Karlsruhe Institute of Technology

Efficient Route Compression for Hybrid Route Planning

Gernot Veit Batz, Robert Geisberger, Dennis Luxen, Peter Sanders, and Roman Zubkov
{batz, luxen, sanders}@kit.edu

Institute of Theoretical Informatics, Algorithmics II

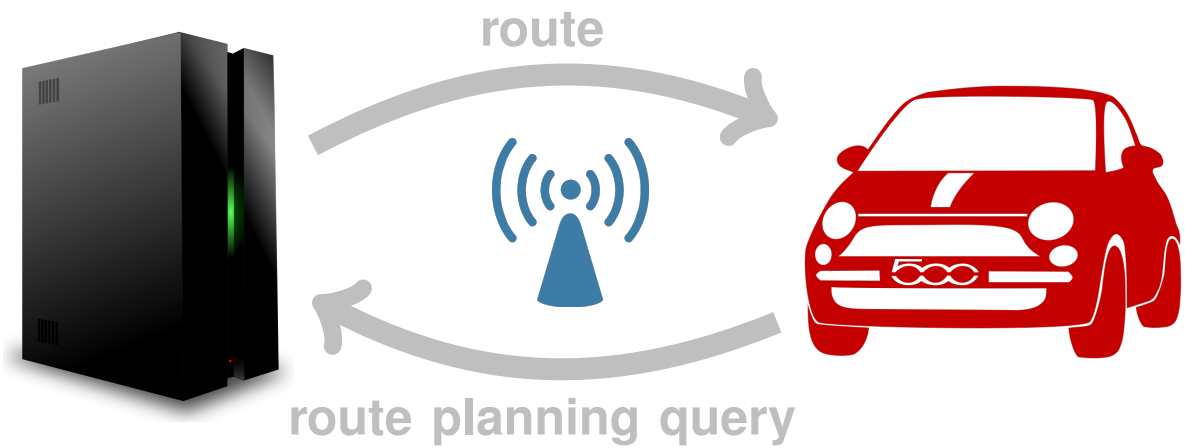
Motivation

Why is hybrid route planning interesting?

Why is it a problem?

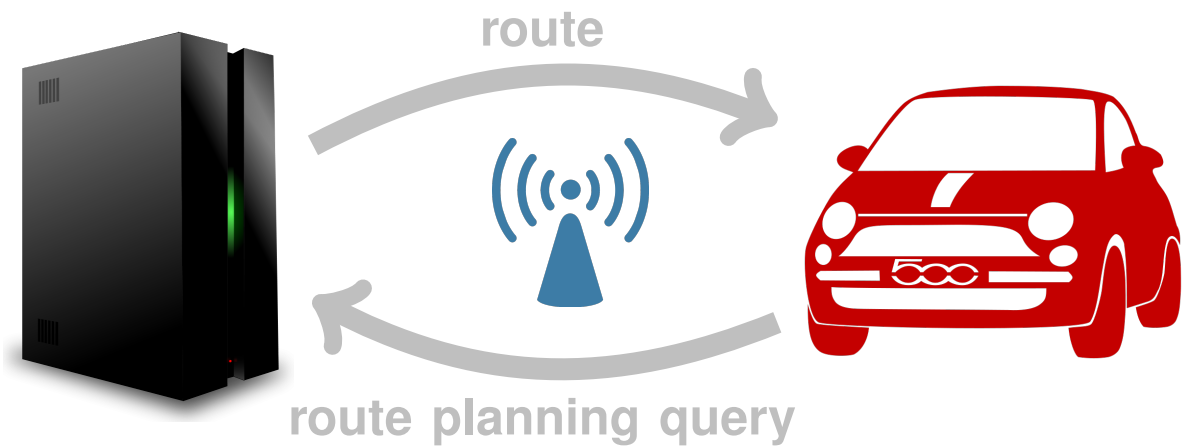
Mobile vs. Hybrid Route Planning

- **Mobile route planning:**
  Routes computed by mobile device in the car
- **Hybrid route planning:**
  Routes computed by server, then transmitted to car



Mobile vs. Hybrid Route Planning

- **Mobile route planning:**
  Routes computed by mobile device in the car
- **Hybrid route planning:**
  Routes computed by server, then transmitted to car



⇒ Hybrid route planning requires mobile radio communication

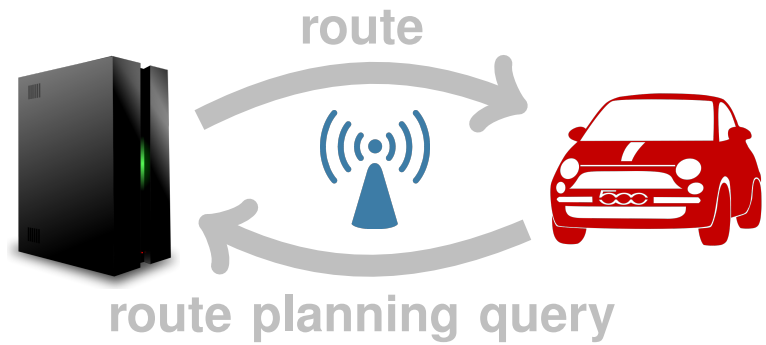
Benefits of Hybrid Route Planning

Advanced route planning algorithms

- high quality routes within milliseconds
- suited for server systems
- difficult to adapt to mobile devices

Hybrid route planning...

- makes server algorithms...
- ...available to car drivers



Server-Based Route Planning (1)

Time-dependent route planning:

[TD SHARC 08] [TCH 09] [ATCH 10] [Brunel et al. 10] [Batz, Sanders 12]

- exploit statistical data, e.g., rush hour
- yields time-dependent edge weights
- route depends on time of day

Flexible route planning:

[flexible CH 10]

- dynamically fine tune cost function...
- ...with a parameter: $c_e + p \cdot d_e$
- tradeoff, e.g., energy cost vs. travel time

Server-Based Route Planning (1)

Time-dependent route planning:

[TD SHARC 08] [TCH 09] [ATCH 10] [Brunel et al. 10] [Batz, Sanders 12]

- exploit statistical data, e.g., rush hour
- yields time-dependent edge weights
- route depends on time of day

Flexible route planning:

[flexible CH 10]

- dynamically fine tune cost function...
- ...with a parameter: $c_e + p \cdot d_e$
- tradeoff, e.g., energy cost vs. travel time

Server-Based Route Planning (1)

Time-dependent route planning:

[TD SHARC 08] [TCH 09] [ATCH 10] [Brunel et al. 10] [Batz, Sanders 12]

- exploit statistical data, e.g., rush hour
- yields time-dependent edge weights
- route depends on time of day

Flexible route planning:

[flexible CH 10]

- dynamically fine tune cost function...
- ...with a parameter: $c_e + p \cdot d_e$
- tradeoff, e.g., energy cost vs. travel time

Server-Based Route Planning (2)

Customizable route planning:

[Delling et al. 11]

- dynamically change cost function
- e.g., unexpected traffic situations

Multi-criteria route planning:

[Delling, Wagner 09]

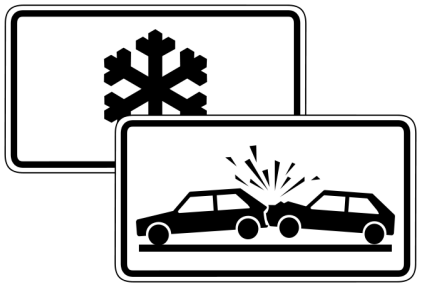
- deal with multiple incomparable costs
- e.g., travel time vs. inconveniency

Server-Based Route Planning (2)

Customizable route planning:

[Delling et al. 11]

- dynamically change cost function
- e.g., unexpected traffic situations



Multi-criteria route planning:

[Delling, Wagner 09]

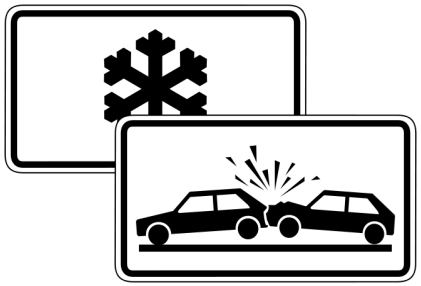
- deal with multiple incomparable costs
- e.g., travel time vs. inconveniency

Server-Based Route Planning (2)

Customizable route planning:

[Delling et al. 11]

- dynamically change cost function
- e.g., unexpected traffic situations



Multi-criteria route planning:

[Delling, Wagner 09]

- deal with multiple incomparable costs
- e.g., travel time vs. inconveniency

Server-Based Route Planning (3)

Alternative routes:

[Abraham et al. 10] [Luxen, Schieferdecker 12]

- a handful of alternatives
- nearly as good as optimal route
- reasonable different

Hub-labeling algorithms:

[Abraham et al. 11] [Abraham et al. 12]

- running times $< 1\ \mu$s implemented with C++
- fast route planning with database servers
- more complicated queries, e.g., route via POI

Server-Based Route Planning (3)

Alternative routes:

[Abraham et al. 10] [Luxen, Schieferdecker 12]

- a handful of alternatives
- nearly as good as optimal route
- reasonable different

Hub-labeling algorithms:

[Abraham et al. 11] [Abraham et al. 12]

- running times $< 1\ \mu$s implemented with C++
- fast route planning with database servers
- more complicated queries, e.g., route via POI

Server-Based Route Planning (3)

Alternative routes:

[Abraham et al. 10] [Luxen, Schieferdecker 12]

- a handful of alternatives
- nearly as good as optimal route
- reasonable different

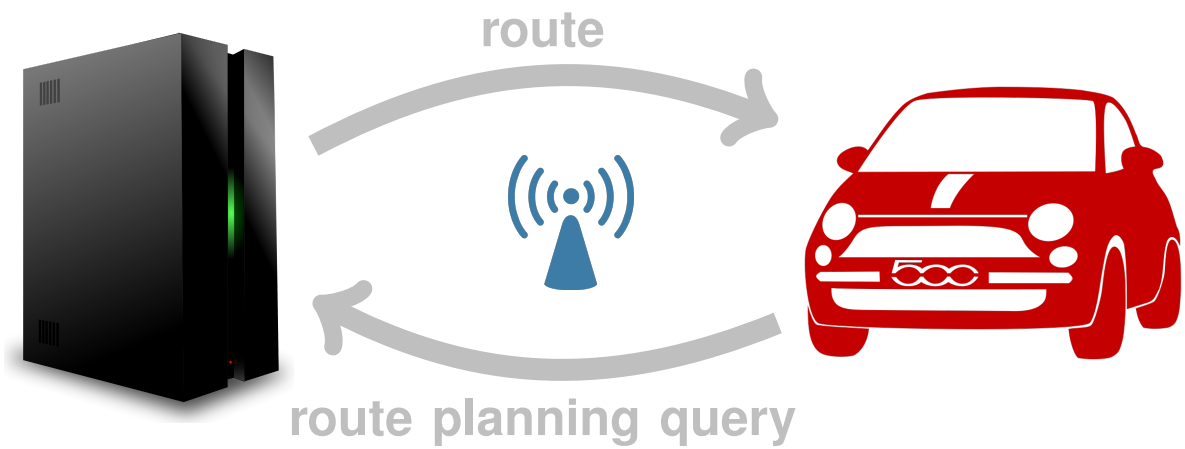
Hub-labeling algorithms:

[Abraham et al. 11] [Abraham et al. 12]

- running times $< 1\ \mu s$ implemented with C++
- fast route planning with database servers
- more complicated queries, e.g., route via POI

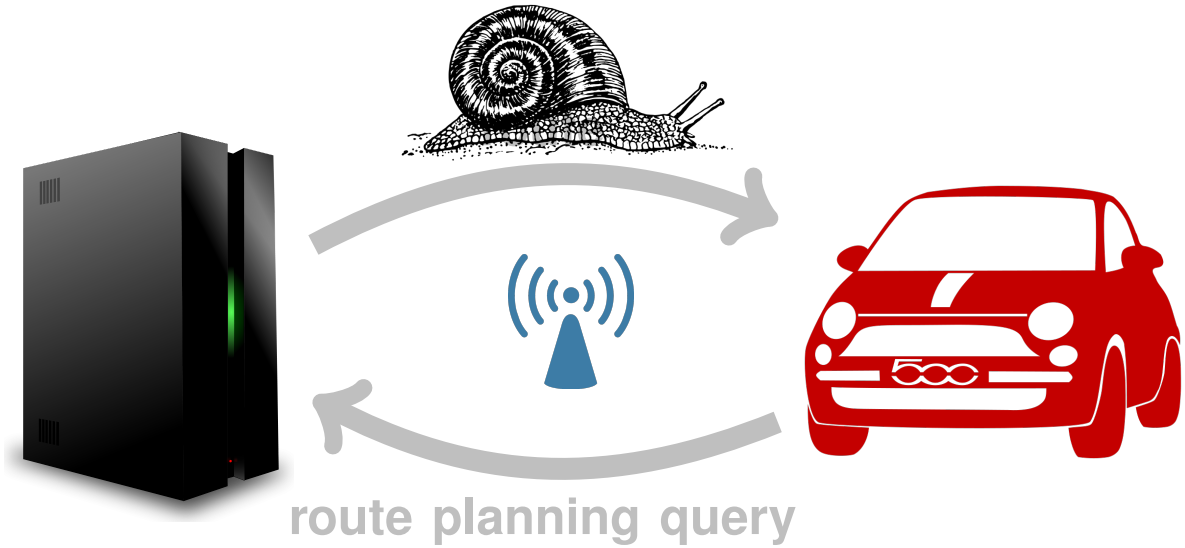
Problems with Radio Communication

- Low bandwidth, but complex routes
- Transmission costs



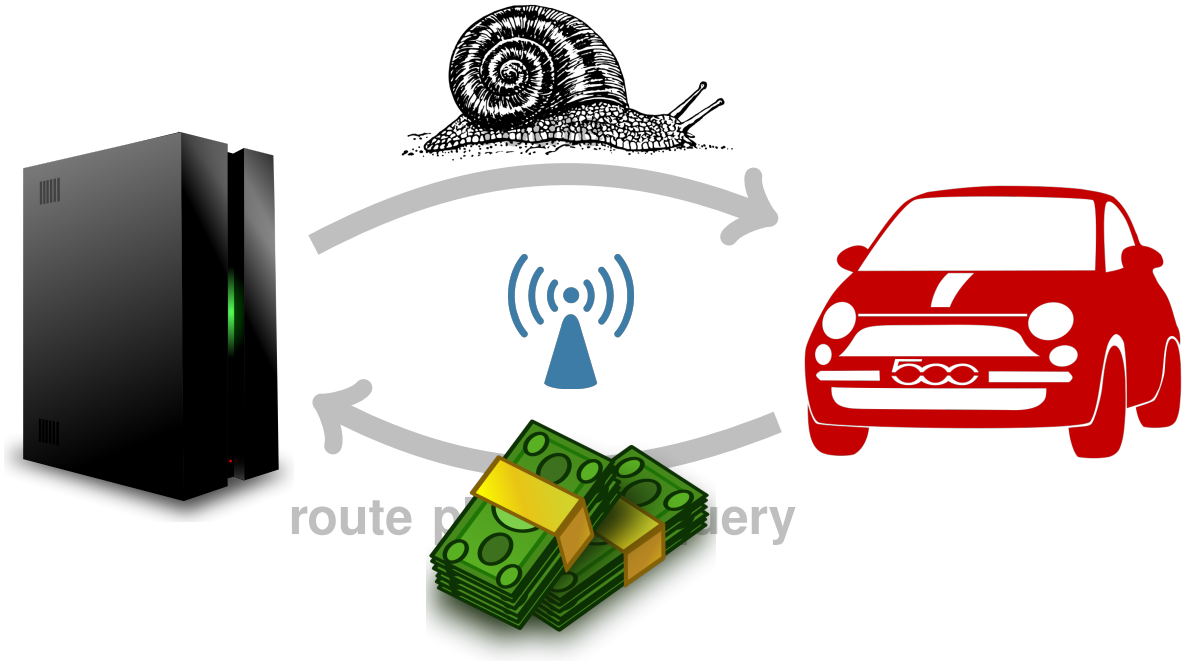
Problems with Radio Communication

- Low bandwidth, but complex routes
- Transmission costs



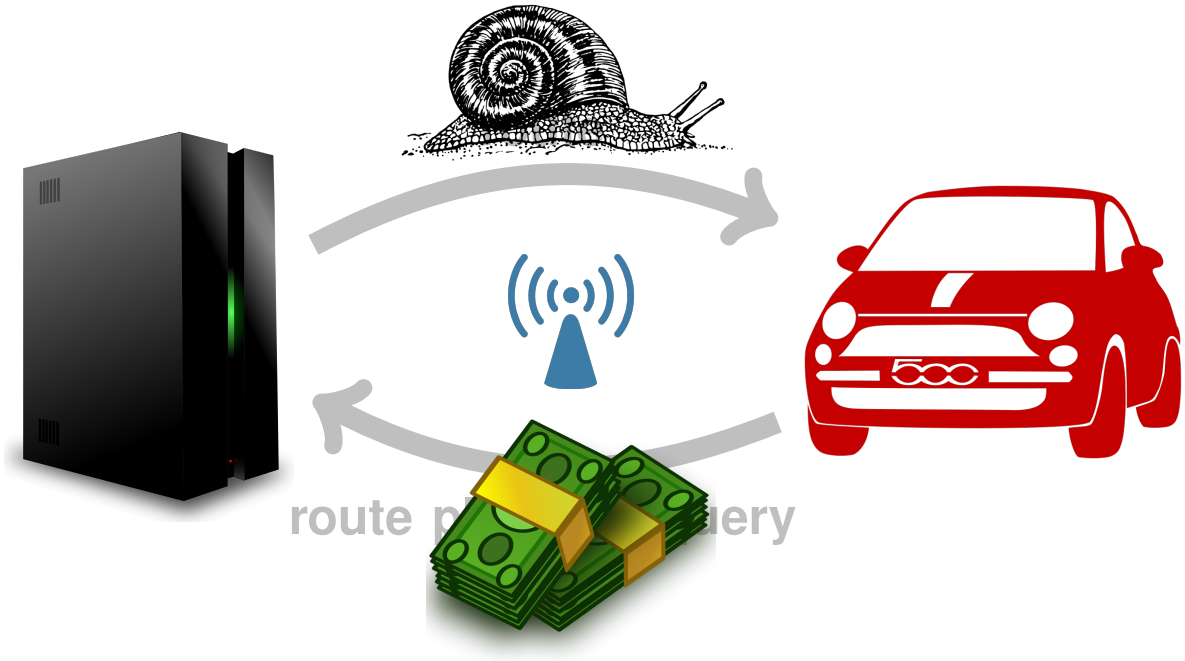
Problems with Radio Communication

- Low bandwidth, but complex routes
- Transmission costs



Problems with Radio Communication

- Low bandwidth, but complex routes
- Transmission costs



⇒ Do data compression!

Contribution

Efficient lossless compression of routes.

Compression of Routes

To Make Hybrid Route Planning Convenient to Use

Fast: User experiences no delay

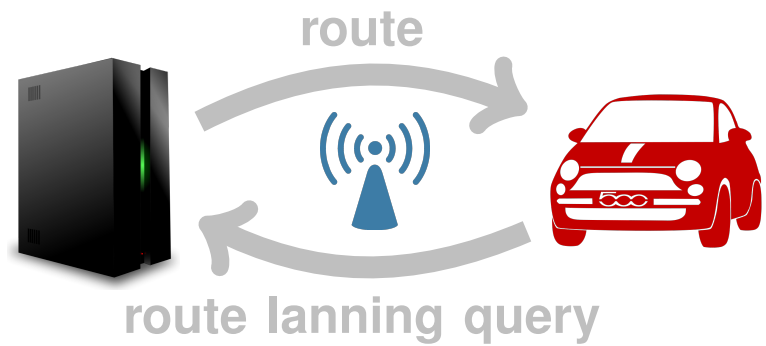
- Driving directions start within ≤ 0.1 sec.
- Fast compression / decompression / transmission

Lossless:

- Reconstructed route = server-provided route

Resulting Requirements:

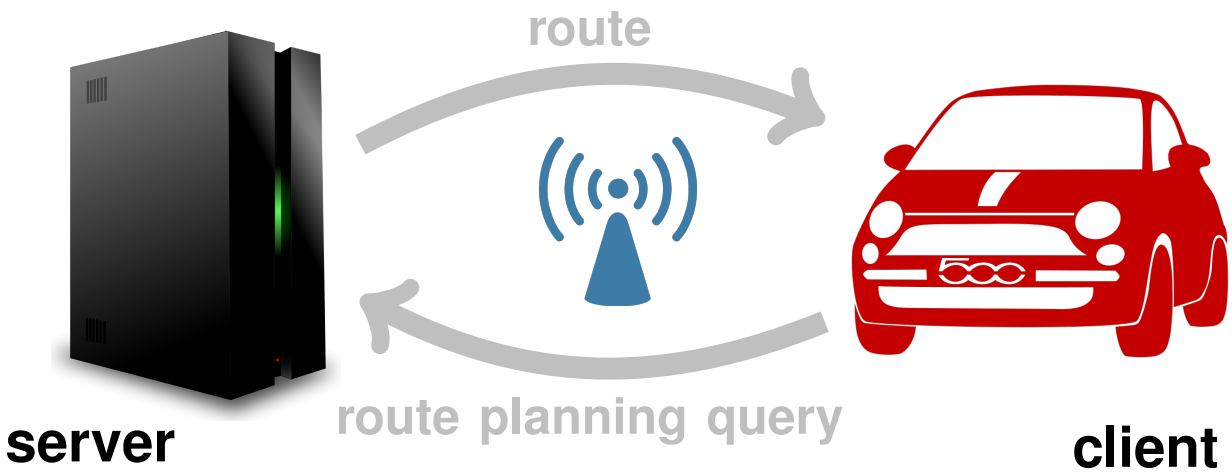
- Fast algorithms
- Good compression rates



Basic Setup...

...Needed by our Approach

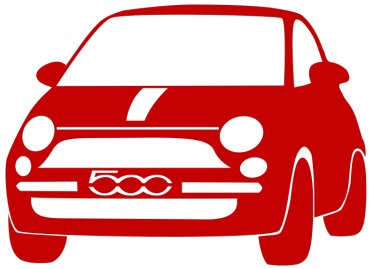
- Client has basic and fast route planning capability
- Client and Server use same road network
- Client uses fixed cost function known to the server



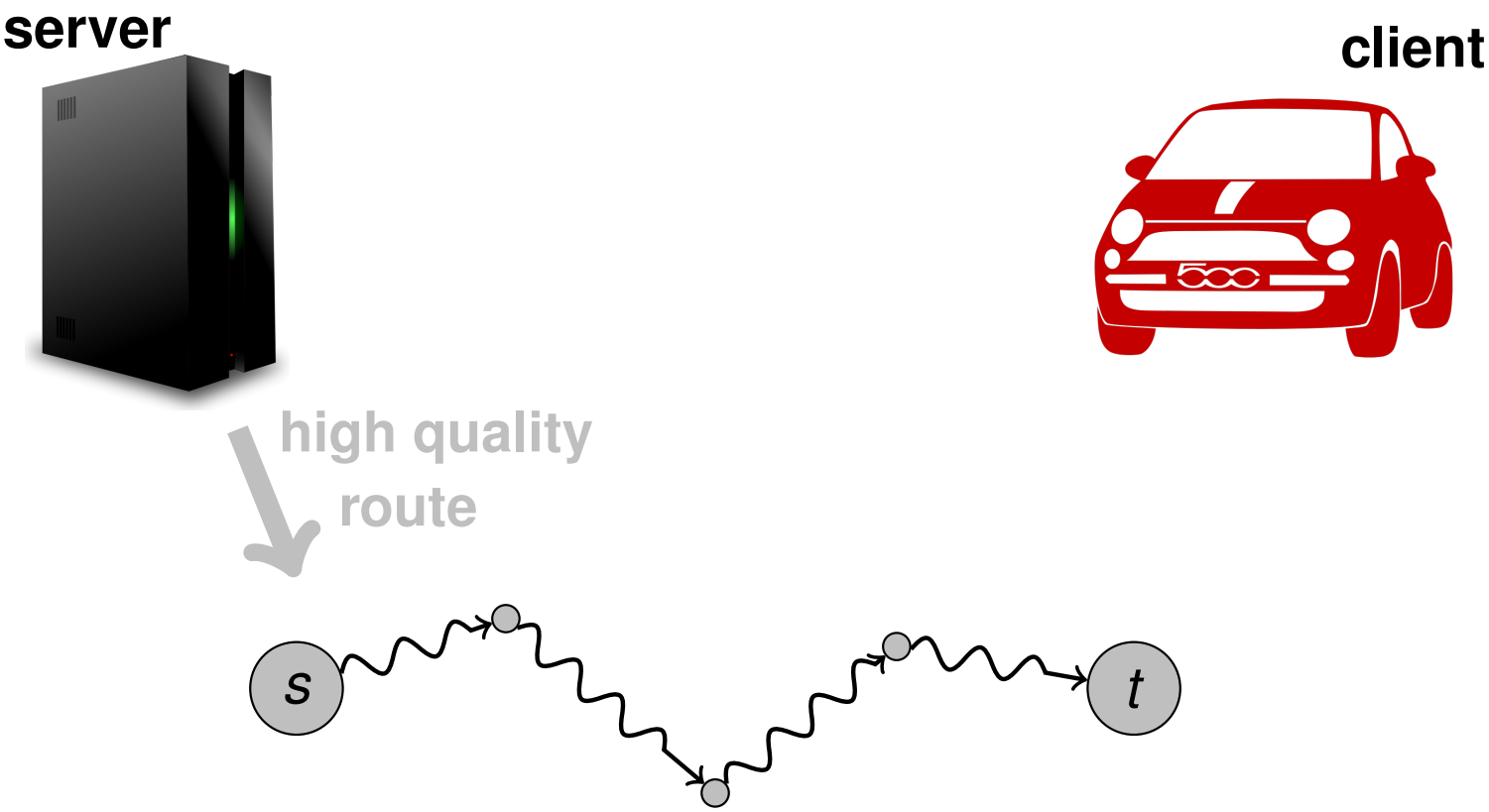
Basic Idea: Use Via Nodes

server

client



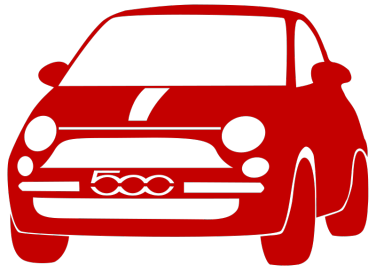
Basic Idea: Use Via Nodes



Basic Idea: Use Via Nodes

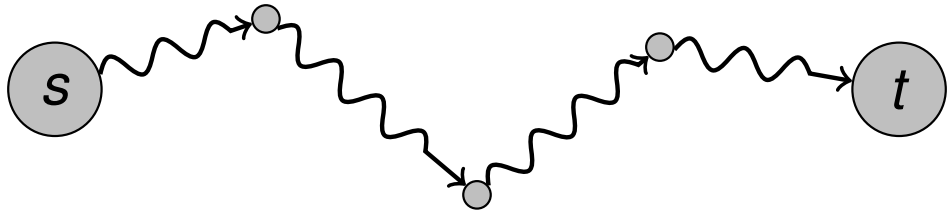
server

client



not optimal with respect to client

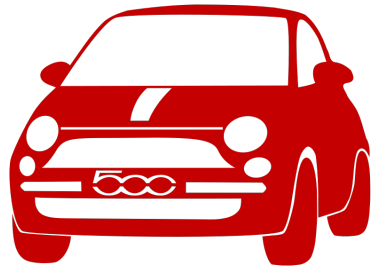
high quality route



Basic Idea: Use Via Nodes

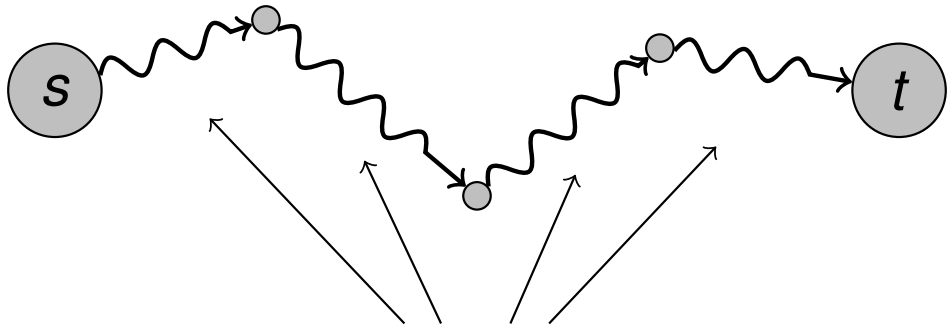
server

client



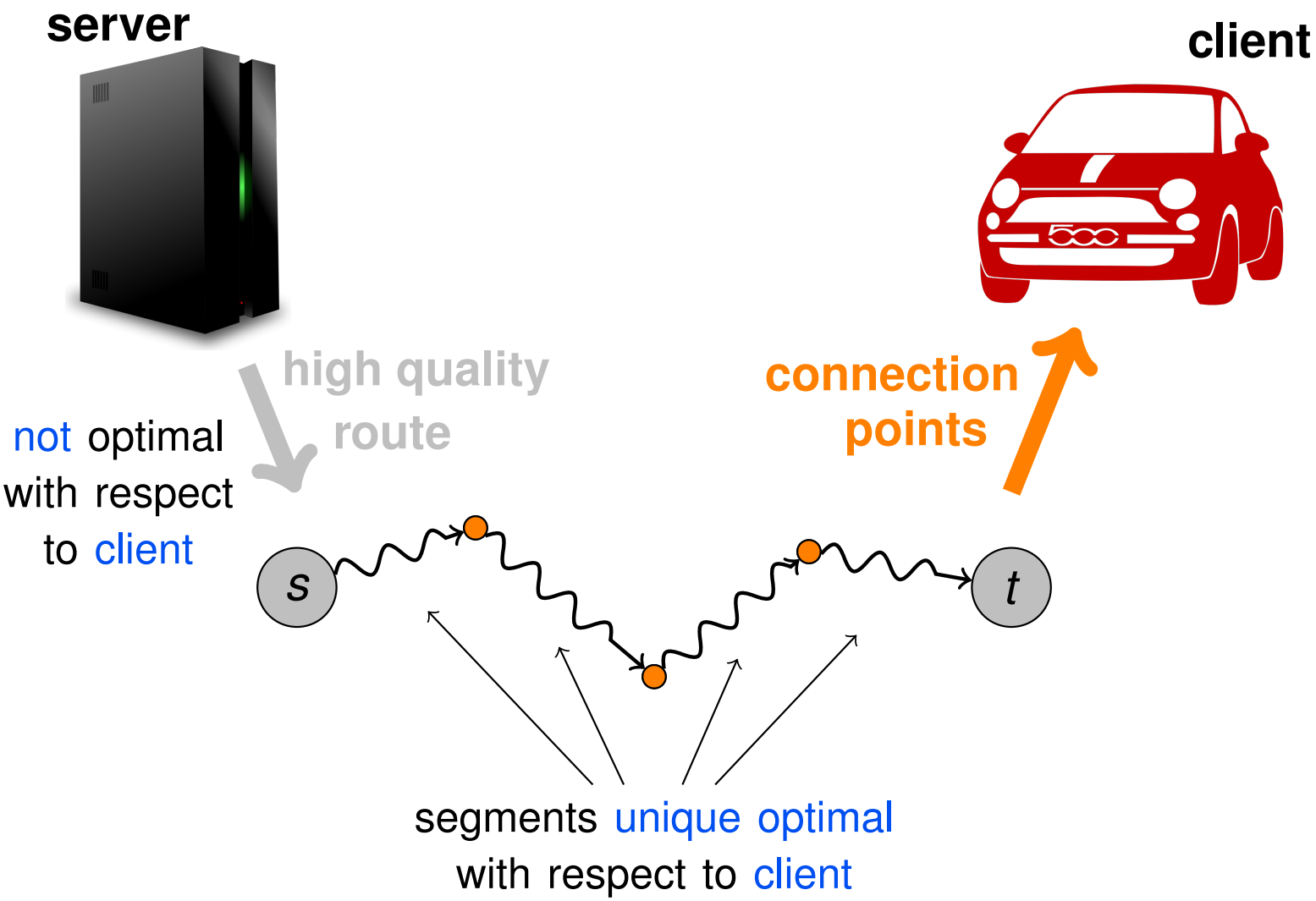
high quality route

not optimal with respect to client

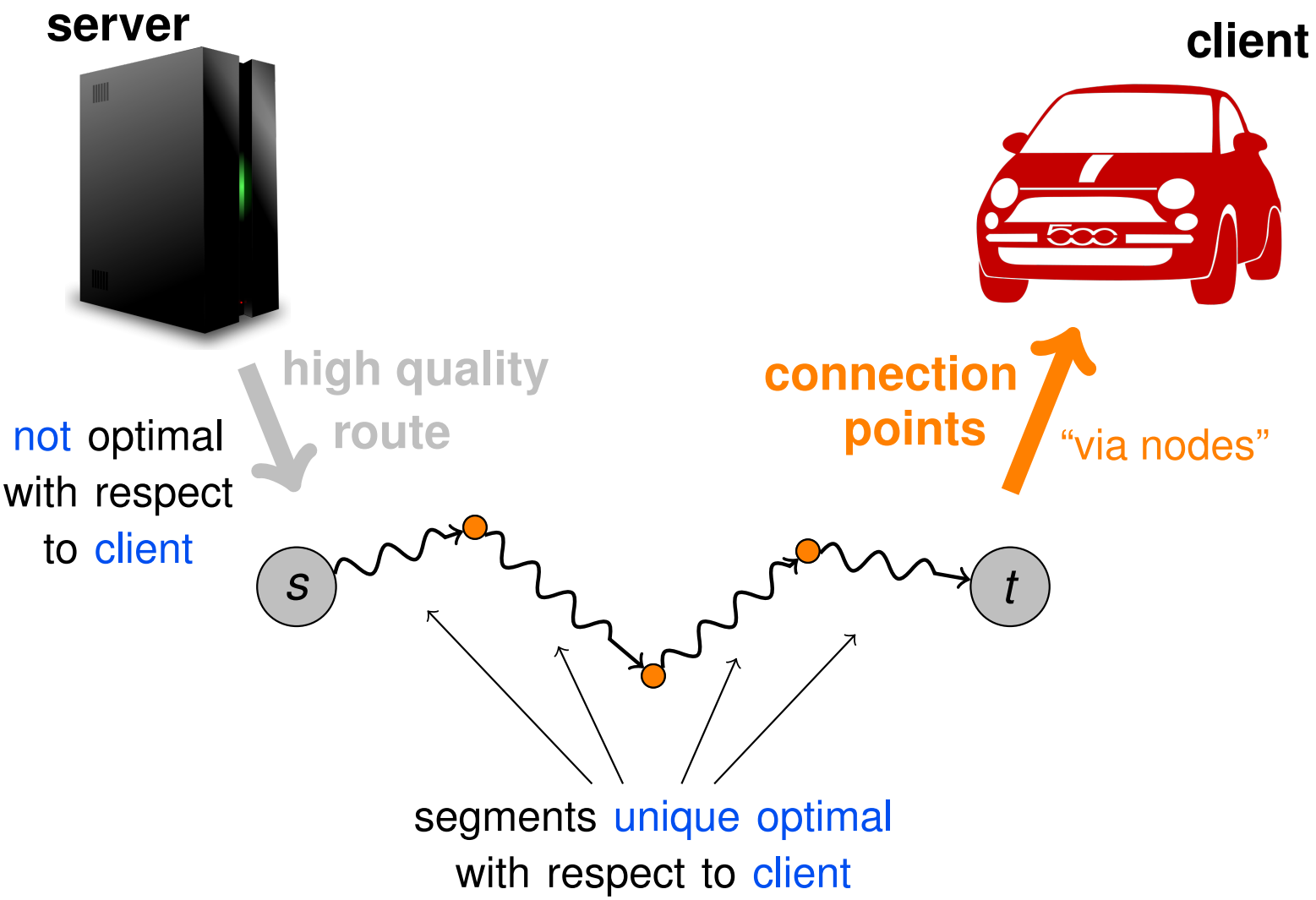


segments unique optimal with respect to client

Basic Idea: Use Via Nodes



Basic Idea: Use Via Nodes



Compression with Via Nodes

Exact Definition

Road network

- Directed graph $G = (V, E)$
- Edge weights client: $c(u, v) \in \mathbb{R}_{\geq 0}$ (simple!)
- Edge weights server: not needed

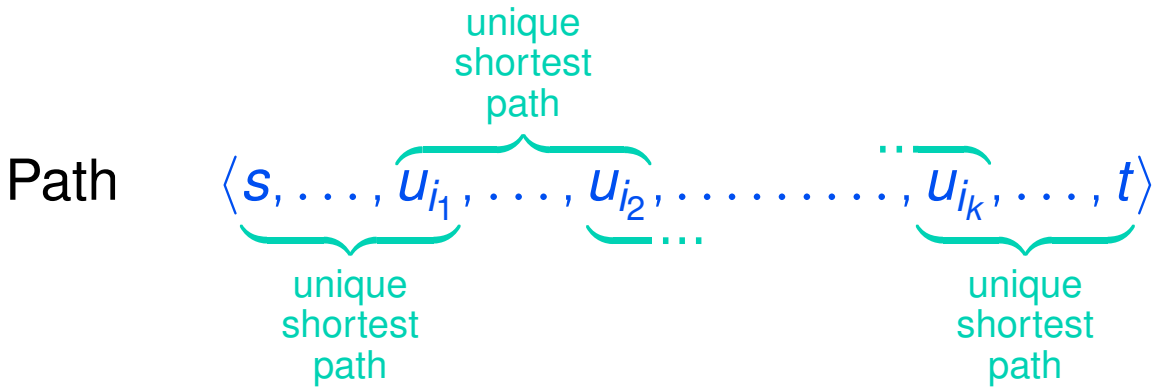
Path $\langle s, \ldots, u_{i_1}, \ldots, u_{i_2}, \ldots\ldots\ldots, u_{i_k}, \ldots, t \rangle$

Compression with Via Nodes

Exact Definition

Road network

- Directed graph $G = (V, E)$
- Edge weights client: $c(u, v) \in \mathbb{R}_{\geq 0}$ (simple!)
- Edge weights server: not needed

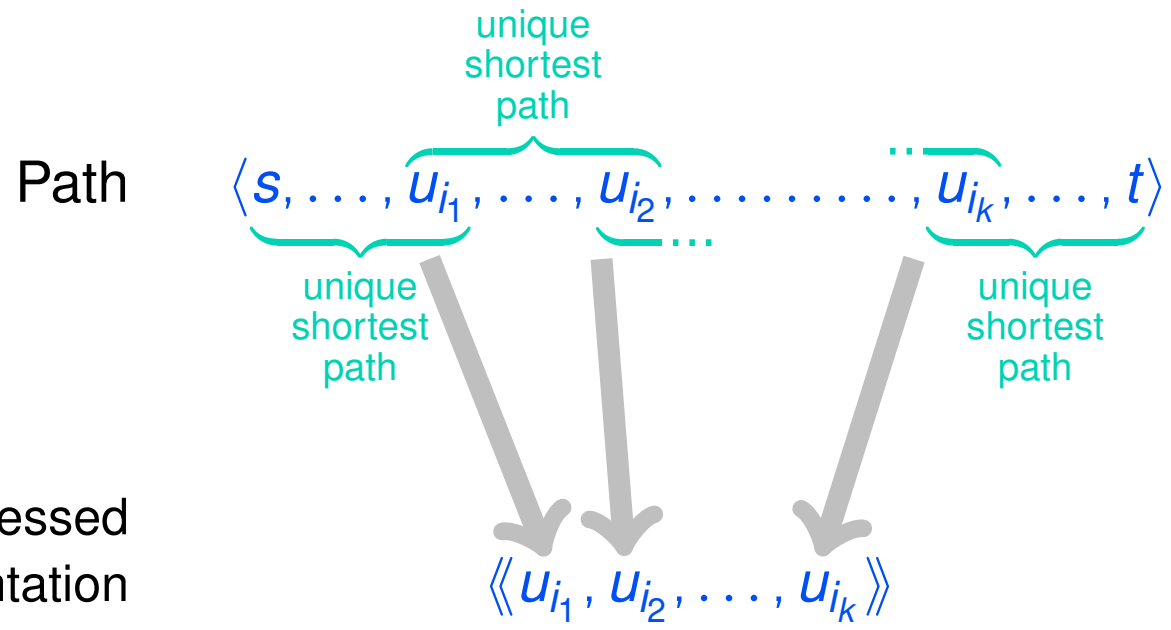


Compression with Via Nodes

Exact Definition

Road network

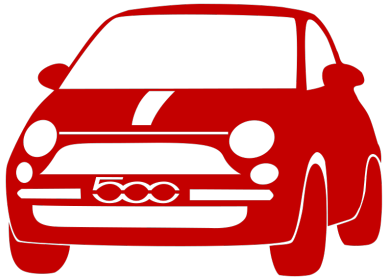
- Directed graph $G = (V, E)$
- Edge weights client: $c(u, v) \in \mathbb{R}_{\geq 0}$ (simple!)
- Edge weights server: not needed



Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness $\Rightarrow$ correctness

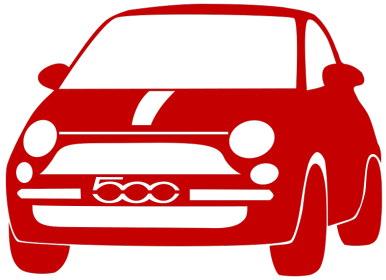


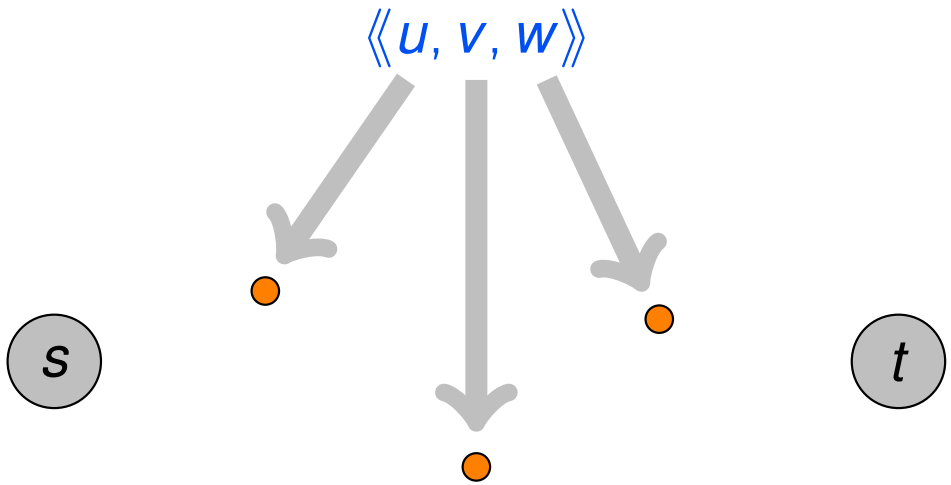
$\langle\langle u, v, w \rangle\rangle$

Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness $\Rightarrow$ correctness

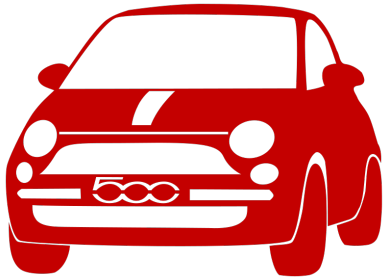


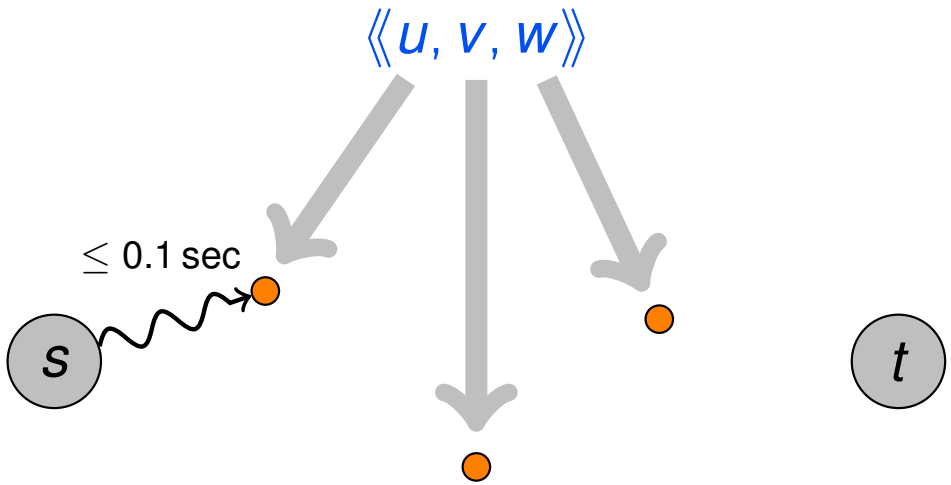


Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness ⇒ correctness

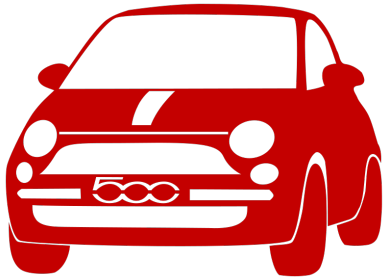


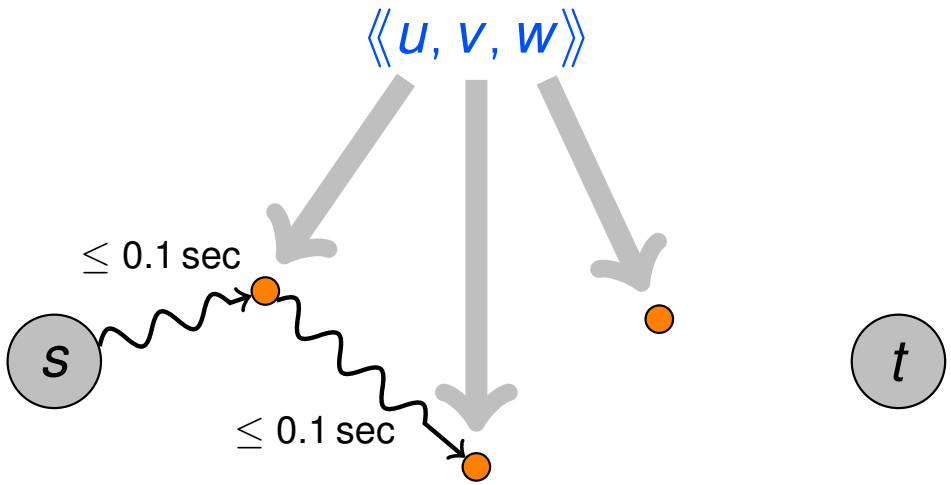


Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness $\Rightarrow$ correctness

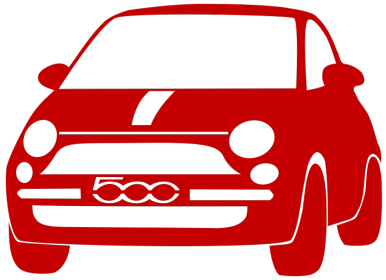


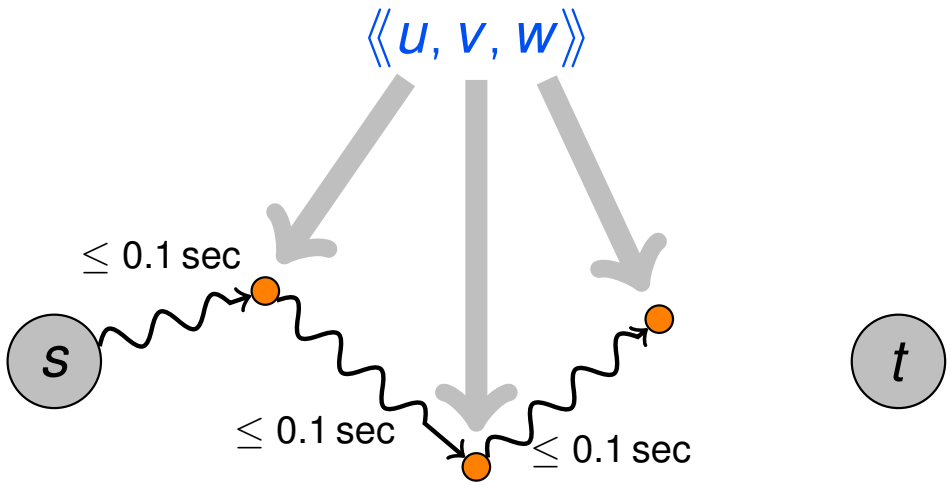


Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness $\Rightarrow$ correctness

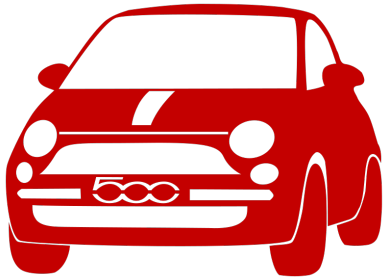


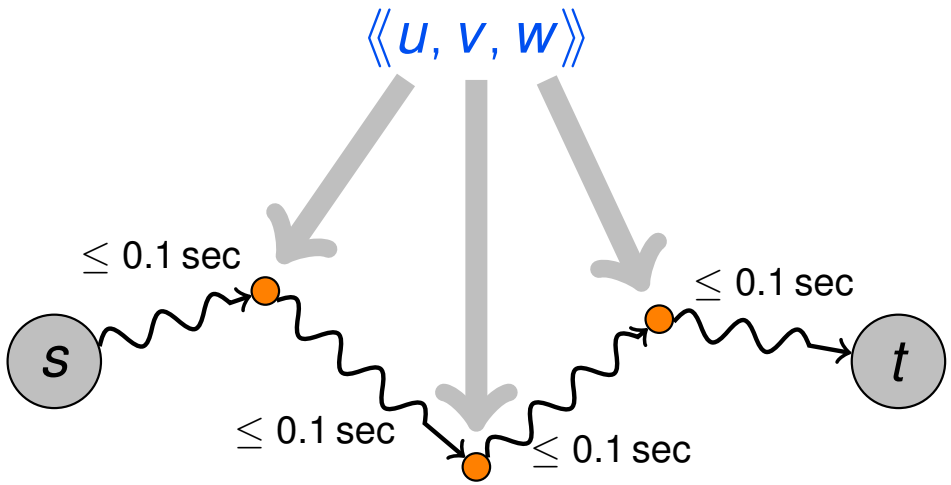


Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness $\Rightarrow$ correctness

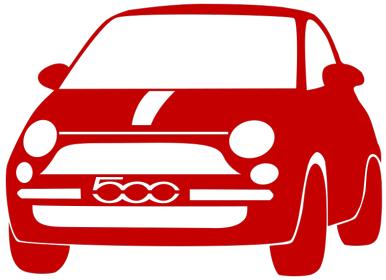


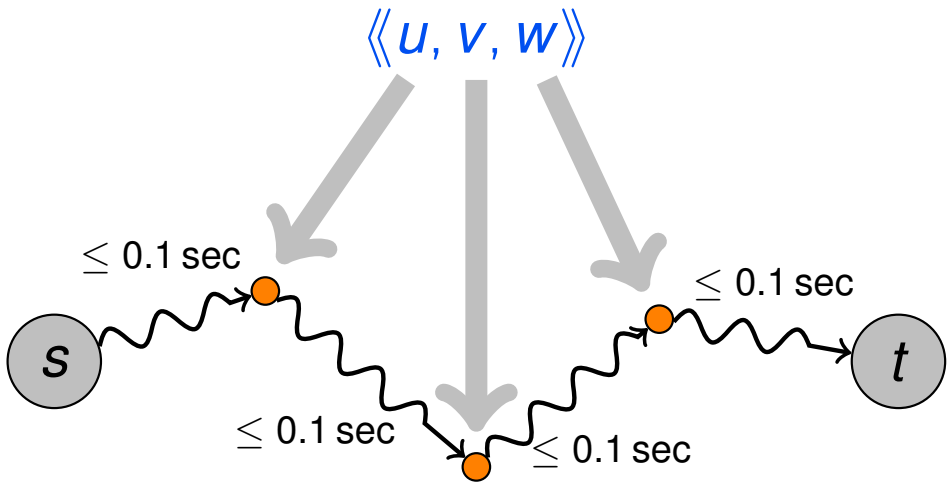


Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness ⇒ correctness



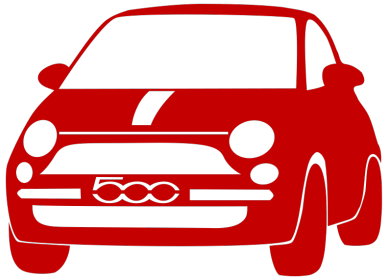


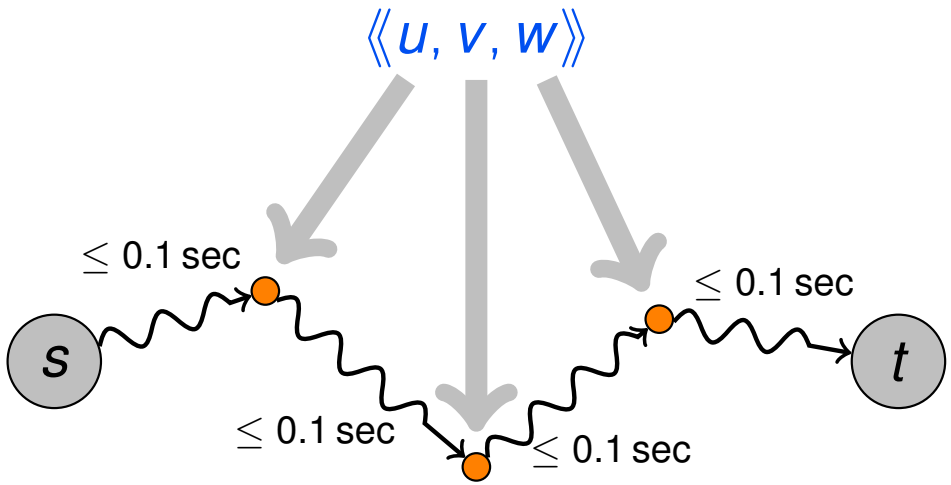
⇒ **Driver experiences no delay!**

Compression with Via Nodes

Method for Decompression

- Compute shortest paths between via nodes
- Mobile CH do this in 0.1 sec. [ESA 08]
- Uniqueness ⇒ correctness





⇒ Driver experiences no delay!

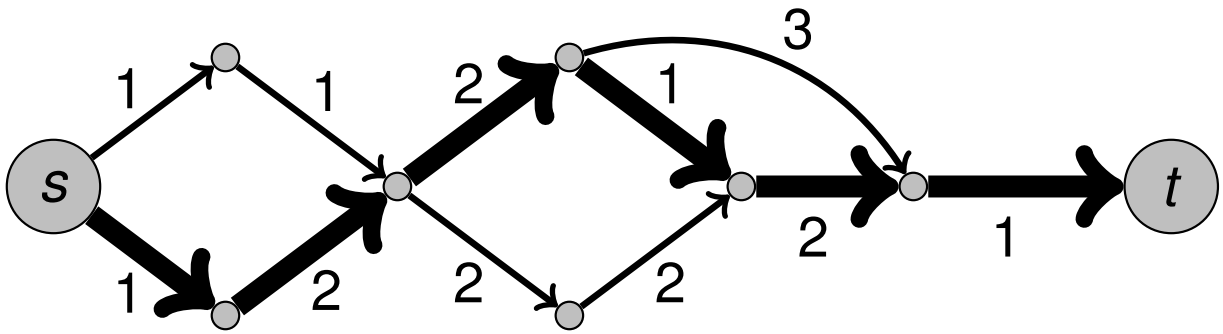
⇒ Convenient to use.

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



$Q = \langle\langle \qquad\qquad \rangle\rangle$

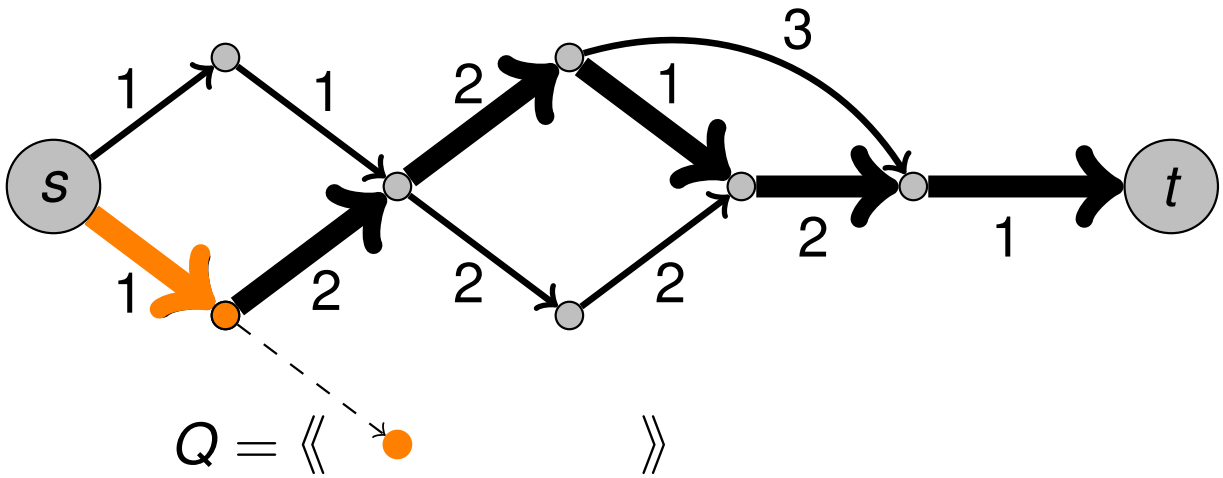
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



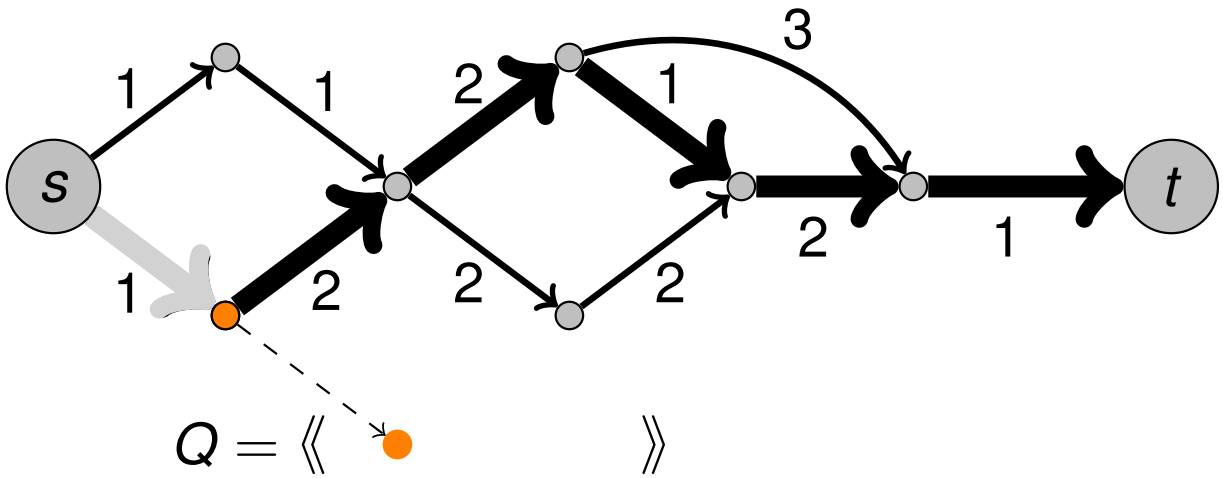
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



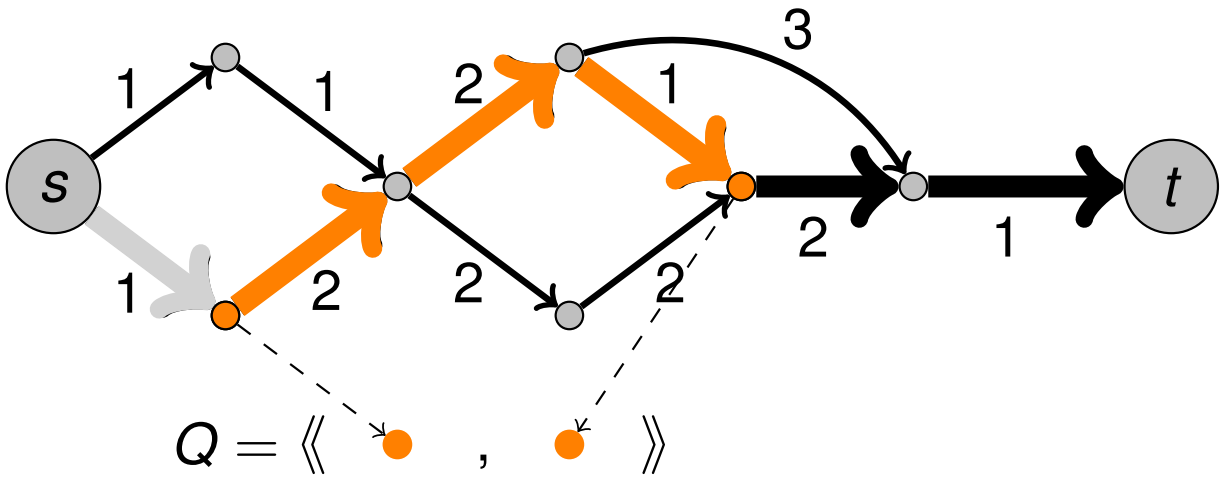
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



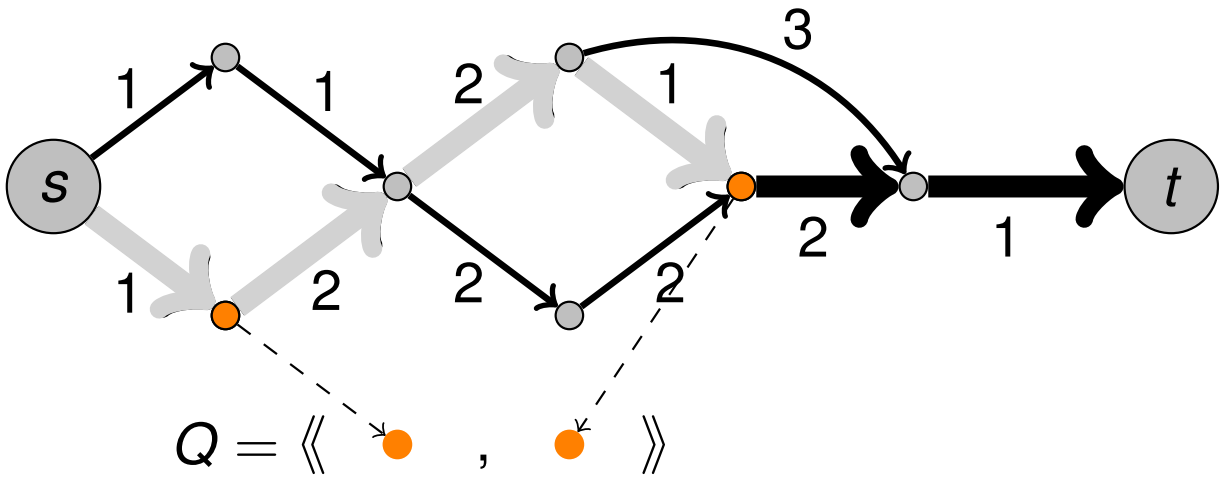
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



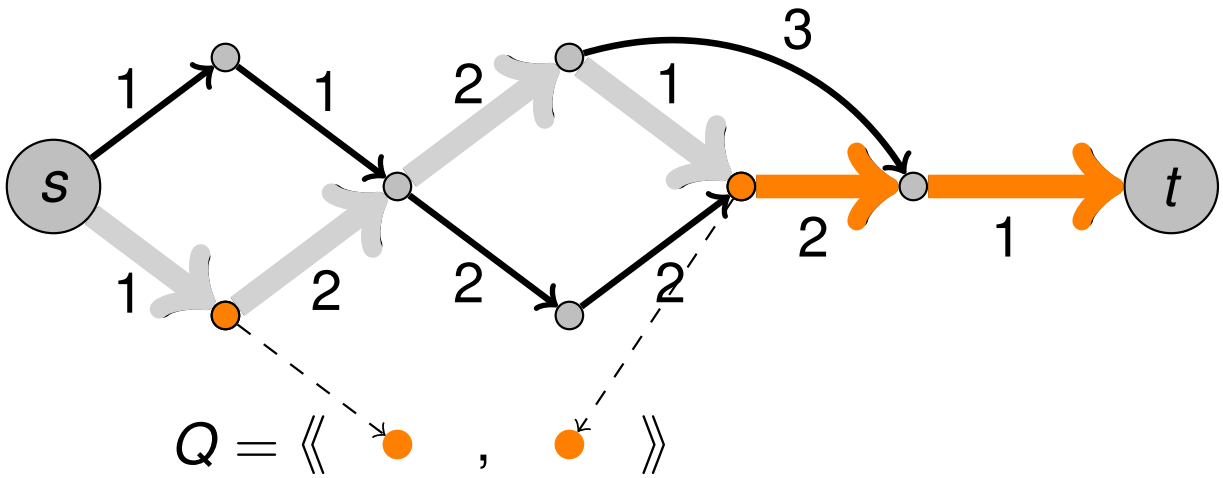
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



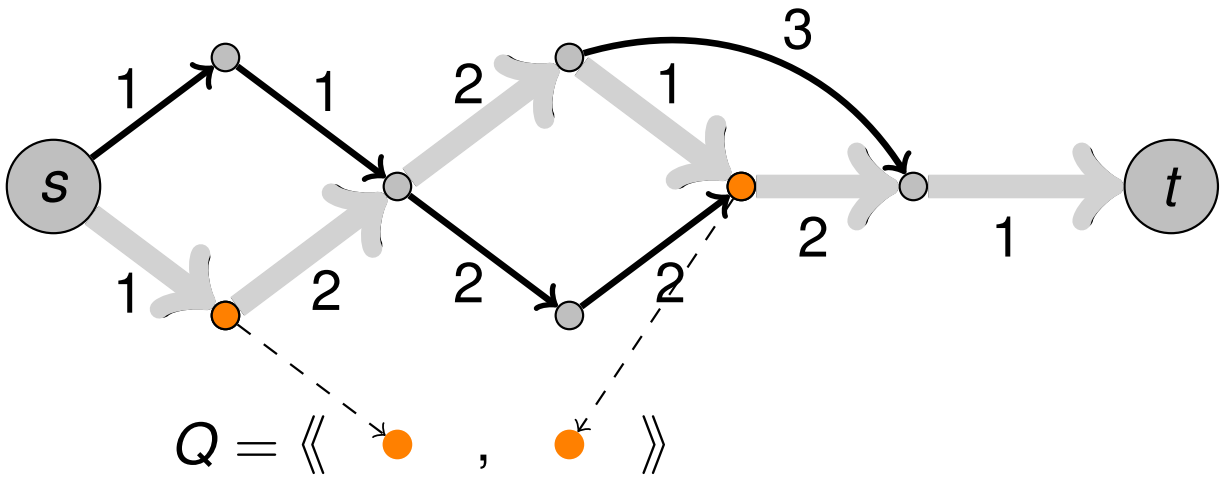
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



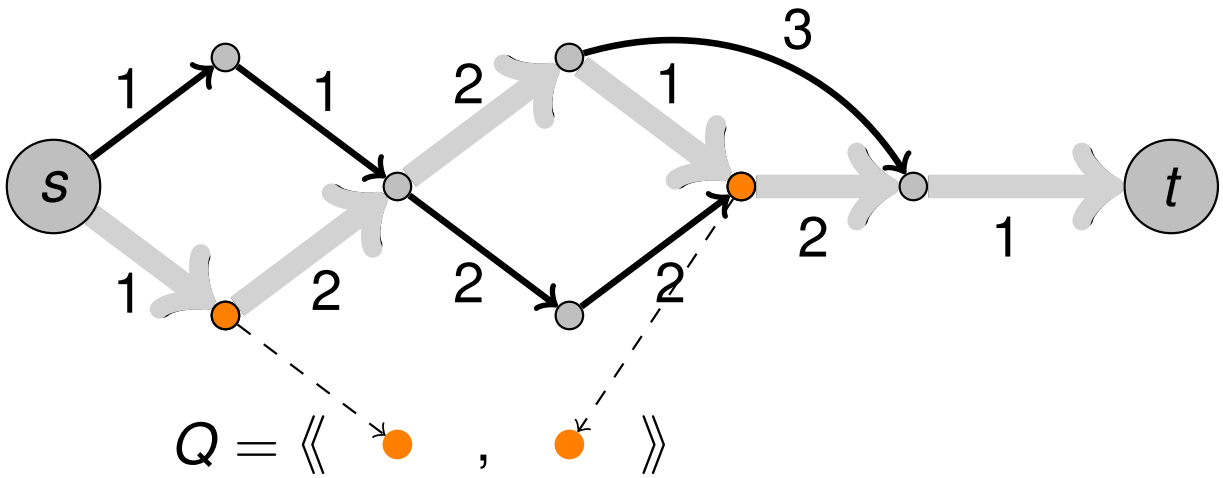
$\Rightarrow$ Finds minimal number of via nodes

Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

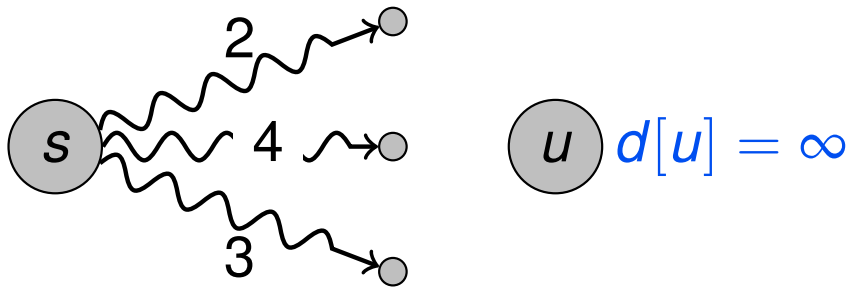
- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal unique shortest prefix from P
- **each** appending its last node to Q



$\Rightarrow$ Finds minimal number of via nodes

Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



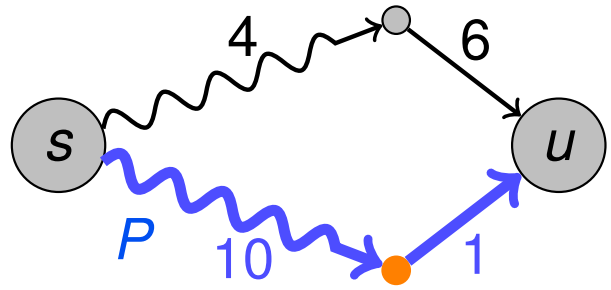
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

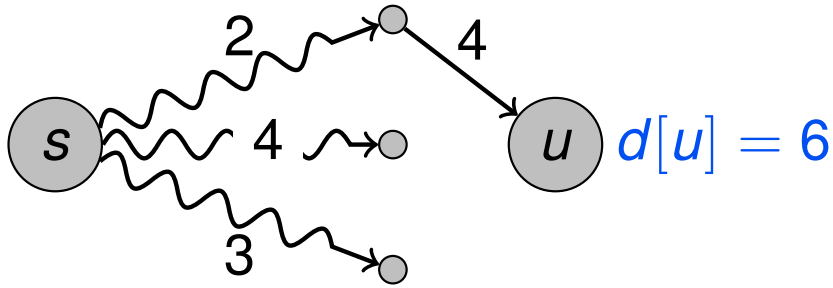
- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!



Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



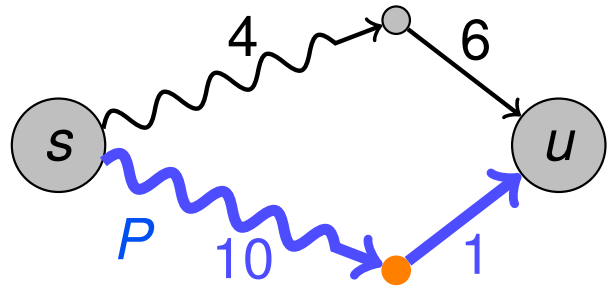
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

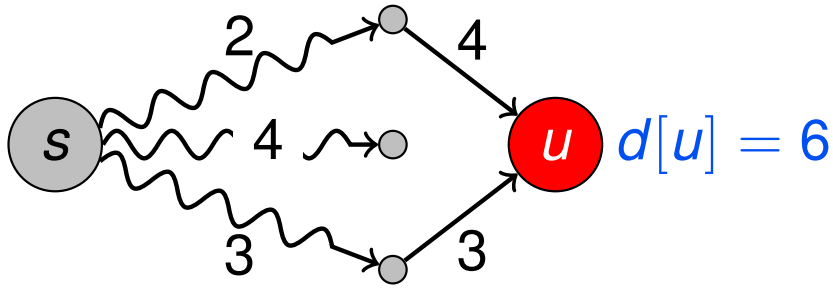
- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!



Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



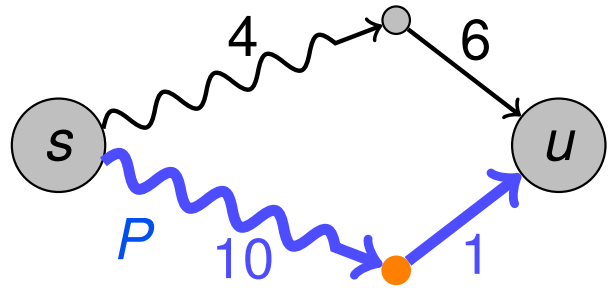
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

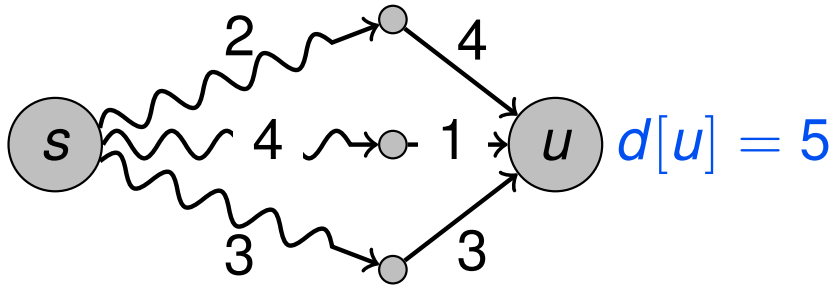
- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!



Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



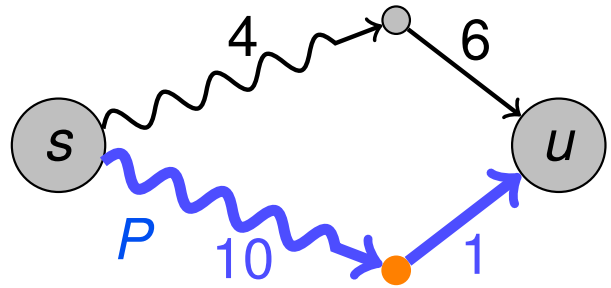
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

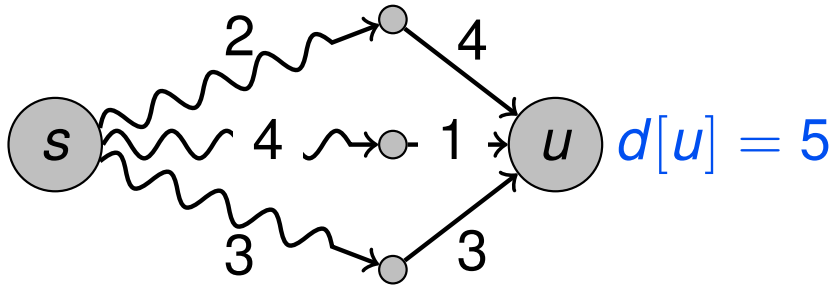
- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!



Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



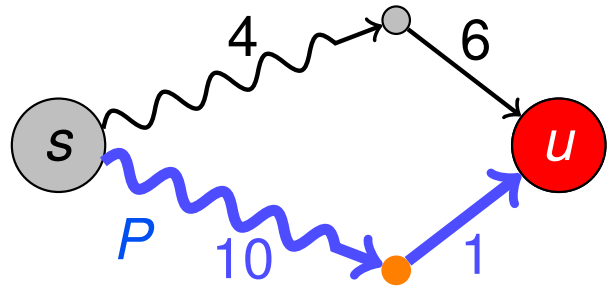
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

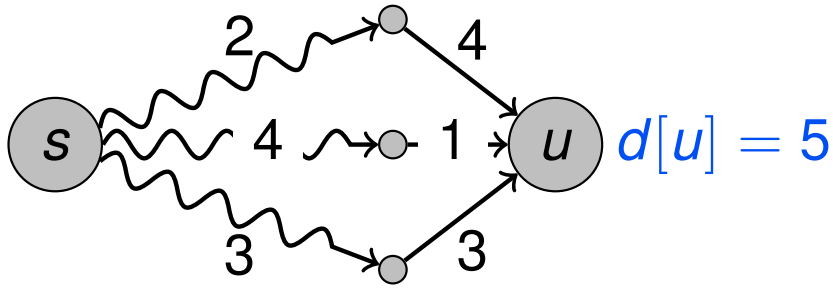
- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!



Find Maximal Unique Shortest Prefix

Modified Dijkstra – simple but slow



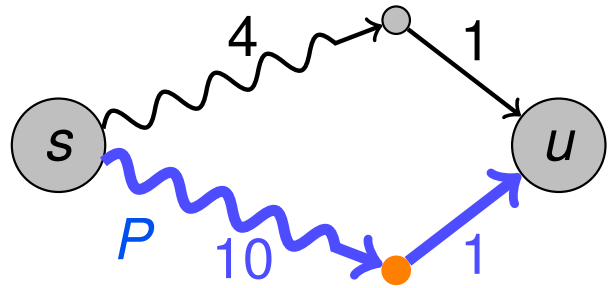
Relaxing an edge:

- Mark node when non-uniquely reached
- Unmark node when reached by shorter path

Settling a node:

- settling a marked node of P or...
- ...prefix not a shortest path

$\Rightarrow$ Unique maximal prefix found!

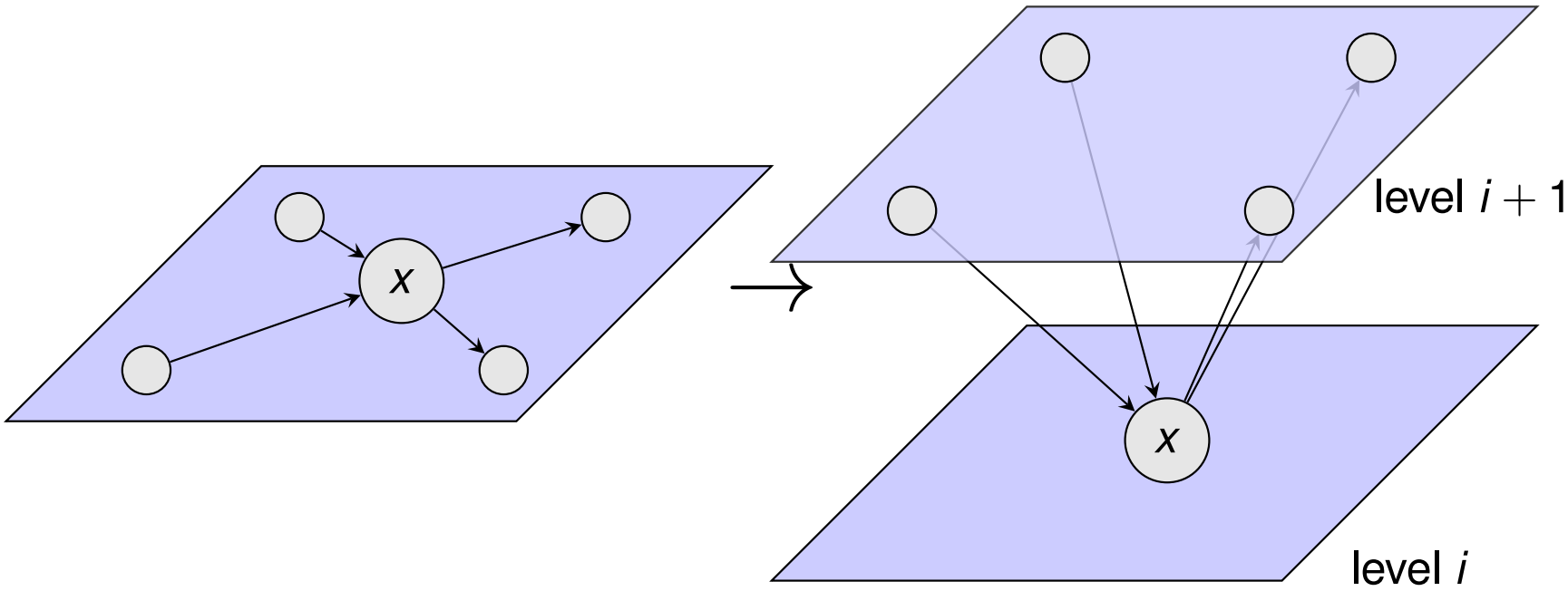


Contraction Hierarchies (CH)

[Geisberger et al. 08]

Construct a hierarchy in a preprocessing step:

- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts

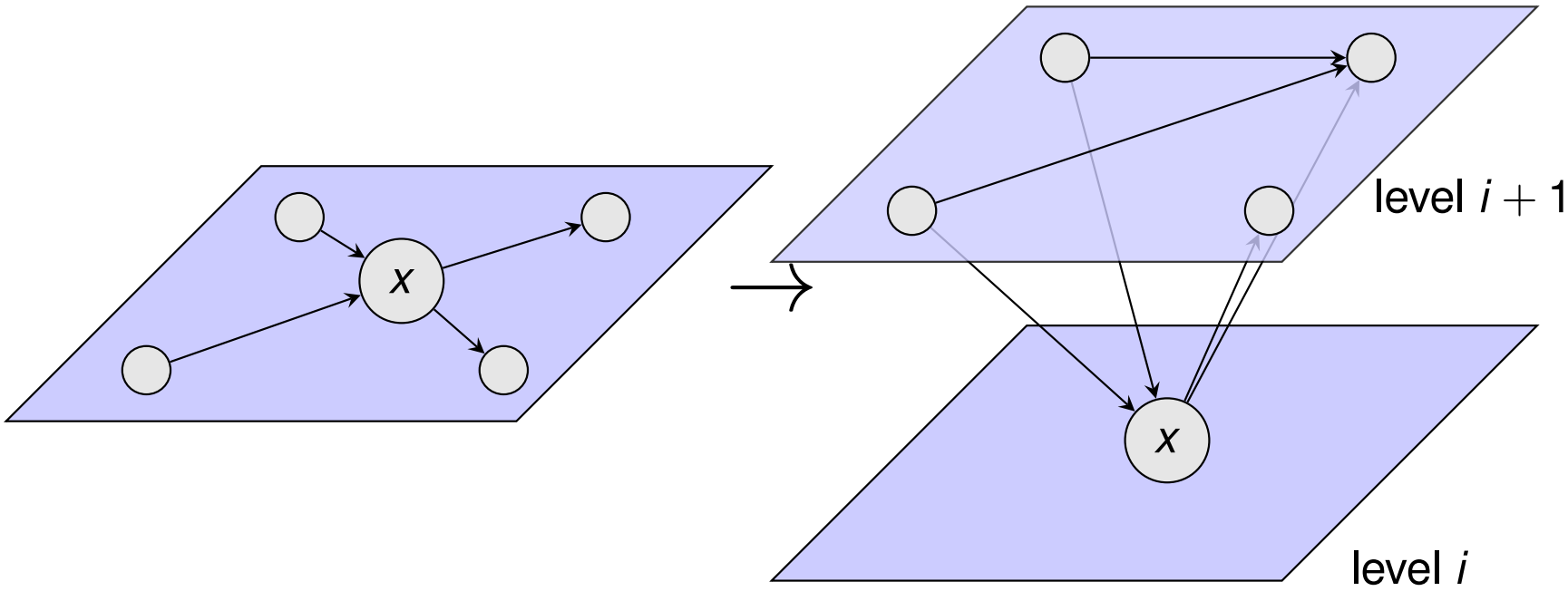


Contraction Hierarchies (CH)

[Geisberger et al. 08]

Construct a hierarchy in a preprocessing step:

- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts

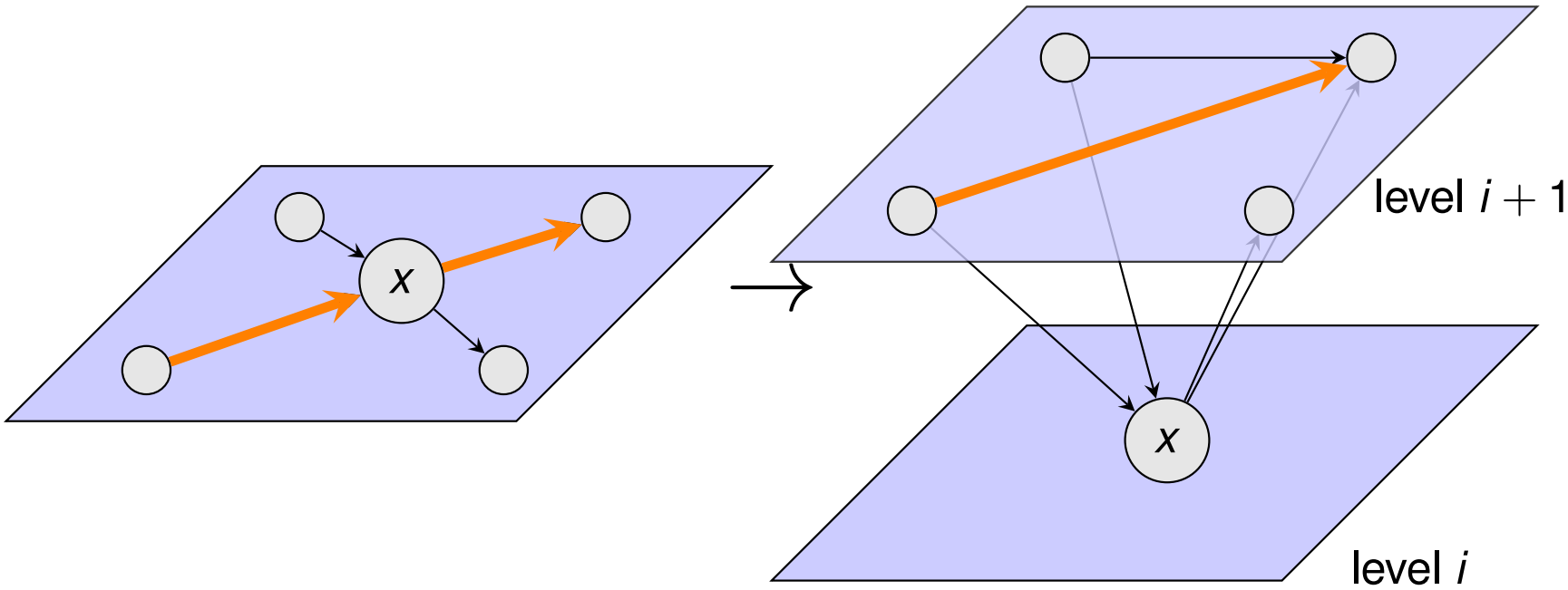


Contraction Hierarchies (CH)

[Geisberger et al. 08]

Construct a hierarchy in a preprocessing step:

- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts

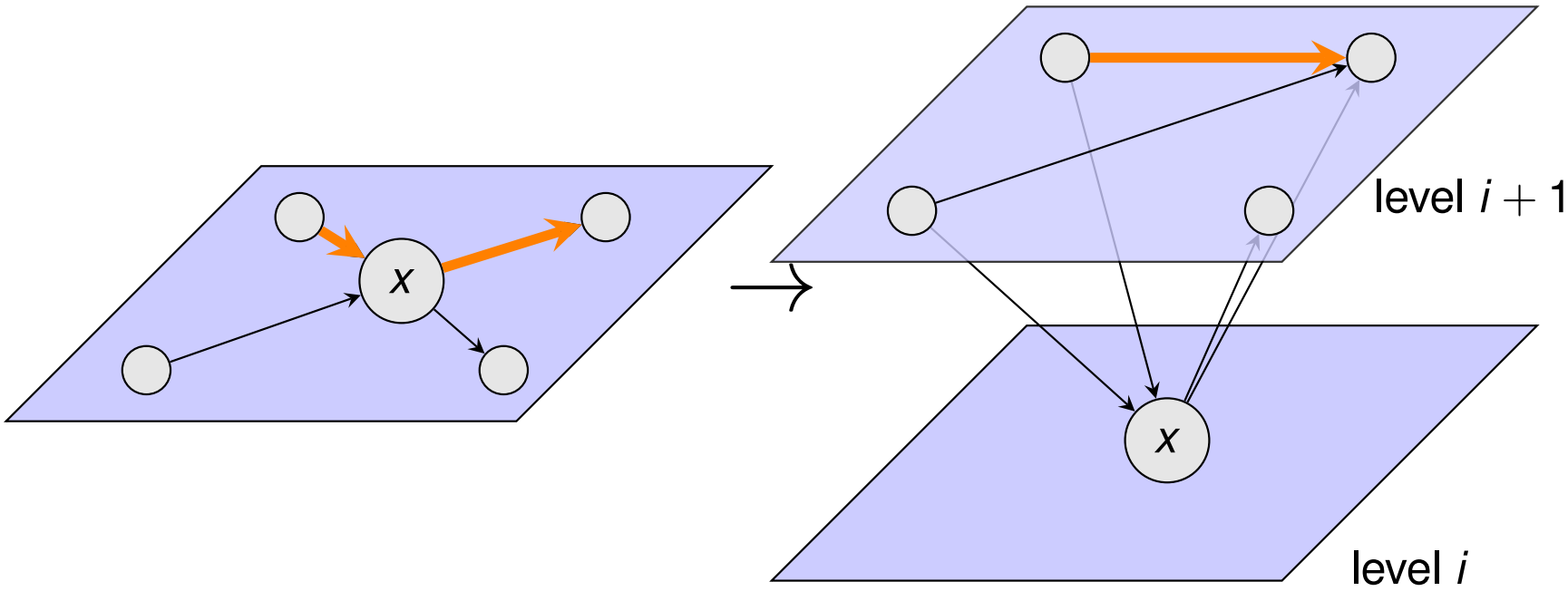


Contraction Hierarchies (CH)

[Geisberger et al. 08]

Construct a hierarchy in a preprocessing step:

- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts

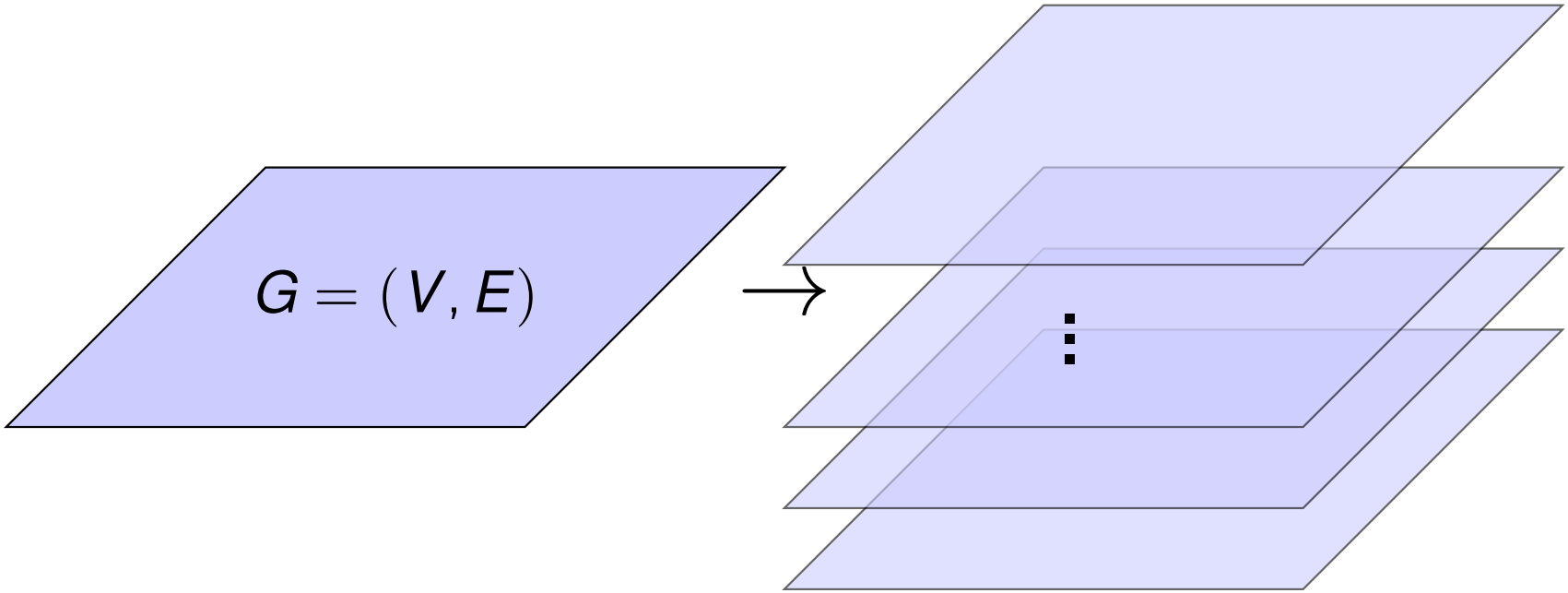


Contraction Hierarchies (CH)

[Geisberger et al. 08]

Construct a hierarchy in a preprocessing step:

- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts

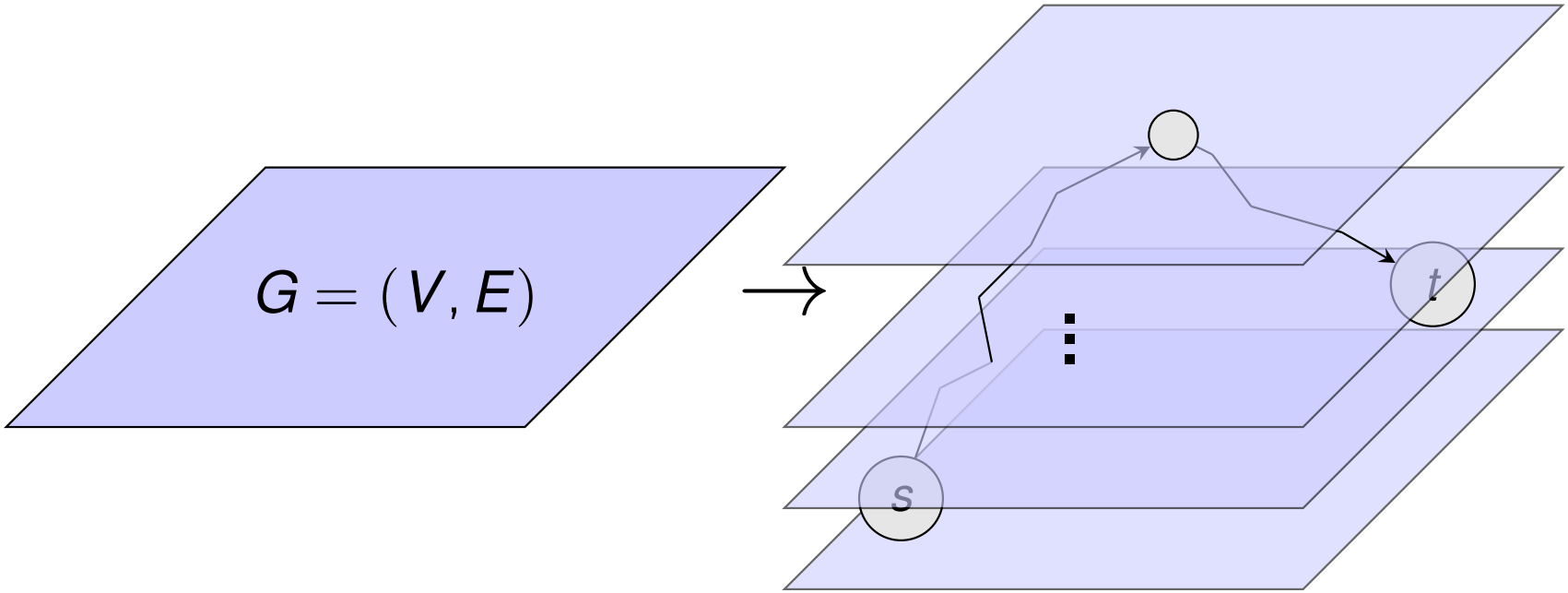


Contraction Hierarchies (CH)

[Geisberger et al. 08]

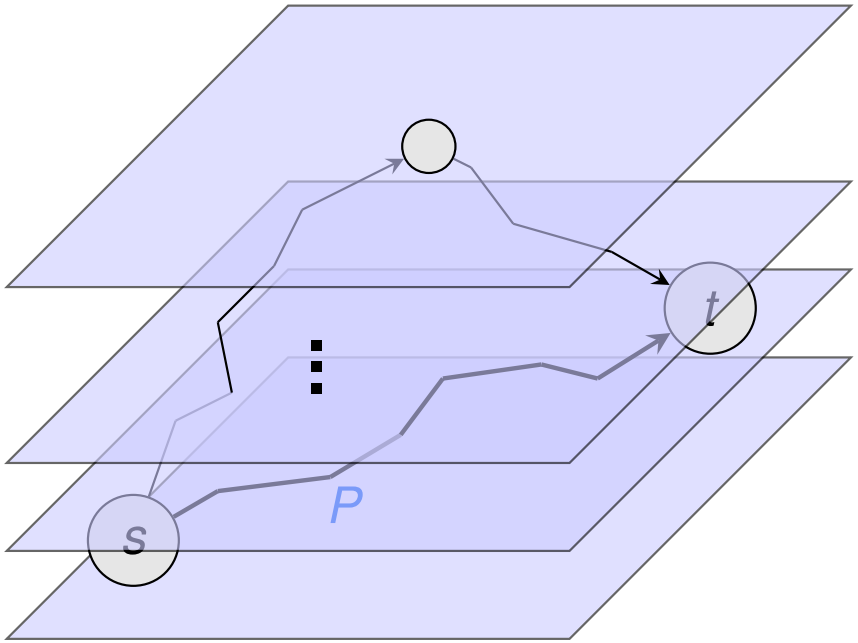
Construct a hierarchy in a preprocessing step:

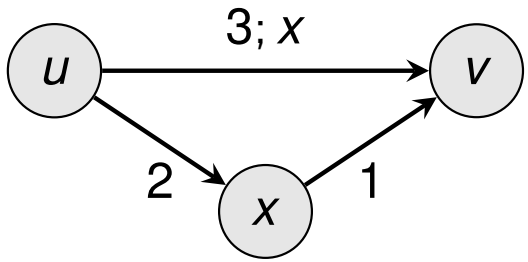
- Order nodes by importance
- Obtain next level by contracting next node
- Preserve shortest paths by inserting shortcuts



$\Rightarrow$ There is always a shortest up-down-path from s to t.

Representing Paths with CH Uniquely

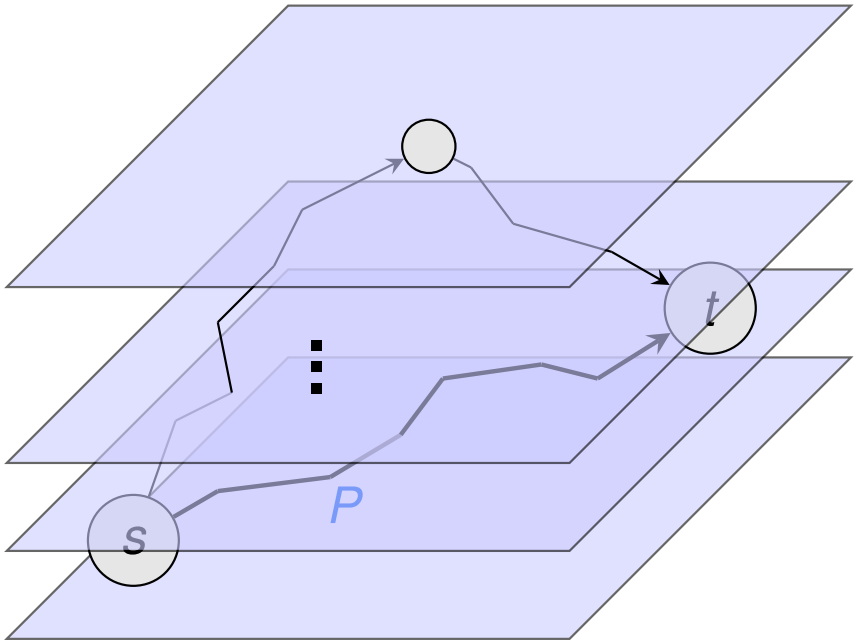


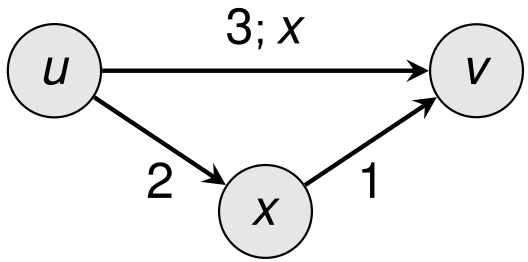


- Shortcuts can be expanded recursively
- Up-down-paths contain several shortcuts
- Expand these shortcuts completely

  $\Rightarrow$ An up-down-path represents an original path

Representing Paths with CH Uniquely

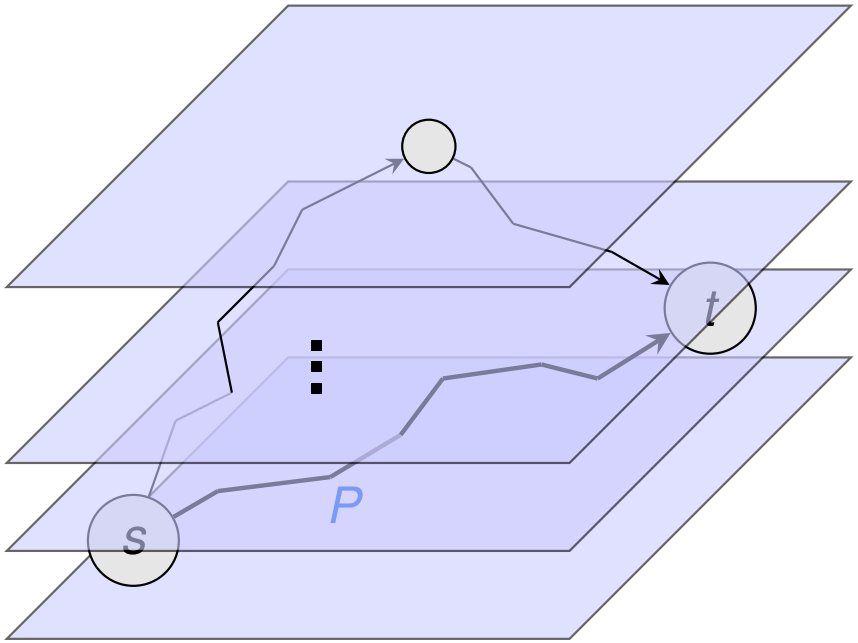


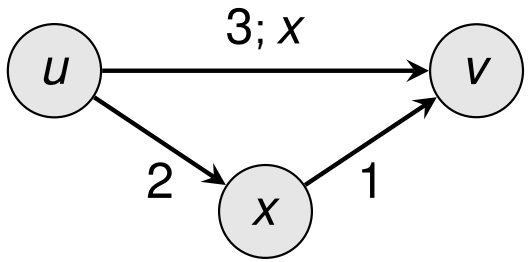


- Shortcuts can be expanded recursively
- Up-down-paths contain several shortcuts
- Expand these shortcuts completely

  ⇒ An up-down-path represents an original path

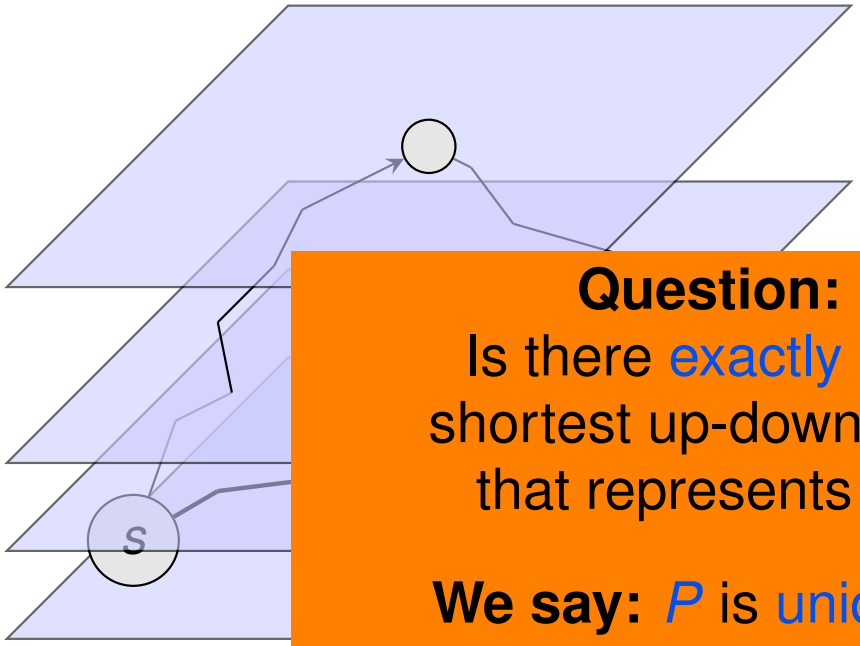
Representing Paths with CH Uniquely





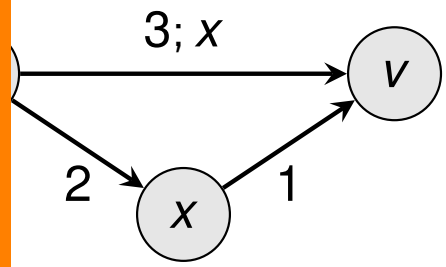
- Shortcuts can be expanded recursively
- Up-down-paths contain several shortcuts
- Expand these shortcuts completely
  ⇒ An up-down-path represents an original path

Representing Paths with CH Uniquely



Question:
Is there exactly one shortest up-down path that represents P?

We say: P is uniquely CH-representable

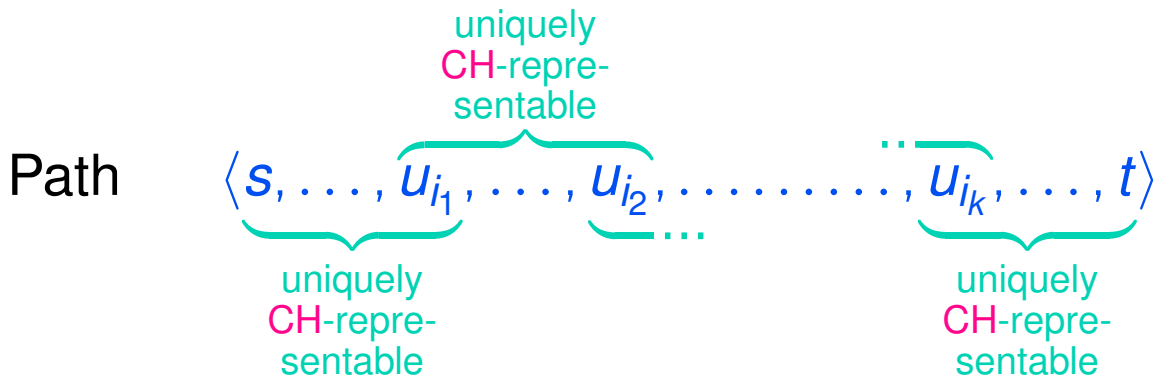


- Shortcuts can be expanded recursively
- Up-down-paths contain several shortcuts
- Expand these shortcuts completely
  $\Rightarrow$ An up-down-path represents an original path

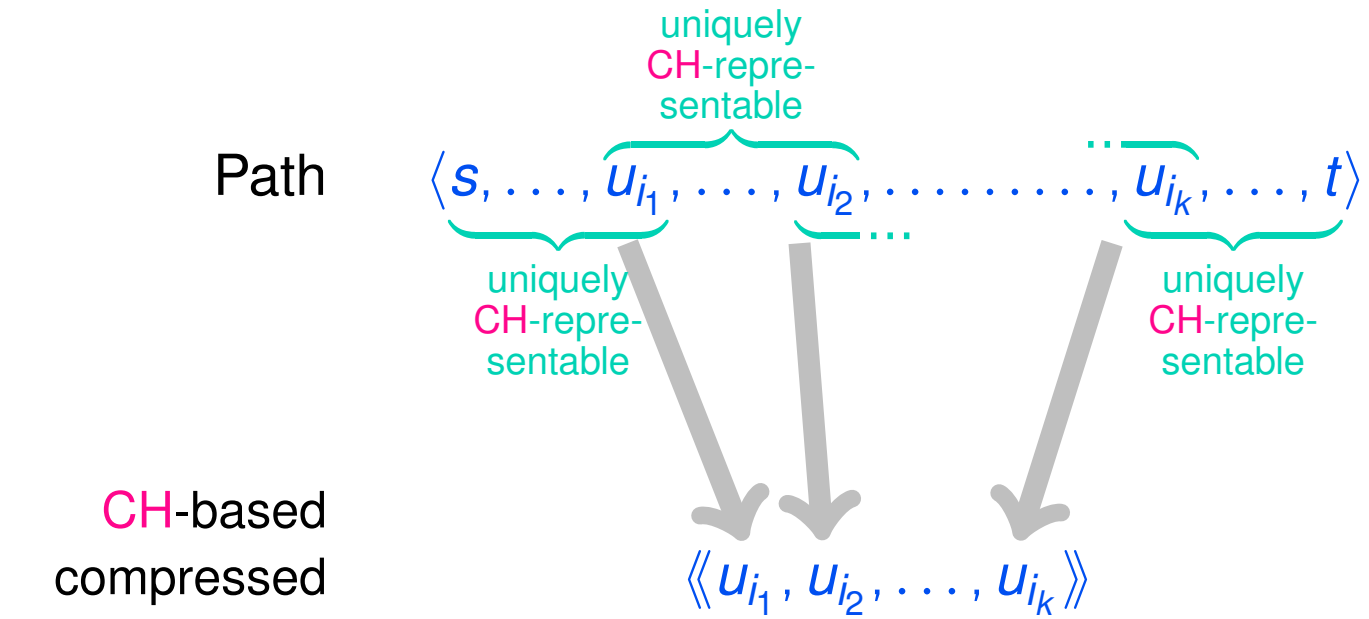
CH-Based Representation of a path with Via Nodes

Path $\langle s, \ldots, u_{i_1}, \ldots, u_{i_2}, \ldots\ldots\ldots, u_{i_k}, \ldots, t \rangle$

CH-Based Representation of a path with Via Nodes



CH-Based Representation of a path with Via Nodes



Compression with Via Nodes

Frame Algorithm for Compression

Compress path P

- $Q := \langle\langle\rangle\rangle$
- **Repeatedly**
- remove the maximal uniquely CH-representable prefix from P
- **each** appending its last node to Q

Find Uniquely CH-Representable Prefix

Inspired by Binary Search

Find uniquely CH-representable prefix of path $\langle u_1, \ldots, u_n \rangle$

- $(\ell, m, r) := (1, n, n)$
- **while** $\ell + 1 < r$ **do**
- **if** $\langle u_1, \ldots, u_m \rangle$ is uniquely CH-representable **then** $\ell := m$
- **else** $r := m$
- $m := \lfloor \ell + 1 + r \rfloor$
- **od**
- **return** $\langle u_1, \ldots, u_\ell \rangle$

Find Uniquely CH-Representable Prefix

Inspired by Binary Search

Find uniquely CH-representable prefix of path $\langle u_1, \ldots, u_n \rangle$

- $(\ell, m, r) := (1, n, n)$
- **while** $\ell + 1 < r$ **do**
- **if** $\langle u_1, \ldots, u_m \rangle$ is uniquely CH-representable **then** $\ell := m$
- **else** $r := m$
- $m := \lfloor \ell + 1 + r \rfloor$
- **od**
- **return** $\langle u_1, \ldots, u_\ell \rangle$

Find Uniquely CH-Representable Prefix

Inspired by Binary Search

Find uniquely CH-representable prefix of path $\langle u_1, \ldots, u_n \rangle$

- $(\ell, m, r) := (1, n, n)$
- **while** $\ell + 1 < r$ **do**
- **if** $\langle u_1, \ldots, u_m \rangle$ is uniquely CH-representable **then** $\ell := m$
- **else** $r := m$
- $m := \lfloor \ell + 1 + r \rfloor$
- **od**
- **return** $\langle u_1, \ldots, u_\ell \rangle$

$\Rightarrow$ Checks $O(\log n)$ times whether uniquely CH-representable.

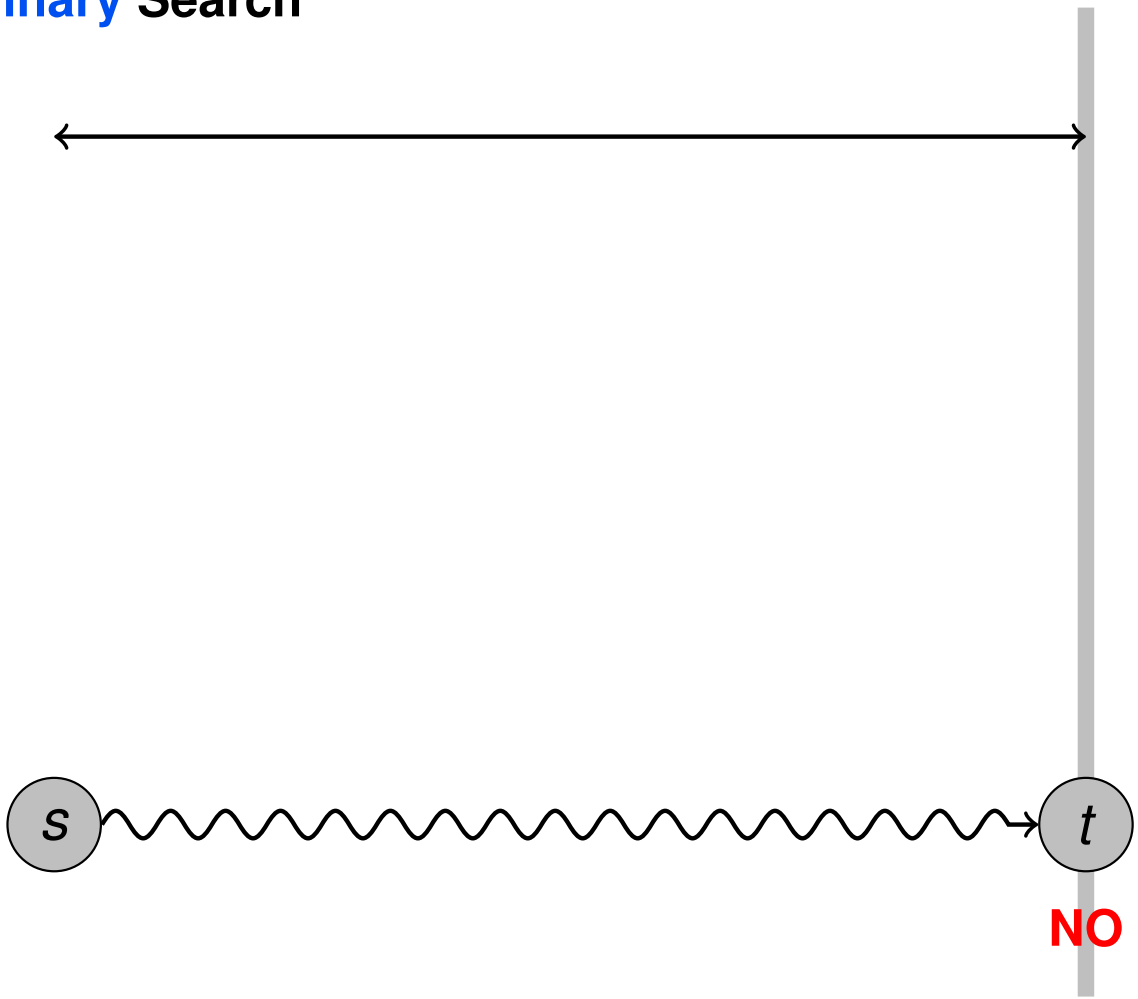
How to Find a Uniquely CH-Representable Prefix

Inspired by Binary Search

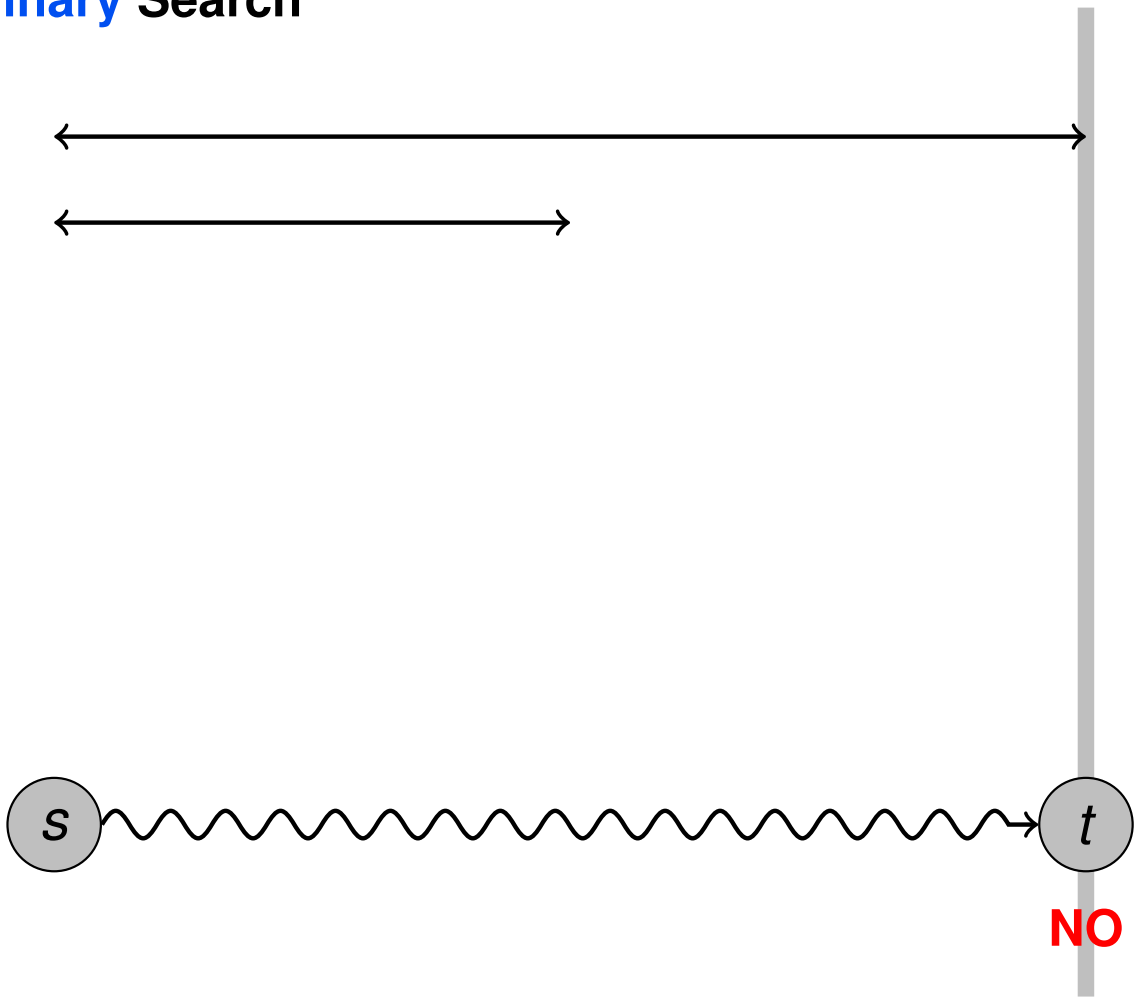
How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search

How to Find a Uniquely CH-Representable Prefix

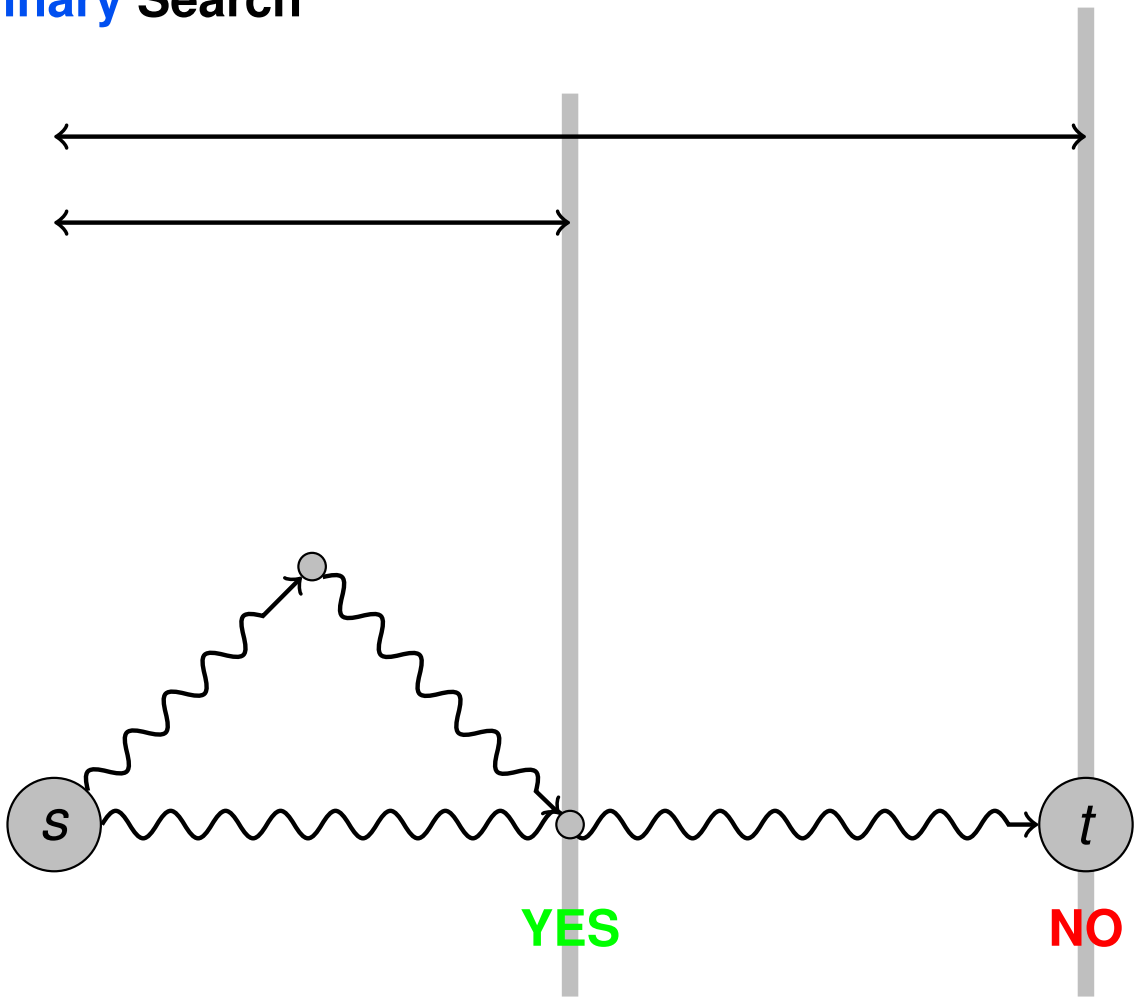
Inspired by Binary Search



How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search

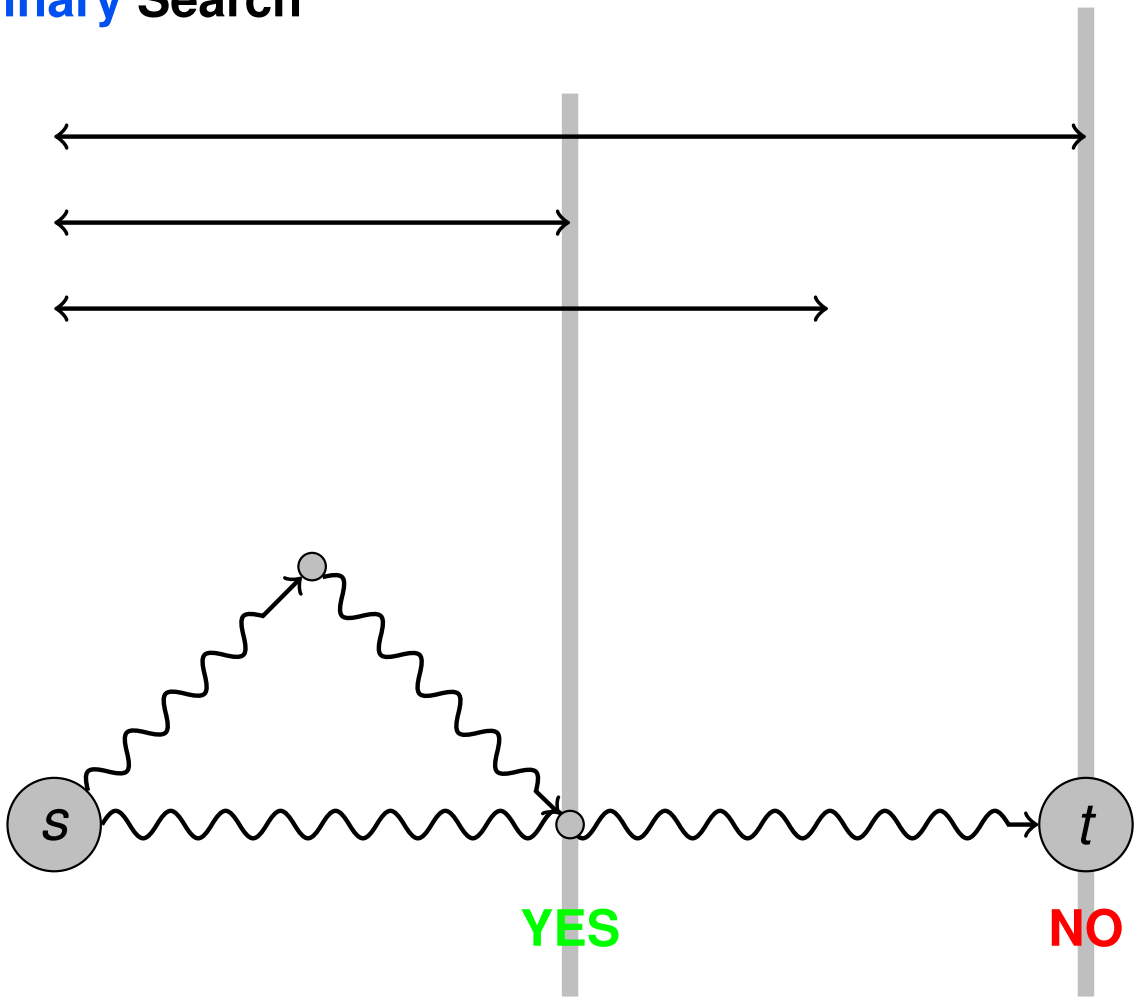


How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search

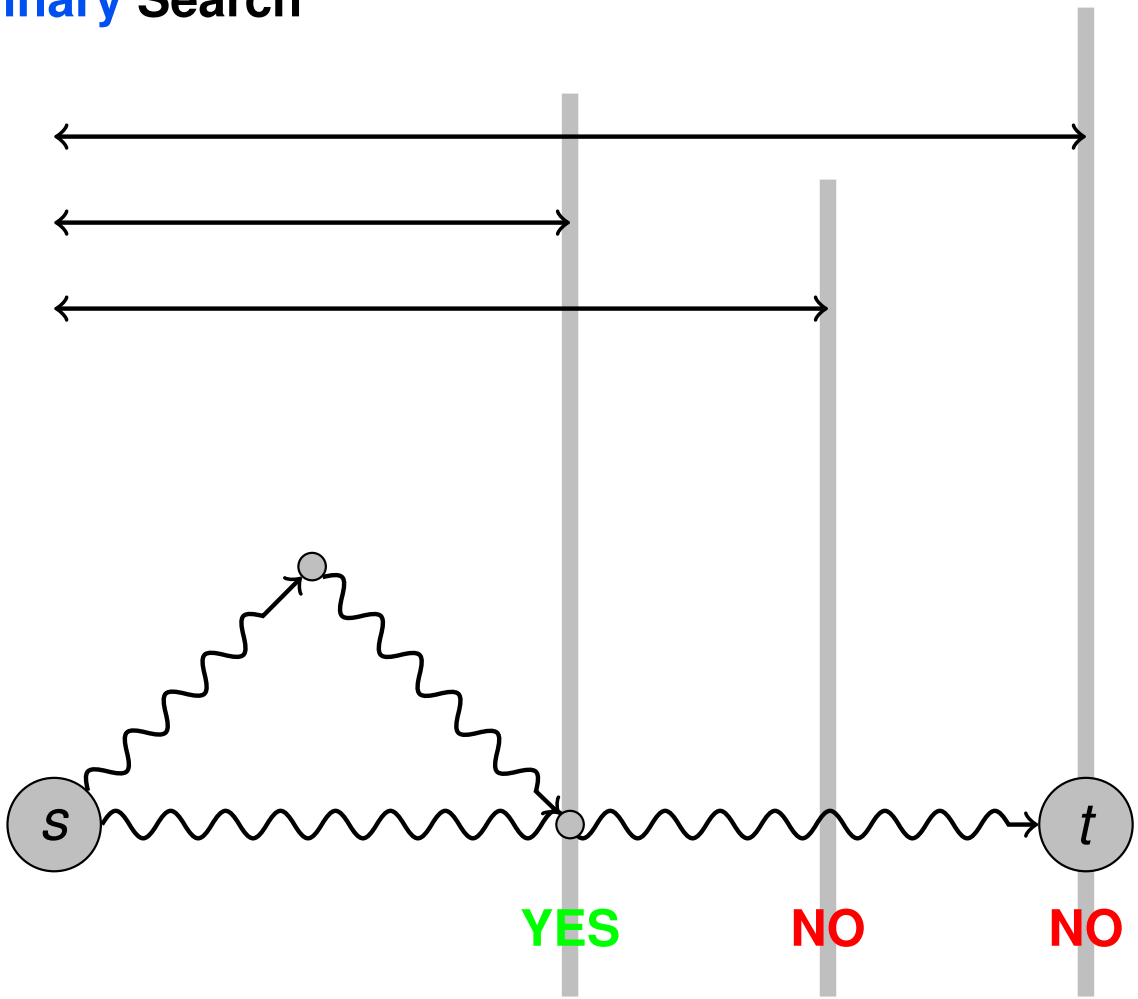


How to Find a Uniquely CH-Representable Prefix

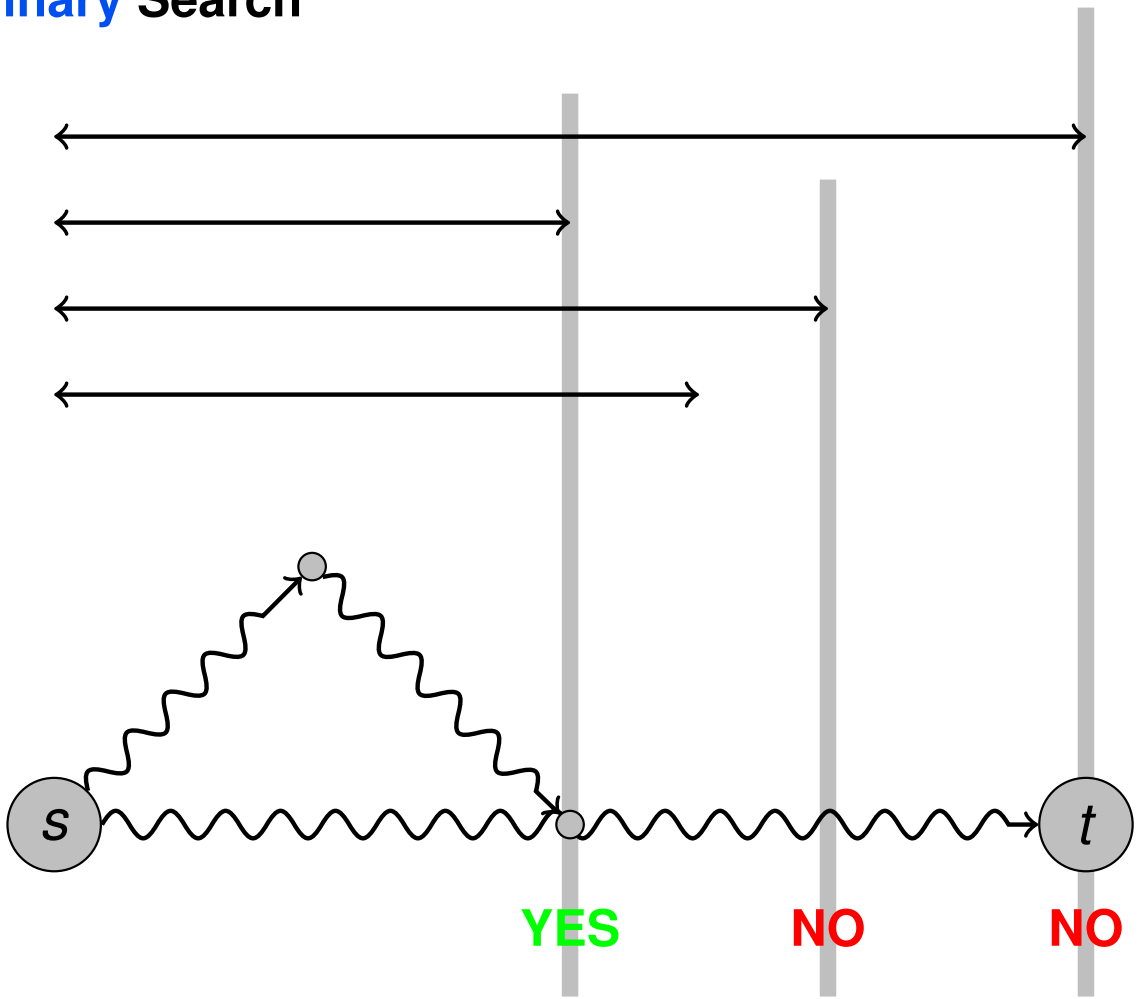
Inspired by Binary Search



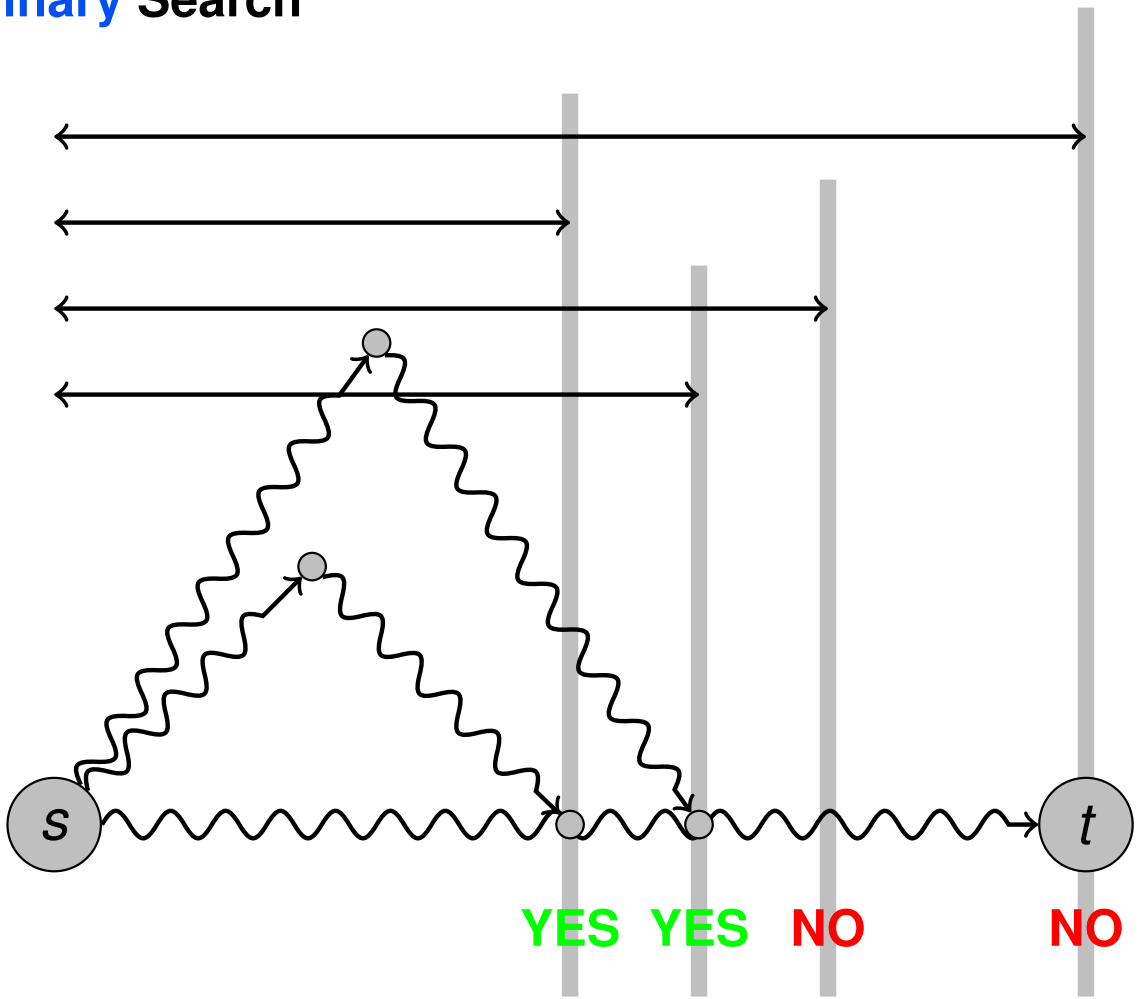
How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search



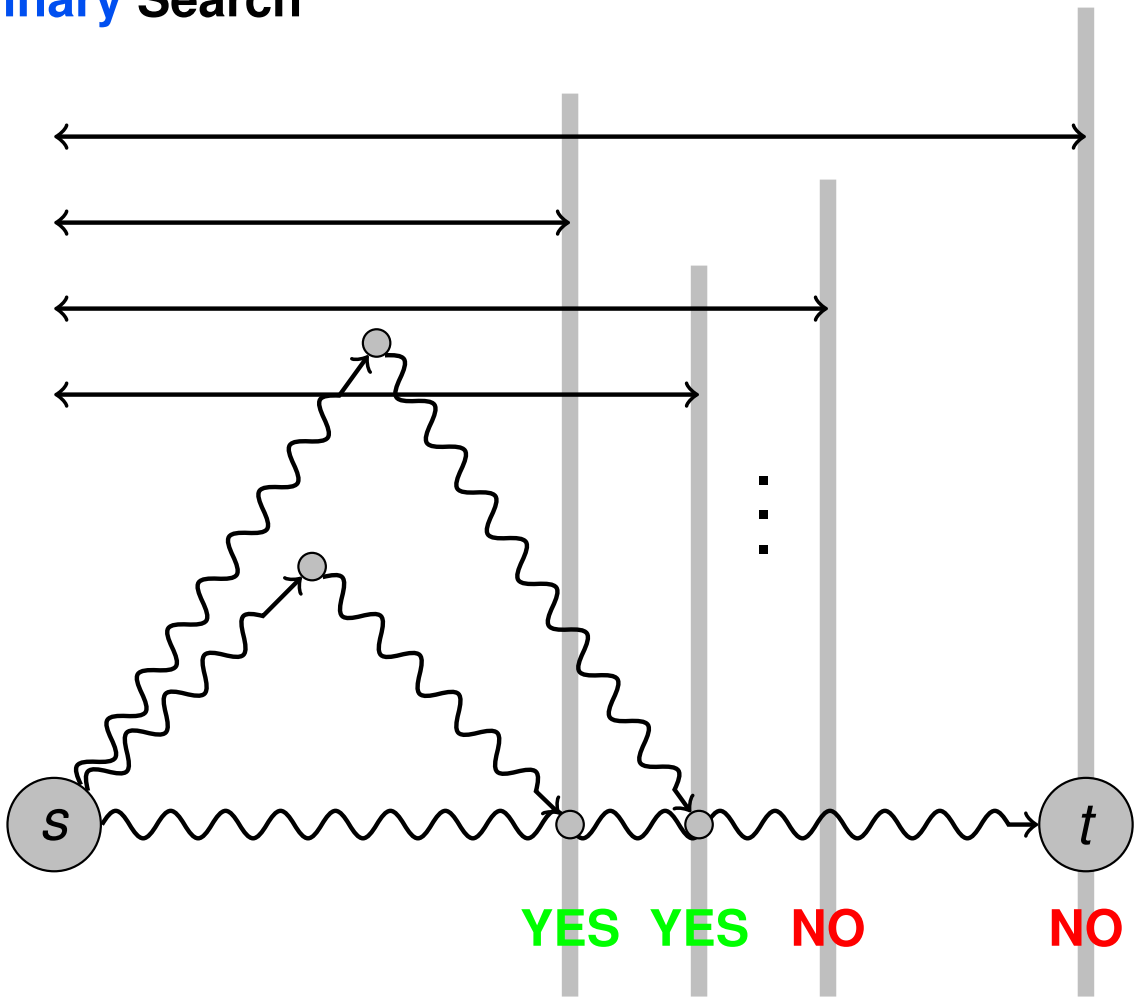
How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search



How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search



How to Find a
Uniquely CH-Representable Prefix
Inspired by Binary Search



Find Uniquely CH-Representable Prefix

Inspired by Binary Search

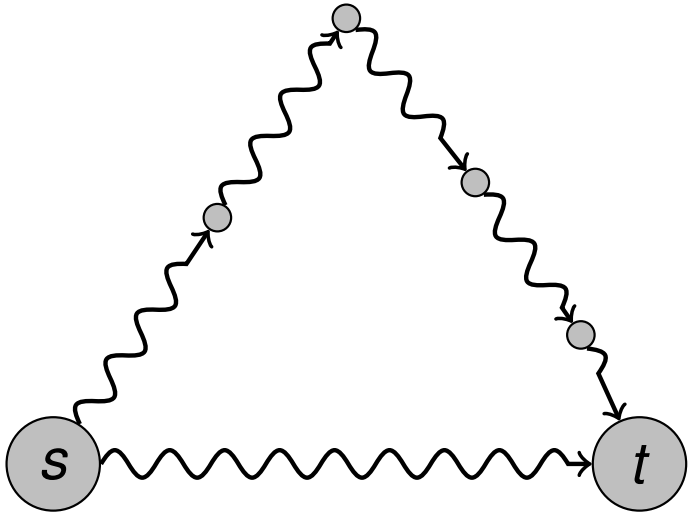
Find uniquely CH-representable prefix of path $\langle u_1, \ldots, u_n \rangle$

- $(\ell, m, r) := (1, n, n)$ **Theorem:**

  Found prefix contains unique shortest prefix or more.
- **while** $\ell + 1 < r$ **do**
- **if** $\langle u_1, \ldots, u_m \rangle$ is uniquely CH-representable **then** $\ell := m$
- **else** $r := m$
- $m := \lfloor \ell + 1 + r \rfloor$
- **od**
- **return** $\langle u_1, \ldots, u_\ell \rangle$

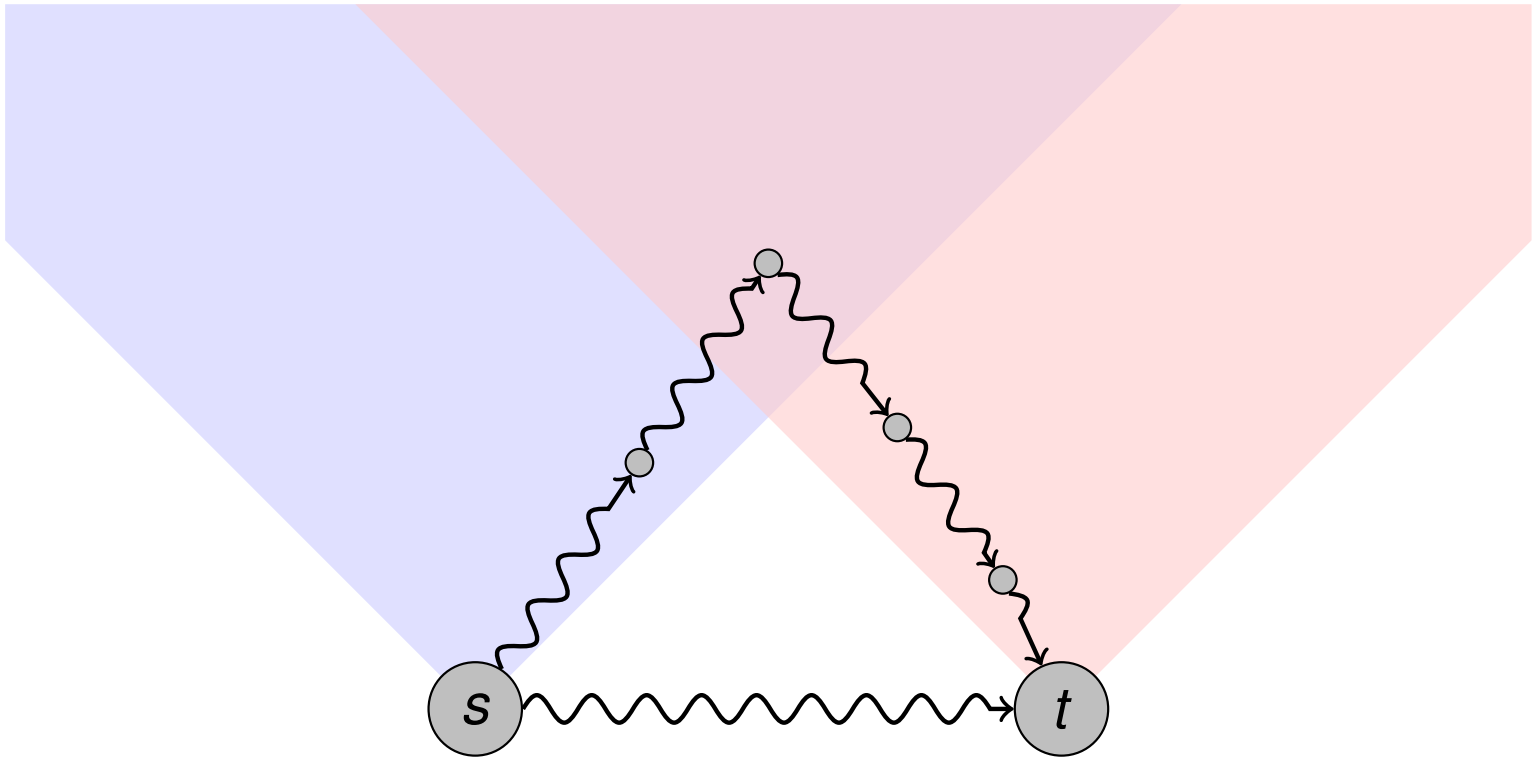
$\Rightarrow$ Checks $O(\log n)$ times whether uniquely CH-representable.

How to Check Whether Path is Uniquely CH-Representable



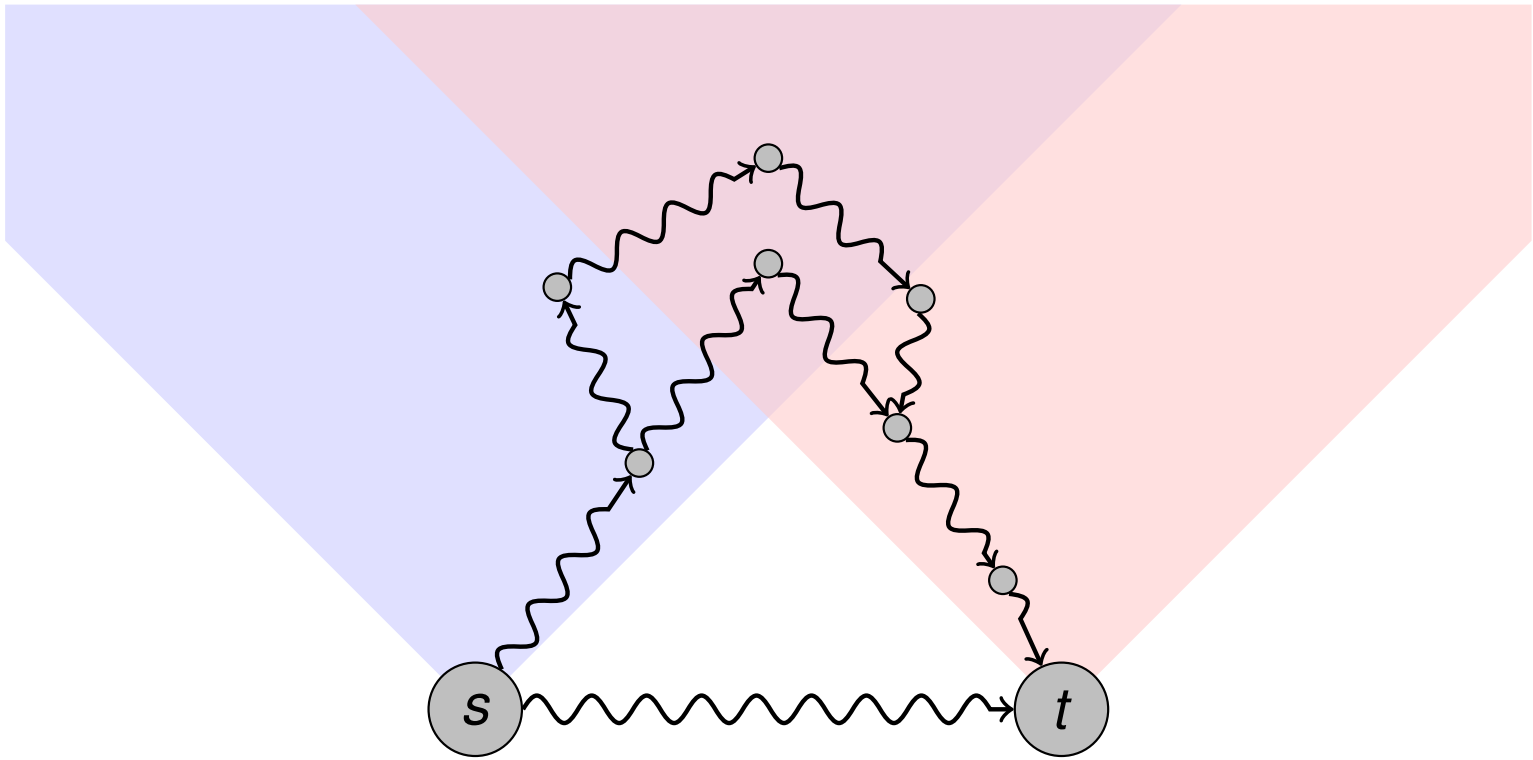
Perform modified bidirectional upward Disjktra search

How to Check Whether Path is Uniquely CH-Representable



Perform modified bidirectional upward Disjktra search

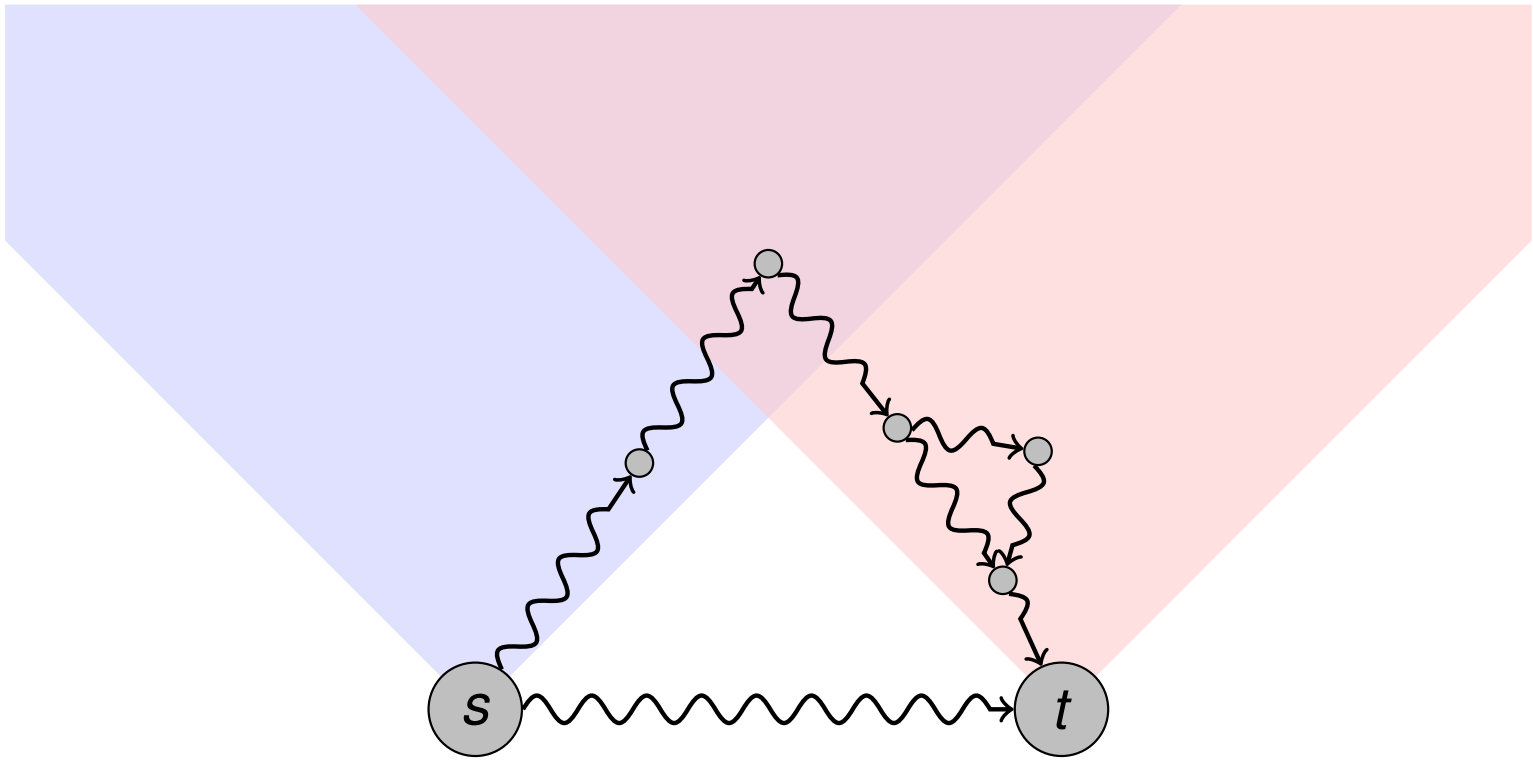
How to Check Whether Path is Uniquely CH-Representable



Perform modified bidirectional upward Disjktra search

Detect multiple top nodes

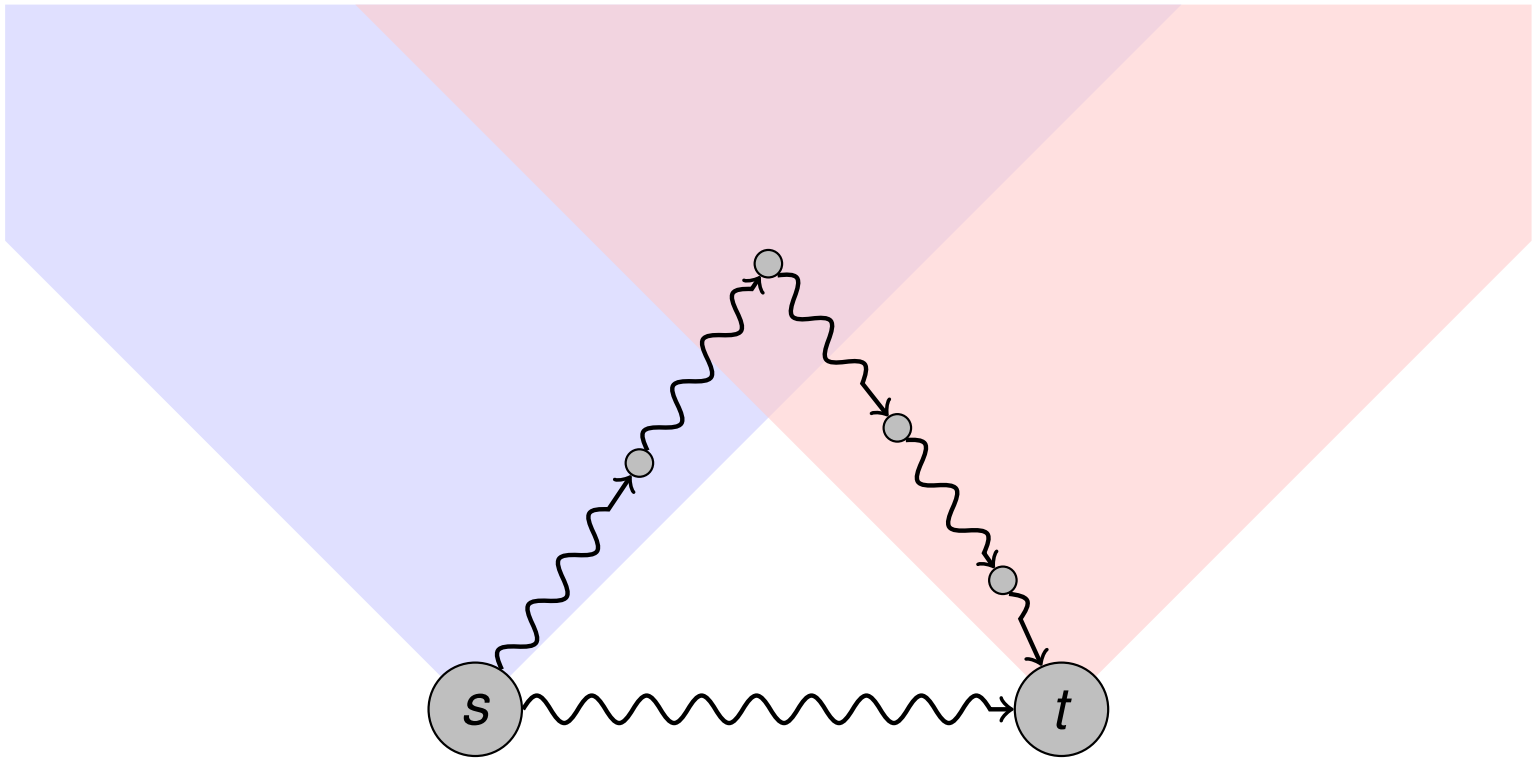
How to Check Whether Path is Uniquely CH-Representable



Perform modified bidirectional upward Disjktra search

Detect non-unique subpaths

How to Check Whether Path is Uniquely CH-Representable



Perform modified bidirectional upward Disjktra search

This is very fast: Hierarchy is flat and sparse.

Compression Rate with CH

Lemma:
Unique shortest path $\Rightarrow$ uniquely CH-representable.
$\not\Leftarrow$

Consequence:
In theory, CH-based representation can have less via nodes.

Compression with Via Nodes

CH-based Compression – Summary

- **Algorithm:**
  Frame-Algor. + binary Scheme + bidir. search in CH
- **Fast:**
  – perform bidir. search $O(\#\,\text{via nodes} \cdot \log |P|)$ times
  – upward search in flat and sparse hierarchy
- **Compression rate:**
  (in theory) even better

Experiments

Compression Time and Rate.

Experimental Setup

German road network

- Nodes: 4.7 million
- Edges: 10.8 million, 7.2 % time-dependent

Simulating of Client and Server

- Different metrics...
- ...for route planning and compression

Evaluated compression algorithms

- Frame-Algor. + modified Dijkstra
- Frame-Algor. + binary Scheme + bidir. search in CH
- **Problem:** Our CH-based implementation pessimistic

Four Metrics (1)

Metric = Edge-Weights + Objective

time-dependent

- **Edge-Weights:** Functions f_e : time $\mapsto \Delta$ travel time
- **Objective:** Find earliest arrival route for departure time τ_0

free flow

- **Edge-Weights:** Time-independent travel times min f_e
- **Objective:** Find shortest path

Four Metrics (2)

Metric = Edge-Weights + Objective

distance

- **Edge-Weights:** Driving distance dd_e
- **Objective:** Find shortest path

energy

- **Edge-Weights:** Estimate energy cost $dd_e + \gamma \min f_e$
- **Objective:** Find shortest path

with $\gamma := 4 \rightsquigarrow$ 1 km costs 0.1€
1 hour costs 14.4€

Results (1)

Server-Metric: Time-Dependent

# route nodes	client metric	method	via nodes #	via nodes max.	via nodes rate[%]	time [ms]
996	free flow	Dijkstra	0.071	3	0.006	1 500.78
		CH-based	0.068	3	0.006	0.36
	distance	Dijkstra	9.771	26	1.045	481.60
		CH-based	9.677	25	1.036	20.98
	energy	Dijkstra	1.103	6	0.125	1 326.17
		CH-based	1.094	6	0.124	1.72

- Fast with CH: less than 21 ms
- Slow with Dijkstra: up to 1.5 s
- Good compression: max 26 via nodes, avg < 10
- 2 bit/edge need $\approx$ 248 byte, 10 via nodes need $\approx$ 104 byte

Results (2)

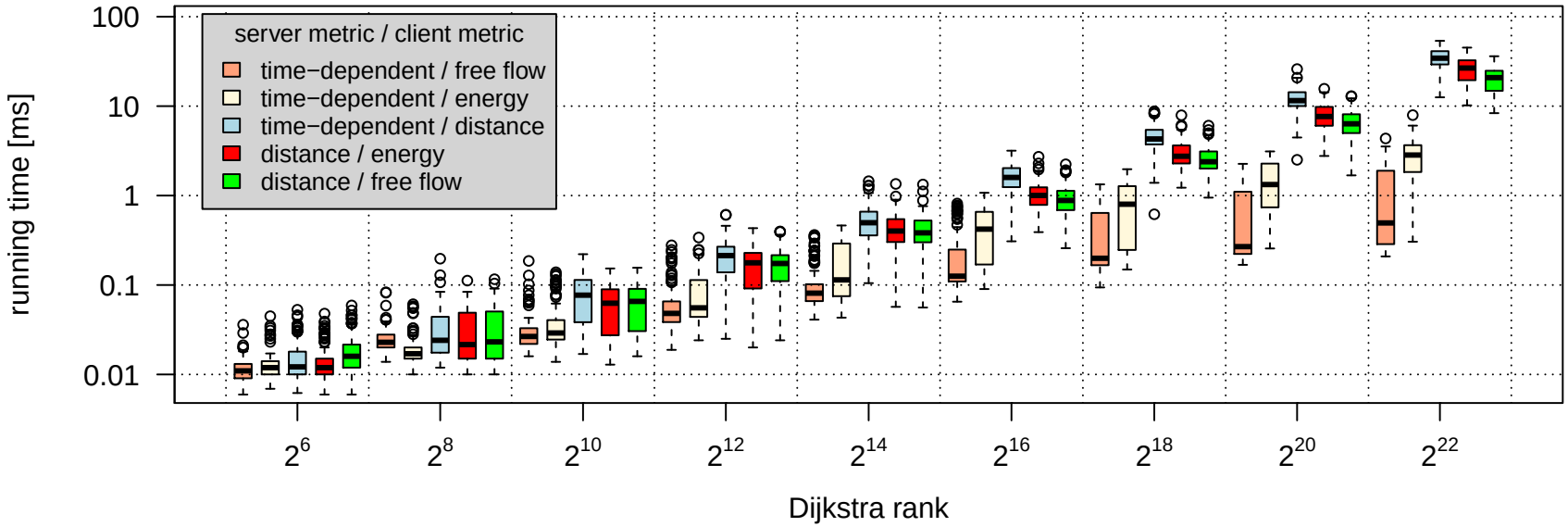
Server-Metric: Distance

# route nodes	client metric	method	via nodes #	via nodes max.	via nodes rate[%]	time [ms]
1 763	free flow	Dijkstra	29.312	76	1.689	162.49
		CH-based	29.284	76	1.688	12.56
	energy	Dijkstra	24.902	69	1.434	182.87
		CH-based	24.876	69	1.433	15.63

- Fast with CH: less than 16 ms
- Slow with Dijkstra: up to 183 ms
- Still good compression: max 76 via nodes, avg < 30
- Dijkstra faster with more via nodes

Results (3)

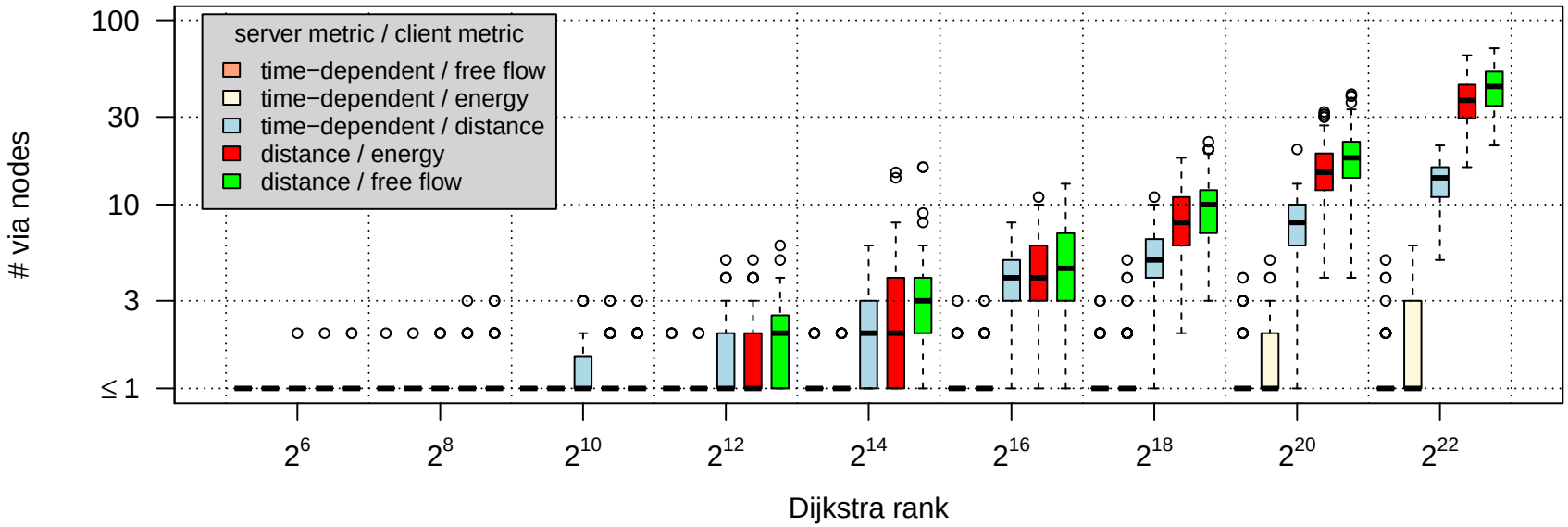
CH-Based: Running Time



- Runnig time increases with route length
- Even longest routes $< 100\ ms$
- Ranks $2^{12} - 2^{14}$: middle sized German town, mostly < 1 ms

Results (4)

CH-Based: # Via Nodes



- # via nodes increases with route length
- There are outliers, but not serios
- Running time roughly increases with via nodes...
  ... remember $O(\log |P| \cdot \#\text{via nodes})$ bidir. searches

Conclusions

Experimental results:

- Fast (de)compression with CH: $< 100\,\text{ms} + 21\,\text{ms}$
- Few via nodes: avg ≤ 30, max 76
- Compression with CH practically not better

Convenience of use:

- Driver experiences no delay...
- ...except for latency of mobile radio network

Future Work

- Speedup Dijkstra-based compression (use ALT or ArcFlags)
- Faster CH-based without repeated bidir. searches?
- Non-pessimistic CH-based better?

Future Work

- Speedup Dijkstra-based compression (use ALT or ArcFlags)
- Faster CH-based without repeated bidir. searches?
- Non-pessimistic CH-based better?

Questions?