Highway Hierarchies Star

Daniel Delling **Peter Sanders**

Dominik Schultes **Dorothea Wagner**

Institut für Theoretische Informatik – Algorithmik I/II

Universität Karlsruhe (TH)

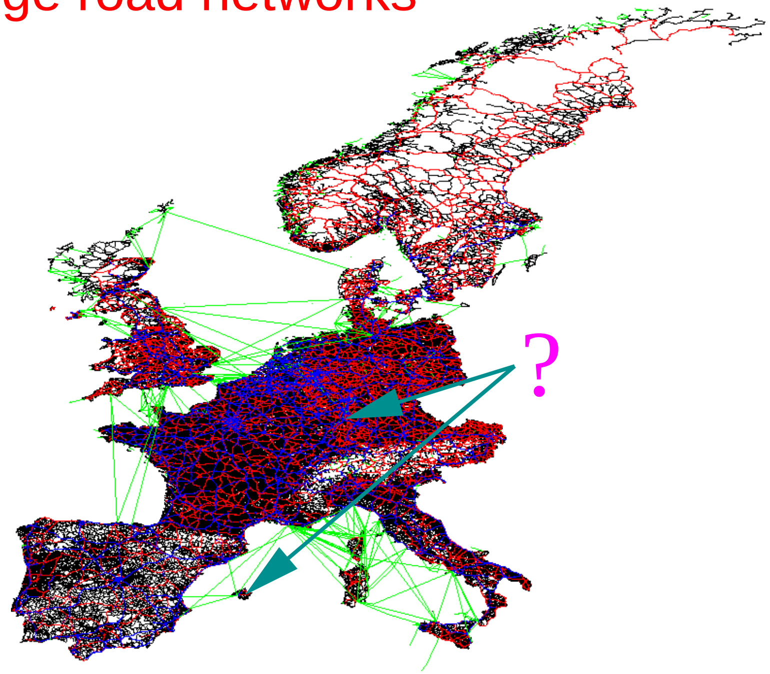
`http://algo2.iti.uka.de/schultes/hwy/`

9th DIMACS Challenge, November 13, 2006

Route Planning

Goals:

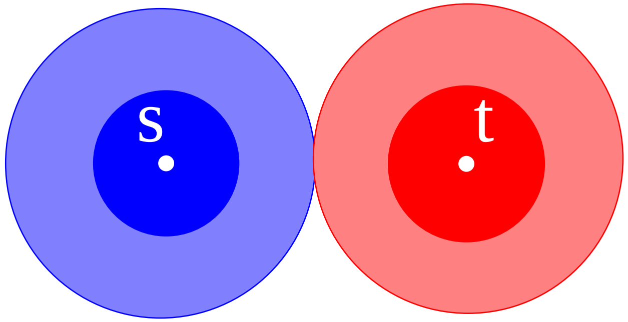
- exact shortest (i.e. fastest) paths in large road networks
- fast queries
- fast preprocessing
- low space consumption



Applications:

- route planning systems in the internet
- car navigation systems
- . . .

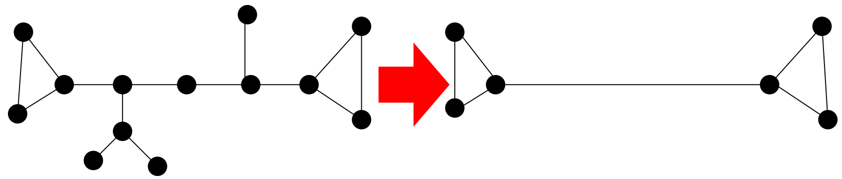
Our Approach: Highway Hierarchies[1]



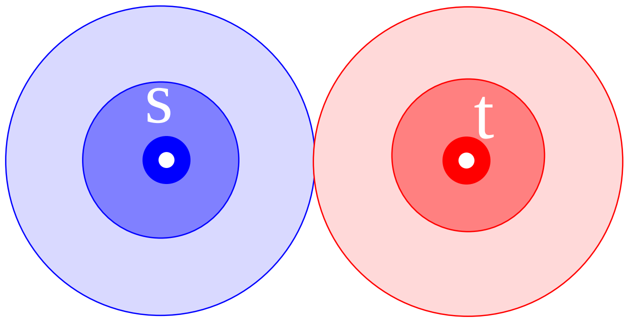
- complete search within a local area
- search in a (thinner) highway network

= minimal graph that preserves all shortest paths

- contract network, e.g.,



- iterate ⇝ highway hierarchy



---

[1] presented at ESA 2005 and ESA 2006

Local Area

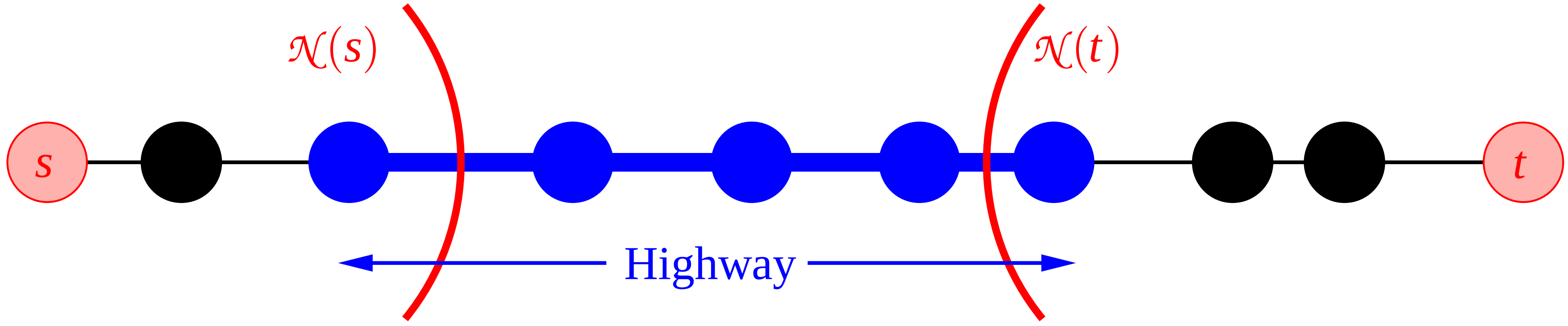
- choose neighbourhood radius $r(s)$

  (by a heuristic)

- define neighbourhood of s

$$\mathcal{N}(s) := \{v \in V \mid d(s,v) \leq r(s)\}$$

Highway Network



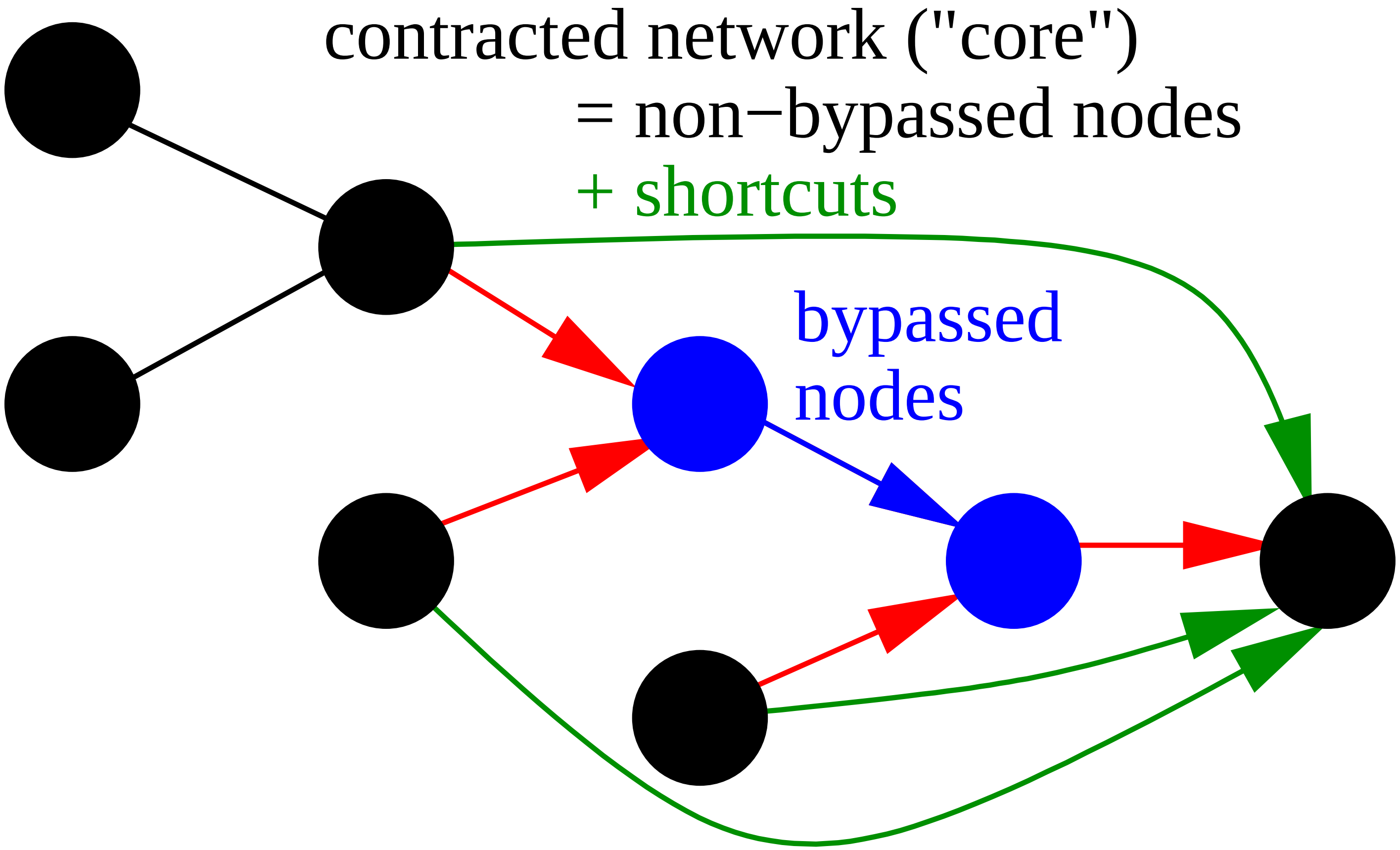
Edge (u,v) belongs to highway network *iff* there are nodes s and t s.t.

- (u,v) is on the "*canonical*" shortest path from s to t

and

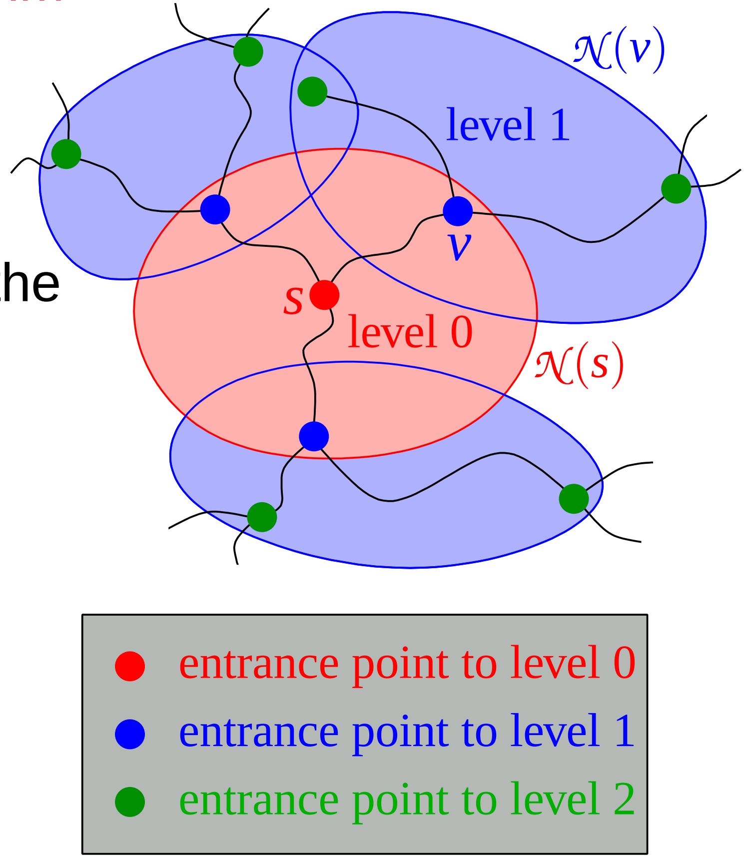
- (u,v) is not entirely within $\mathcal{N}(s)$ or $\mathcal{N}(t)$

Contraction



Query

Bidirectional version of Dijkstra's Algorithm



Restrictions:

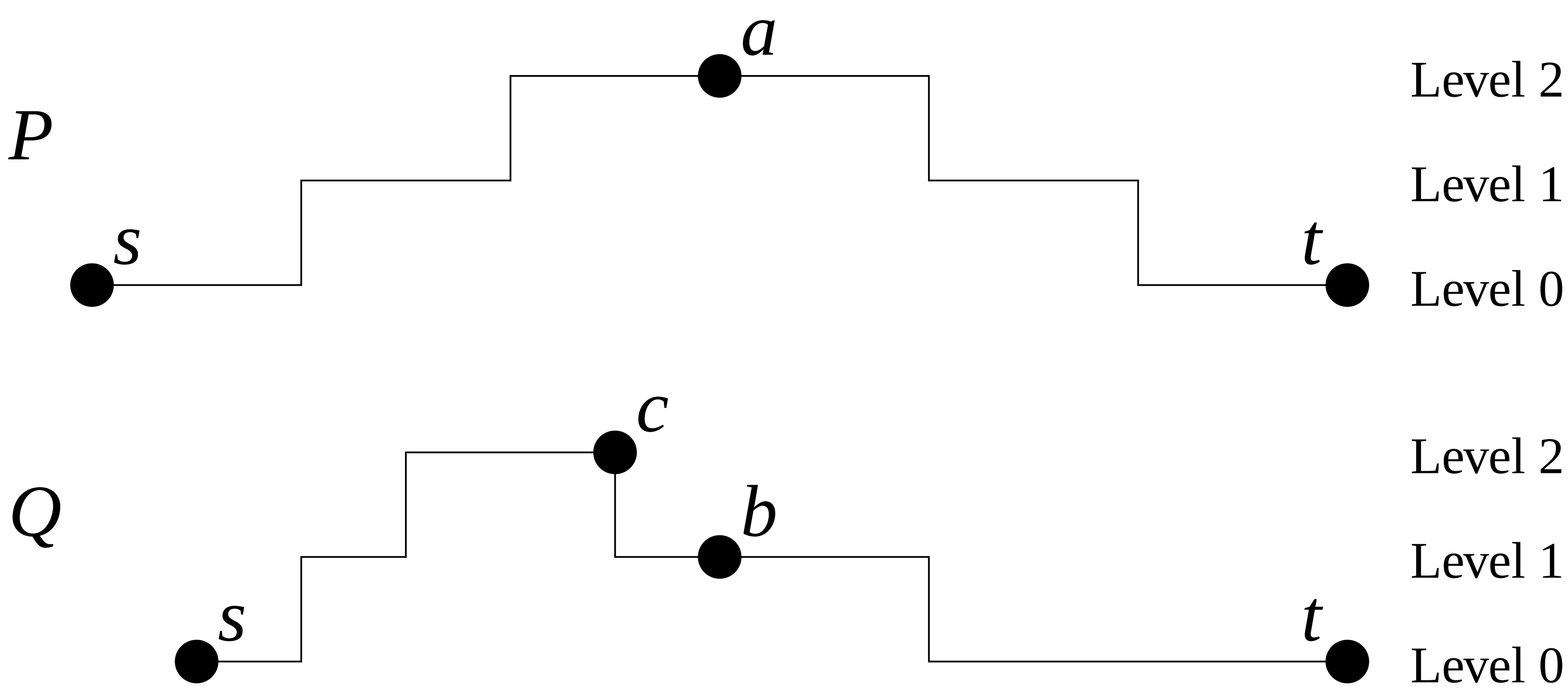
- Do not leave the neighbourhood of the entrance point to the current level.

  Instead: switch to the next level.

- Do not enter a component of bypassed nodes.

Drawbacks

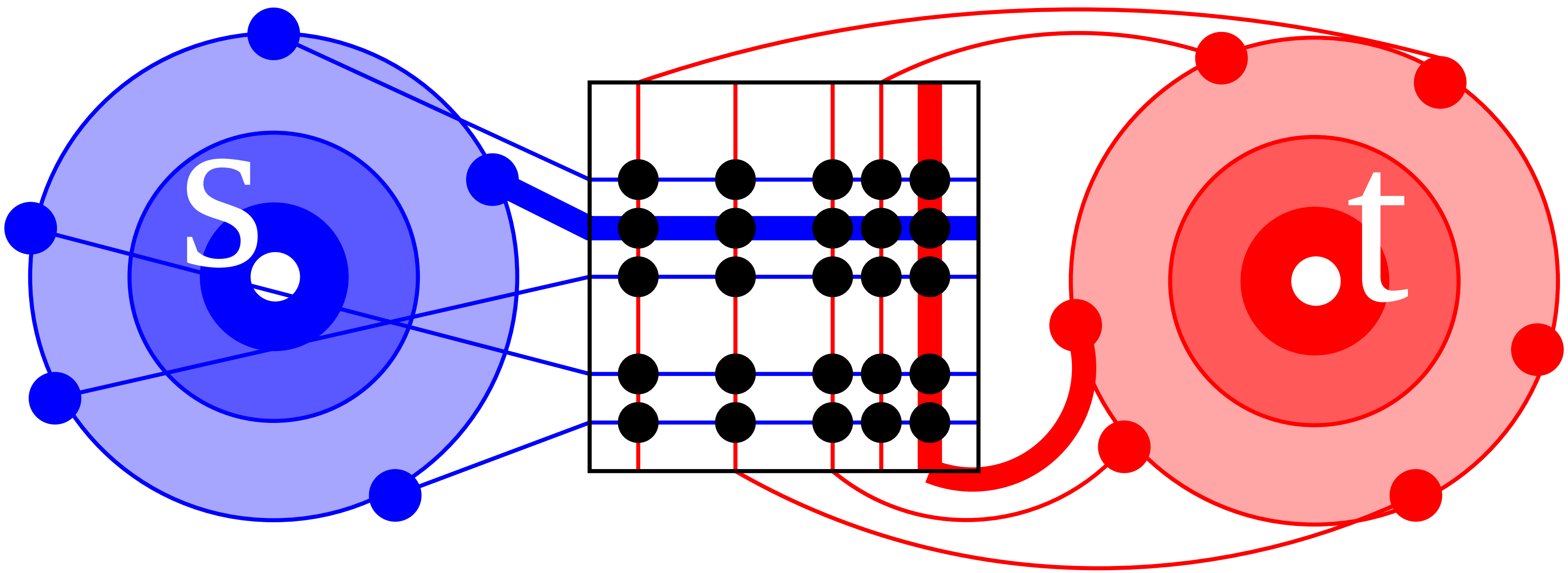
- No effective abort when forward and backward search meet.



Distance Table: Construction

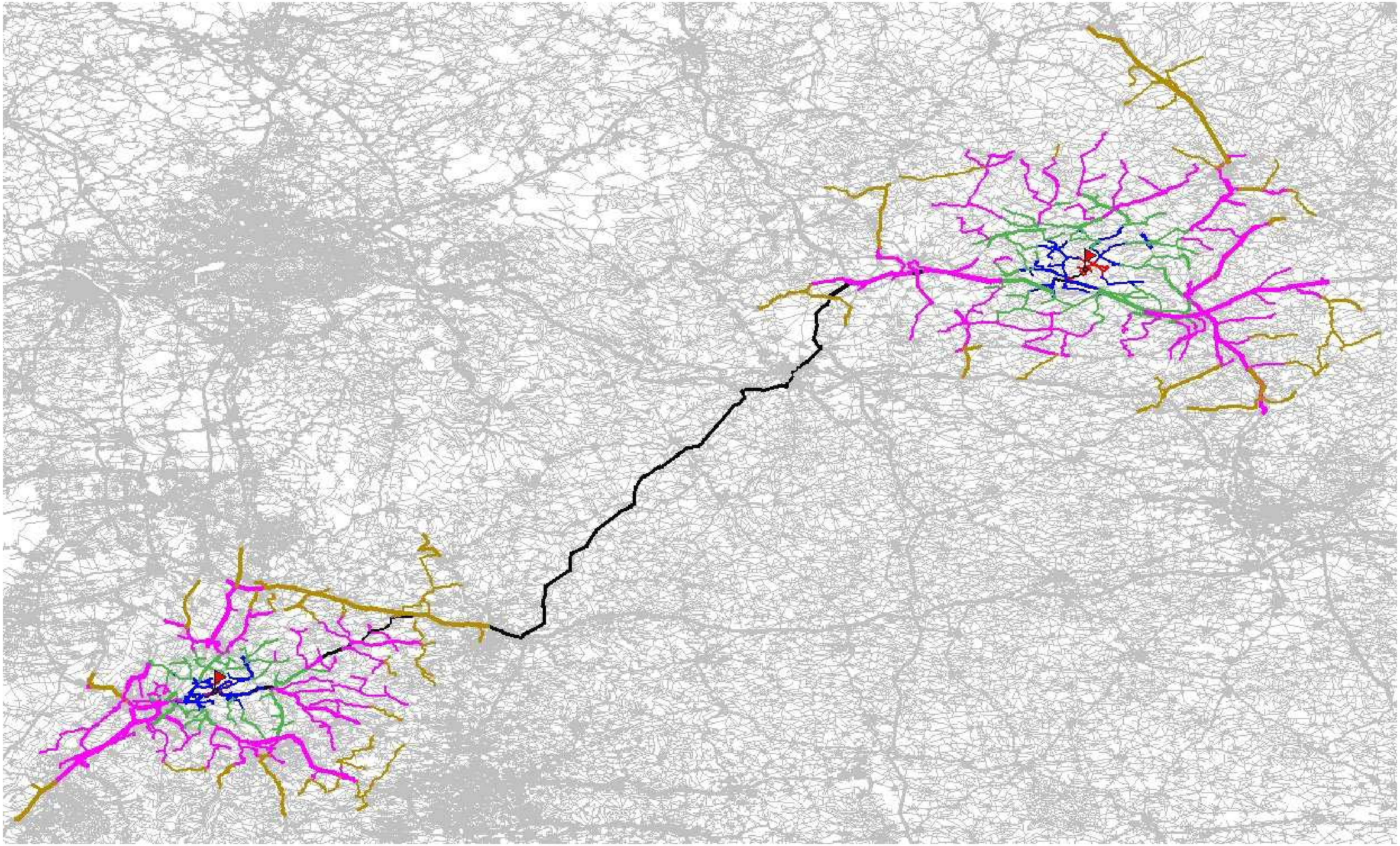
- Construct fewer levels. e.g. 4 instead of 9
- Compute an all-pairs distance table
  for the core of the topmost level L. 13 465 $\times$ 13 465 entries

Distance Table: Query



- Abort the search when all entrance points in the core of level L have been encountered. $\approx$ 55 for each direction
- Use the distance table to bridge the gap. $\approx 55 \times 55$ entries

Distance Table: Search Space Example

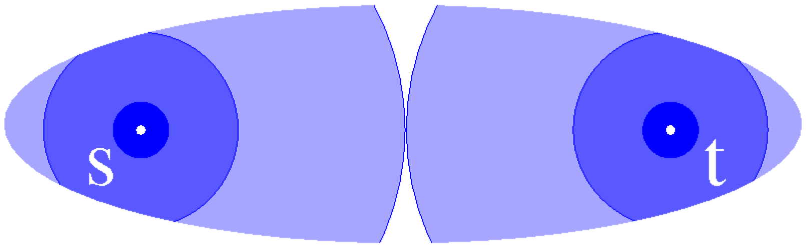


Drawbacks

- Search is not goal-directed.

  → **main topic** of this talk:

  combination with a goal-directed approach



- No effective abort when forward and backward search meet.

  → **main problem** that we face

Goal-directed Search

- push search towards target
- add potential π to priority of each node
- A* equivilant to DIJKSTRA's algorithm on graph with reduced costs

$$w_\pi(u,v) = w(u,v) - \pi(u) + \pi(v)$$

- potential feasible if reduced costs ≥ 0
- better potential $\longrightarrow$ smaller search space

Bidirectional A*

- problems bidirectional variant:
  - forward potential π_f, backward potential π_r
  - both searches might operate on different graphs
- solution:
  - potentials consistent iff $w_{\pi_f}(u,v)$ in G equal $w_{\pi_r}(v,u)$ in reverse graph
  - use average potentials: $p_f = (\pi_f - \pi_r)/2 = -p_r$
  - but: leads to worse lower bounds

ALT

- bidirected goal-directed search (**A***)
- use **L**andmarks to compute potentials (lower bounds)
- Preprocessing:
  - choose landmarks from node set
  - calculate distances from and to all nodes
- On-line stage:
  - use **T**riangle inequality to compute lower bounds on the distance to the target

$$d(s,t) \geq d(L_1,t) - d(L_1,s) \text{ and } d(s,t) \geq d(s,L_2) - d(t,L_2)$$

Landmark-Selection

- reduction of search space highly depends on quality of landmarks
- several selection strategies:
  - avoid:

    identify regions that are not covered by landmarks
  - maxCover:

    $4 \cdot$avoid $+$ local optimisation
  - advancedAvoid:

    reselect first avoid landmarks
- long computation time: $\approx$ 90 minutes for 16 maxCover landmarks

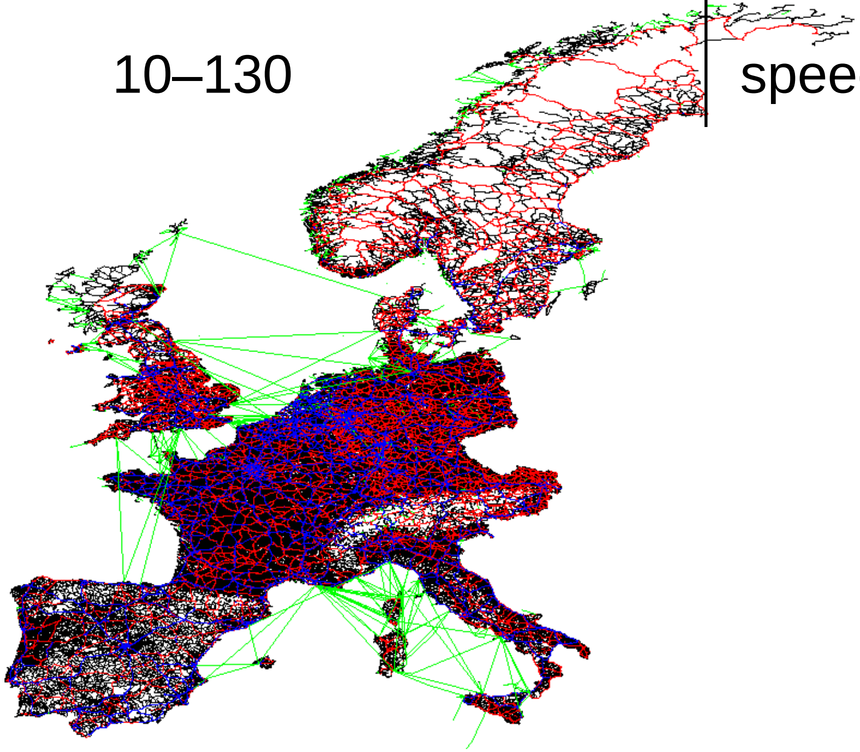
Using Highway Hierarchies for Selection

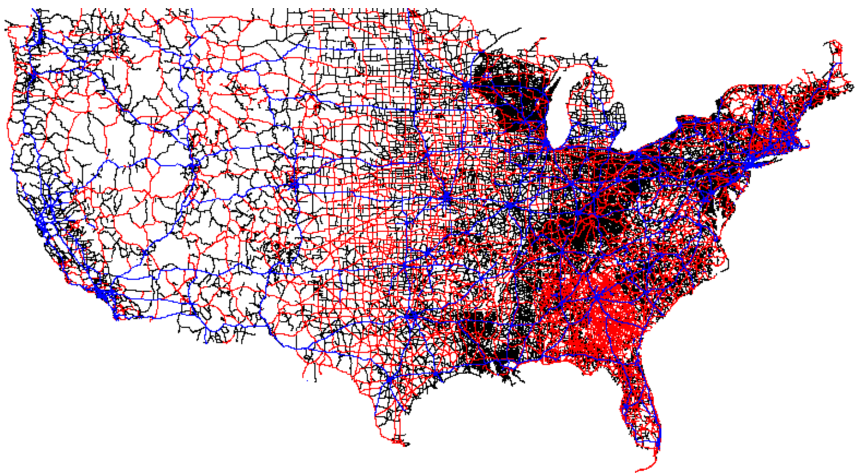
idea: reduce preprocessing by using hierarchy for selection

- advantages:
  - reduction of prepocessing:
    $<$1 minute for selecting 16 maxCover landmarks on core-3
  - important edges are covered
- disadvantages:
  - highway hierarchy shrinks to the center
  - nodes on the edge of the map are good landmarks
  - for ALT: compute distances (6 minutes)

Experiments

W. Europe (PTV)	**Our Inputs**	USA (TIGER/Line)
18 010 173	#nodes	23 947 347
42 560 279	#directed edges	58 333 344
13	#road categories	4
10–130	speed range [km/h]	40–100





Landmark Quality (different cores)

	times metric (Europe)			distance metric (Europe)		
[#settled nodes]	avoid	adv.avoid	maxCov	avoid	adv.avoid	maxCov
full graph	93 520	86 340	75 220	253 552	256 511	230 110
core-1	84 515	82 423	75 992	254 596	252 002	230 979
core-2	89 001	86 611	75 379	259 145	257 963	230 310
core-3	91 201	91 163	72 310	264 821	275 991	239 584

- (almost) no loss of quality for higher cores
- advancedAvoid not worth the effort
- maxCover core-3 outperforms avoid in general

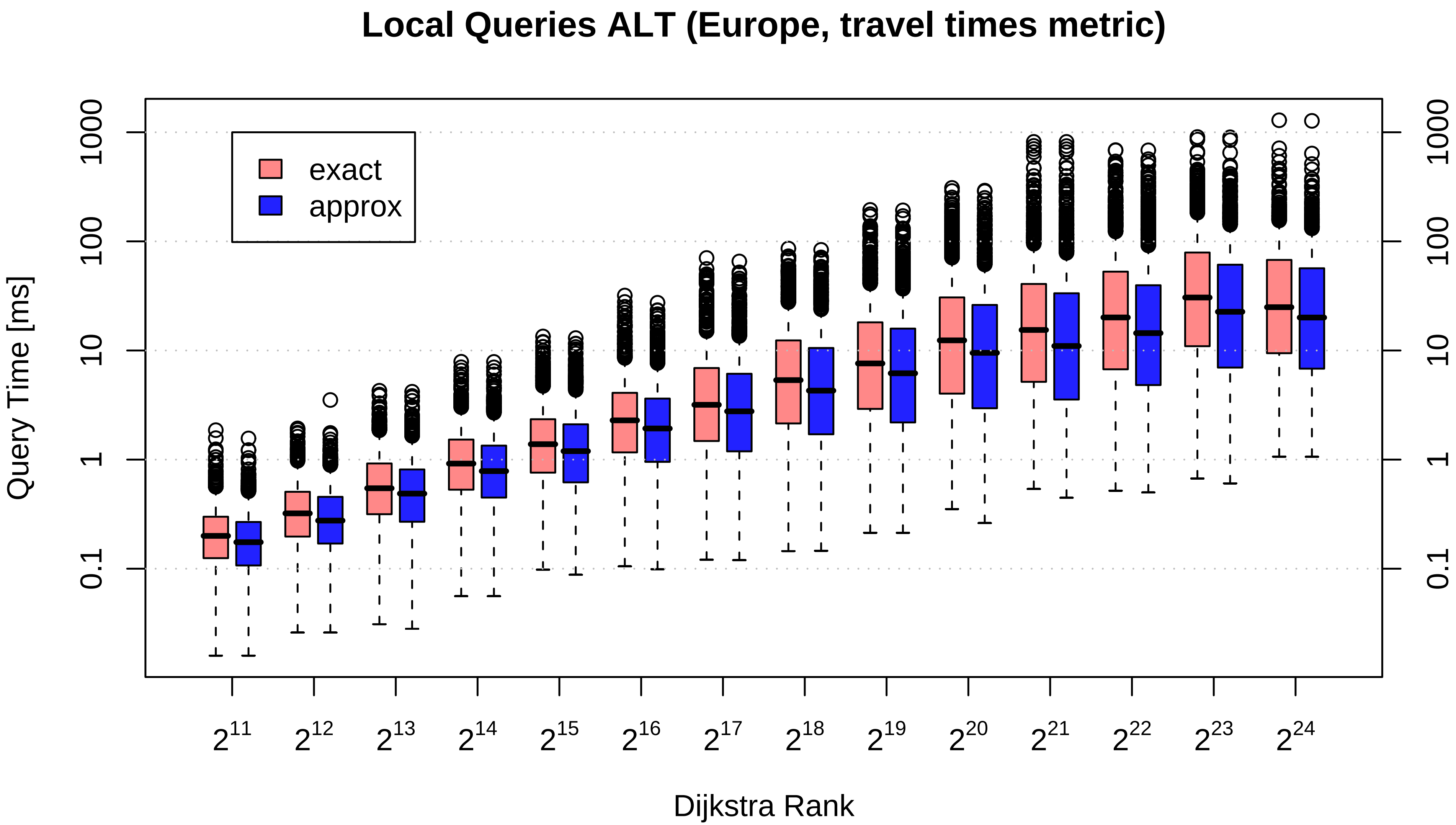
→ switching to higher cores seems promising

Landmark Quality (different strategies)

- all landmarks from full graph, 10 different sets

	times metric (Europe)			distance metric (Europe)		
[#settled nodes]	average	min	max	average	min	max
avoid	93 520	72 720	103 929	253 552	241 609	264 822
adv.avoid	86 340	72 004	95 663	256 511	218 335	283 911
maxCov	75 220	71 061	77 556	230 110	212 641	254 339

- minimum the same for all strategies
- maxCover more robust



$\longrightarrow$ approximate queries not much faster

Highway Hierarchies*

Combination of Highway Hierarchies with ALT

- replace edge weights by reduced costs
- use potential functions π_f and π_r

⇝ search directed to the respective target

⇝ we quickly find a good (or even the best) path

⇝ good upper bounds are available early

However: only the order of events is changed;

search space is not reduced

Highway Hierarchies*

Solution:

- abort when forward and backward search meet
  - works well for ALT (combined with reach-based routing)
  - does not work with highway hierarchies
- pruning
  - edge pruning
  - node pruning (used by HH*):

    $\underbrace{\text{key of node } u}_{\text{lower bound}} > \text{upper bound} \longrightarrow$ do not relax u's edges

Positive Aspects

- no consistent potential functions required

  ⇝ more effective goal-direction

  ⇝ good upper bounds available very early

- node pruning very simple

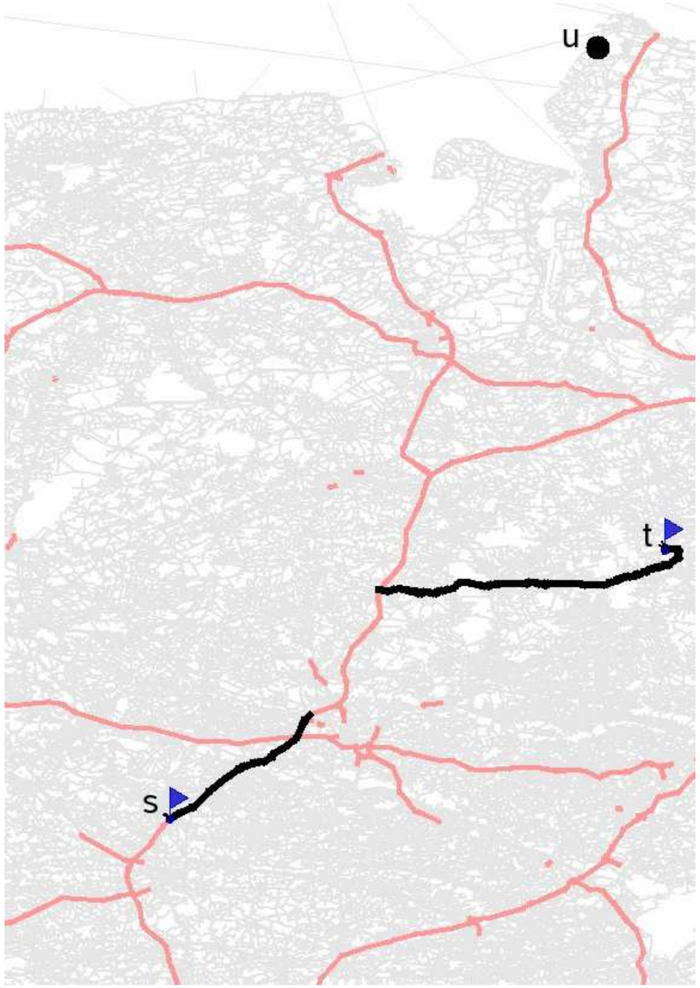
  if one node is pruned, search can be stopped (in that direction)

- select only one landmark for each direction,

  no dynamic updates of the chosen landmark(s)

  distance table bridges the middle part

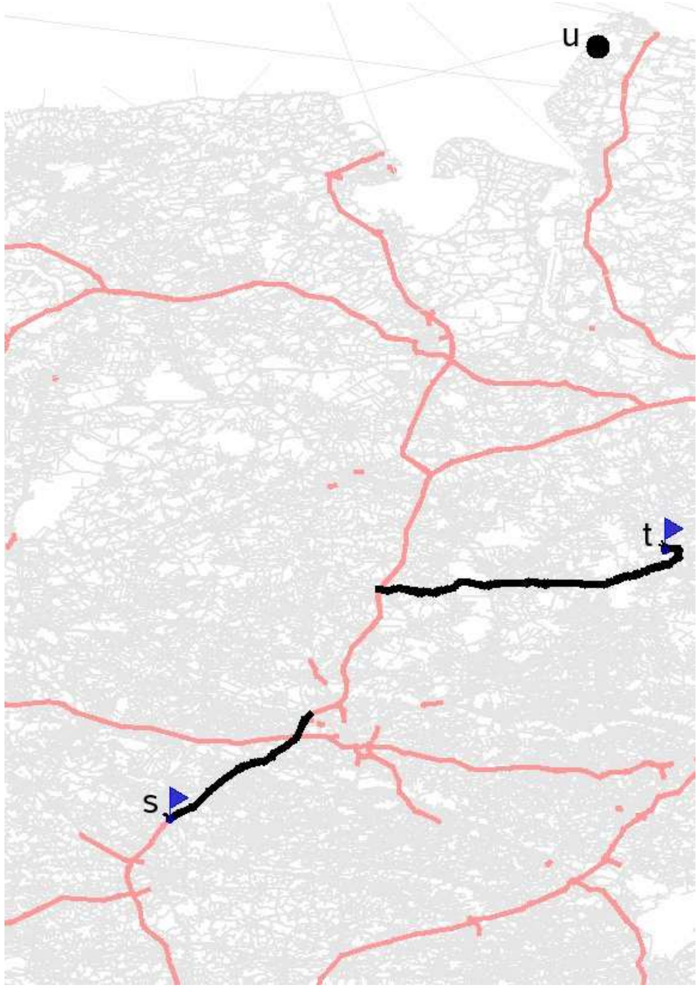
Negative Aspect (for travel time metric)



goal-direction works well

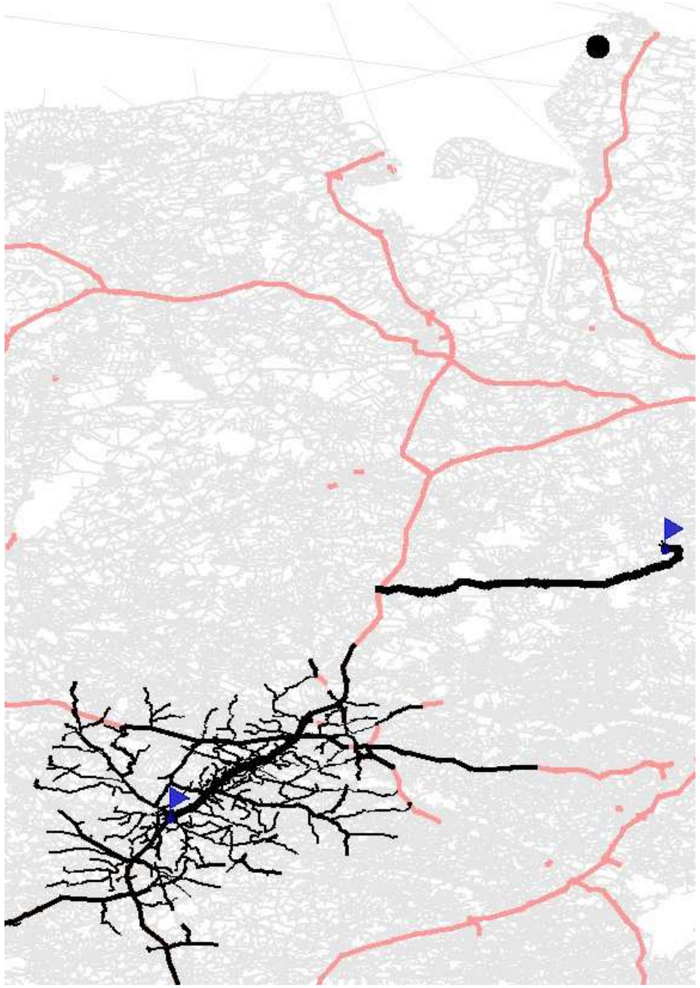
$s \longrightarrow t$ and $s \longrightarrow u$: common subpath

Negative Aspect (for travel time metric)



goal-direction works well

$s \longrightarrow t$ and $s \longrightarrow u$: common subpath



pruning fails

$d(s,t) \geq d(s,u) - d(t,u)$

Approximate Queries

Dilemma:

- finding a good path: very fast
- guaranteeing optimality: comparatively slow

$\rightsquigarrow$ **guarantee weakened**: look for a path P s.t.

$$\text{length of } P \leq (1+\varepsilon)\cdot OPT$$

Implementation:

adapted node pruning: $(1+\varepsilon)\cdot$ (key of node u) > upper bound

Optimisations

Reducing Space Consumption

Store landmark-distances only at all core-1 nodes.

$\leadsto$ split search in two phases:

- initially, non-goal-directed search to the core-1 nodes
- then, goal-directed search from the core-1 nodes

Limiting Component Sizes

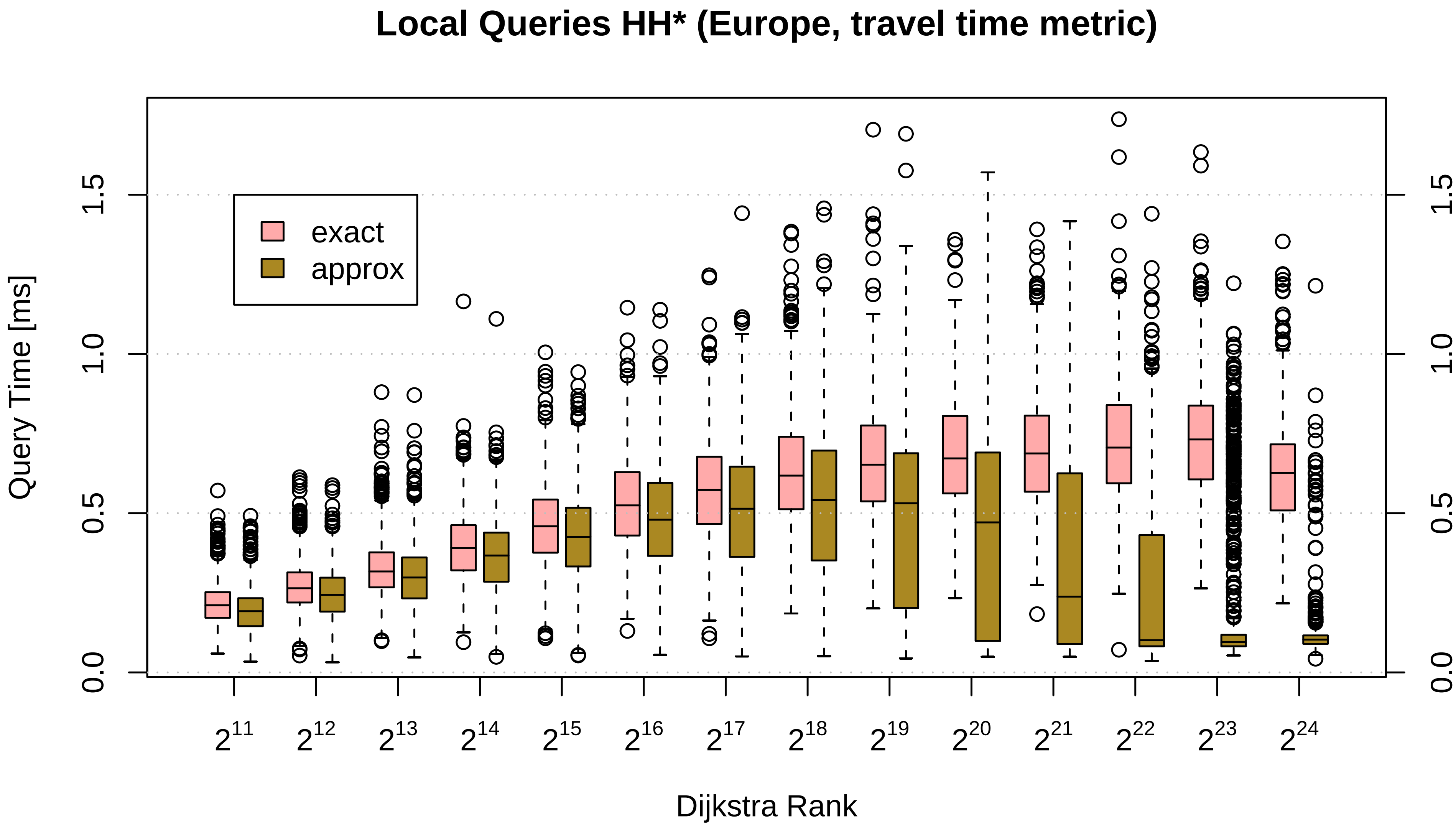
introduce a shortcut hops limit

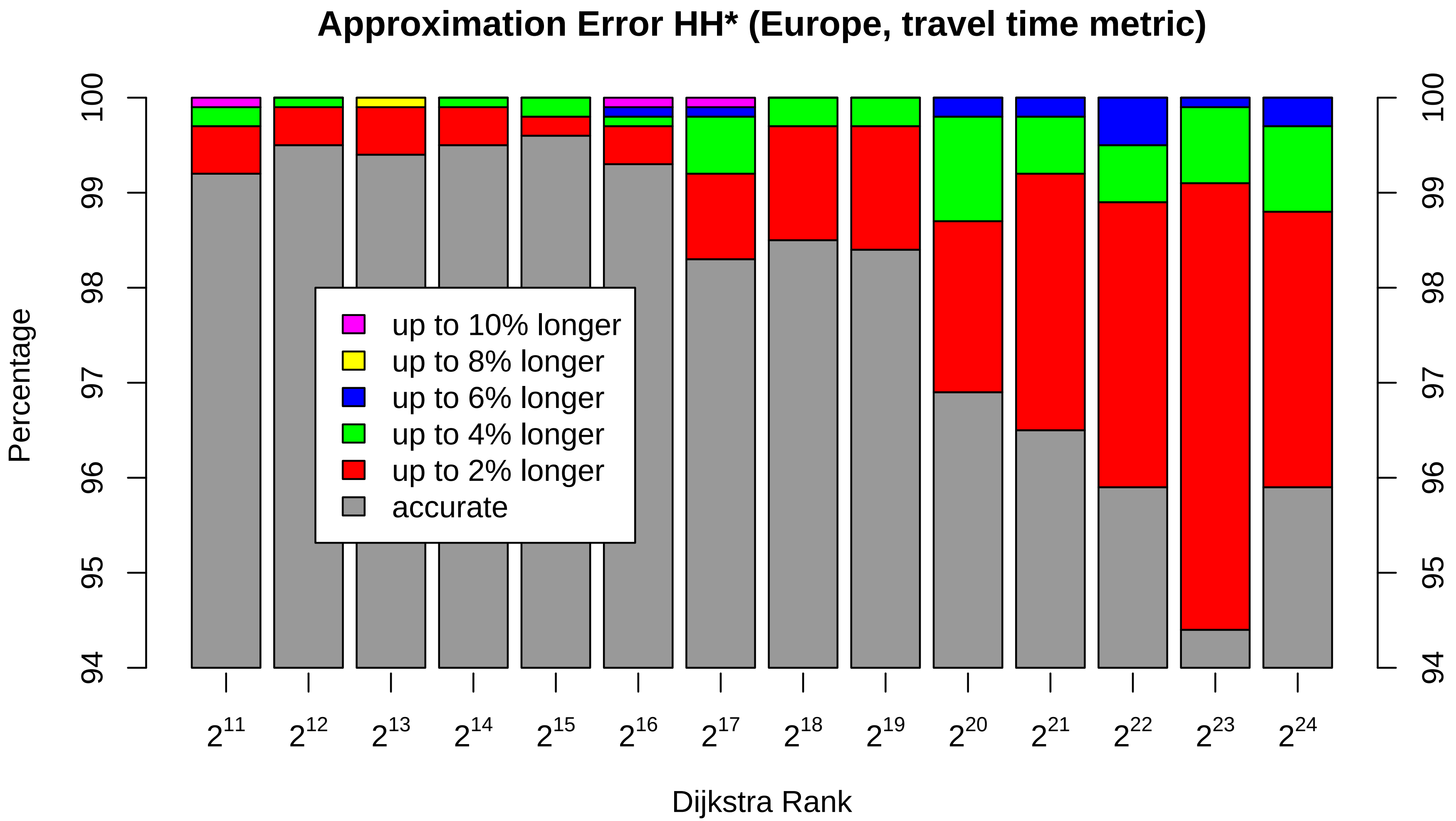
$\leadsto$ indirectly limits the component sizes

important to ensure an efficient initial query phase

Main Results

metric		Europe ∅	DistTab	ALT	both	
time	preproc. time [min]	16	18	19	21	
	total disk space [MB]	886	1 273	1 326	1 713	
	#settled nodes	1 662	916	916	686	(176)
	query time [ms]	1.49	0.79	1.04	0.68	(0.21)
dist	preproc. time [min]	46	46	49	48	
	total disk space [MB]	894	1 506	1 337	1 948	
	#settled nodes	10 284	5 067	3 347	2 138	(177)
	query time [ms]	10.93	6.02	4.33	2.54	(0.30)


Local Queries HH* (Europe, travel time metric)
exact
approx
Query Time [ms]
0.0
0.5
1.0
1.5
2^{11} 2^{12} 2^{13} 2^{14} 2^{15} 2^{16} 2^{17} 2^{18} 2^{19} 2^{20} 2^{21} 2^{22} 2^{23} 2^{24}
Dijkstra Rank



guaranteed maximum error ε = 10%

actual total error (for a random sample of 1 000 000 node pairs) = 0.056%

Determine Shortest Paths

1. Bridge the Distance Table Gap

from the forward entrance point u to the backward entrance point v

greedily determine the next hop:

while $u \neq v$ **do**
    foreach $(u, w) \in$ "edges in the topmost core" **do**
        if $d(u, w) + d(w, v) = d(u, v)$ **then**
            $u := w$;
            break;

Note: $d(w, v)$ can be looked up in the distance table.

Determine Shortest Paths

2. Unpack Shortcuts

- **Variant 1**
  - no additional data
  - for each shortcut (u,v), perform a search from u to v
  - use some pruning rules
- **Variant 2**
  - store for each shortcut unpacking information (recursively)
- **Variant 3**
  - store for important shortcuts complete unpacking information
  - no recursion

Determine Shortest Paths

	Europe		
	preproc. [min]	space [MB]	query [ms]
Variant 1	0:00	0	16.70
Variant 2	1:11	112	0.45
Variant 3	1:15	180	0.17

Summary

- selecting landmarks only on a contracted graph

  $\leadsto$ saves preprocessing time.

- storing landmark-distances only on a contracted graph

  $\leadsto$ saves space

- highway hierarchies can handle the distance metric
  - increased preprocessing time ($\approx$ factor 2)
  - similar memory usage
  - increased query time ($\approx$ factor 3–4)

Summary

- for the travel time metric:
  - distance table optimisation slightly better than ALT
  - combination yields only small improvement
- for the distance metric:
  - ALT better than distance table optimisation
  - combination worthwhile
- approximate queries: very fast, only small errors
- fast computation of complete descriptions of the shortest paths

Work in Progress

- computation of $M \times N$ distance tables

  (e.g. $10\,000 \times 10\,000$ table in one minute)

  joint work with [S. Knopp, F. Schulz, D. Wagner][2,3]

  to be presented at ALENEX '07

- storing all entrance points into the core of the topmost level

  $\rightsquigarrow$ *very* fast queries ($\longrightarrow$ tomorrow's talk)

---

[2] Universität Karlsruhe, Algorithmik I

[3] PTV AG, Karlsruhe

Future Work

- fast, local updates on the highway network

  (e.g. for traffic jams)

- implementation for mobile devices

  (flash access, . . .)

- multi-criteria shortest paths

  joint work with [M. Müller-Hannemann, M. Schnee][4]

- . . .

---

[4]Technische Universität Darmstadt