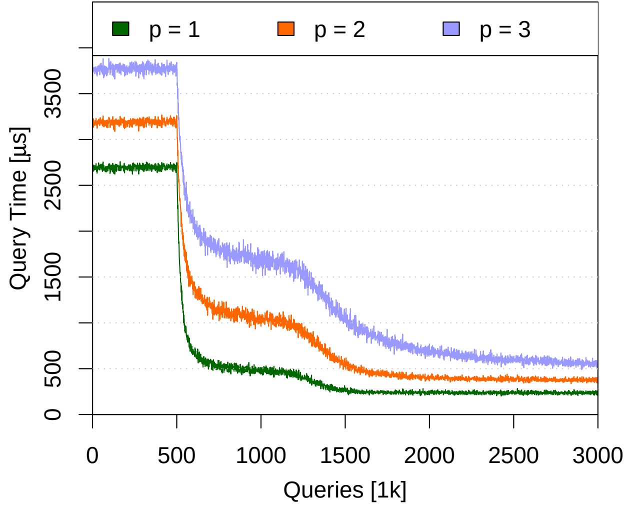
Online Setting

Simulation results (basic CH)



Online Setting

Simulation results (relaxed CH)



Online Setting

Simulation results (relaxed CH)