Algorithm Engineering for Large Graphs

Engineering Route Planning Algorithms

Peter Sanders **Dominik Schultes**

Universität Karlsruhe (TH)

Online Topological Ordering for Dense DAGs

Deepak Ajwani **Tobias Friedrich** **Ulrich Meyer**

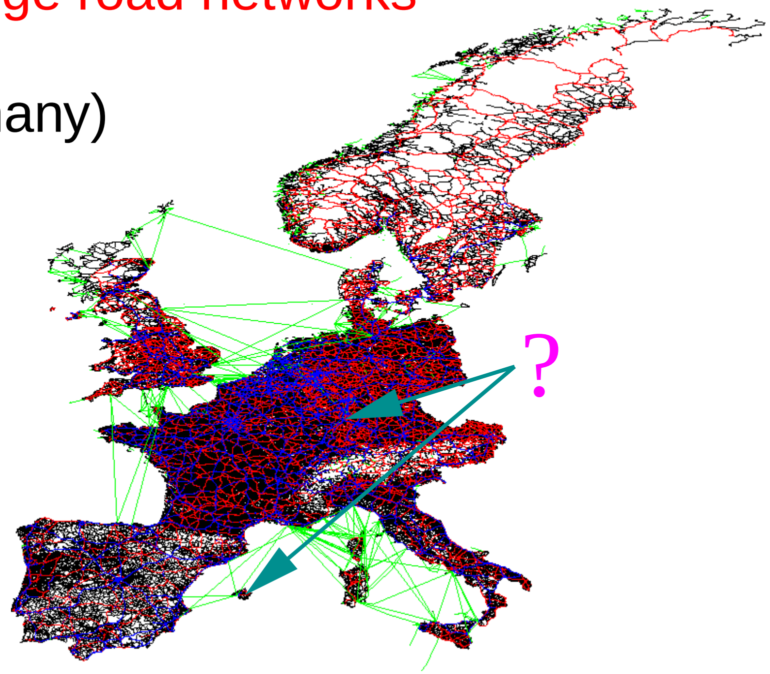
MPII Saarbrücken Universität Frankfurt

Freiburg, July 4, 2007

Route Planning

Goals:

- exact shortest (i.e. fastest) paths in large road networks
- fast queries (point-to-point, many-to-many)
- fast preprocessing
- low space consumption
- fast update operations



Applications:

- route planning systems in the internet, car navigation systems,
- traffic simulation, logistics optimisation

Overview

Transit Node Routing
very fast queries
[DIMACS 06, ALENEX 07, Science 07]

HH Star
goal−directed
[DIMACS 06]

Highway Hierarchies
foundation
[ESA 05, ESA 06]

Many−to−Many
compute distance tables
[ALENEX 07]

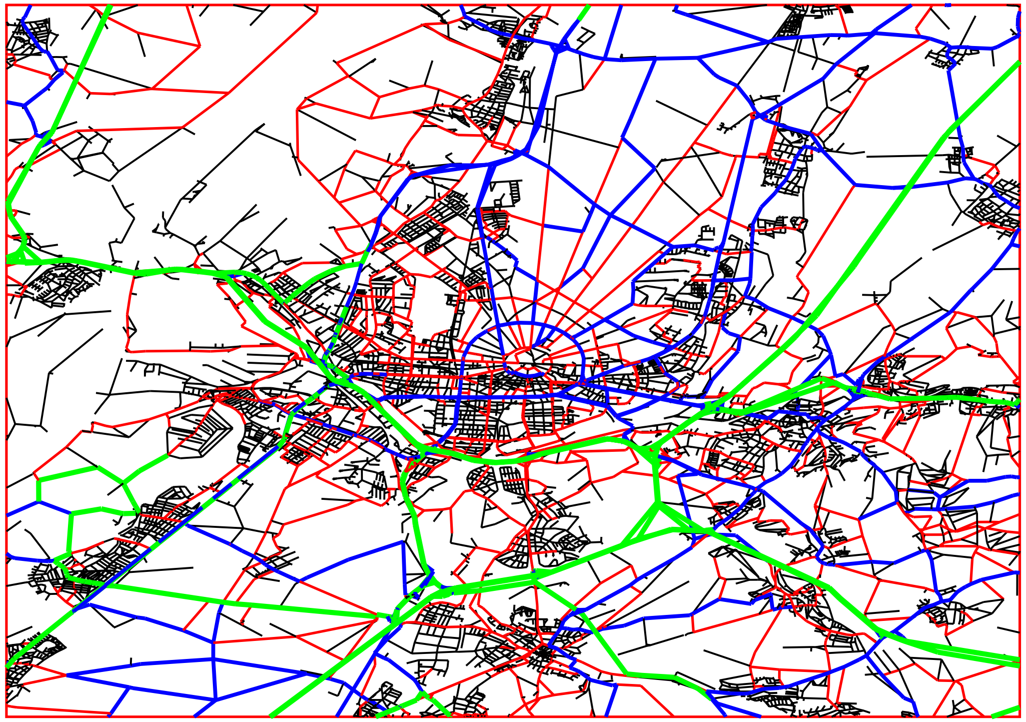
Hwy−Node Routing
allow edge weight changes
[WEA 07]

Highway Hierarchies

[ESA 05, ESA 06]

Construction: iteratively alternate between

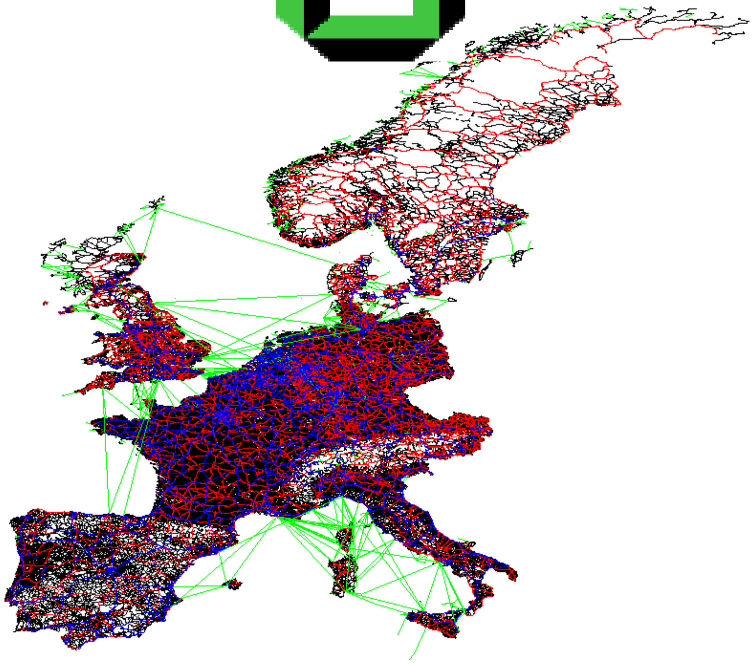
- removal of low degree nodes
- removal of edges that only appear on shortest paths close to source or target



yields a hierarchy of highway networks

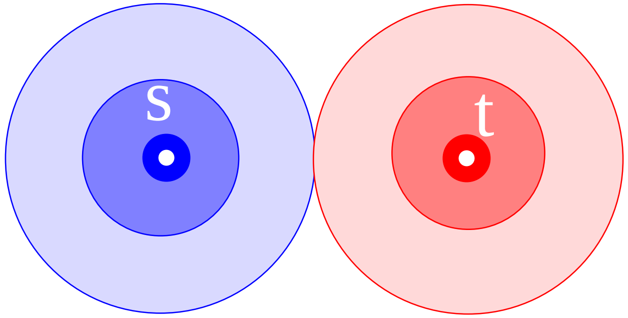
in a sense, classify roads / junctions by 'importance'

Highway Hierarchies



- foundation for our other methods
- directly allows point-to-point queries
- 16 min preprocessing
- 0.76 ms to determine the path length
- 0.93 ms to determine a complete path description
- reasonable space consumption (68 bytes/node)

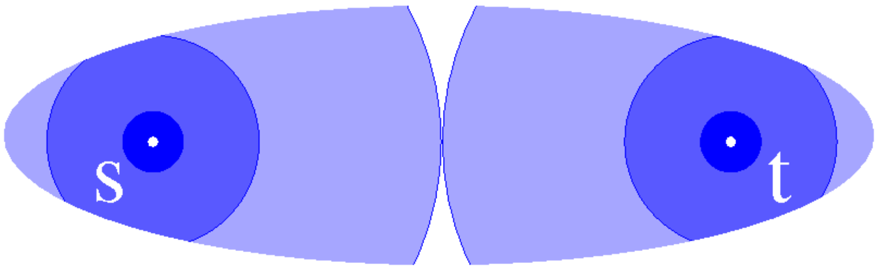
  can be reduced to 17 bytes/node



Highway Hierarchies Star

joint work with D. Delling, D. Wagner [DIMACS Challenge 06]

- combination of highway hierarchies with goal-directed search
- slightly reduced query times (0.68 ms)
- more effective
  - for approximate queries or
  - when a distance metric instead of a travel time metric is used

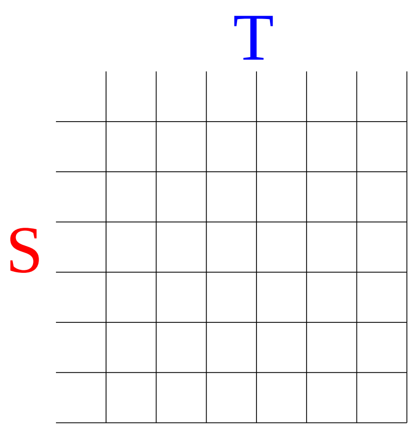


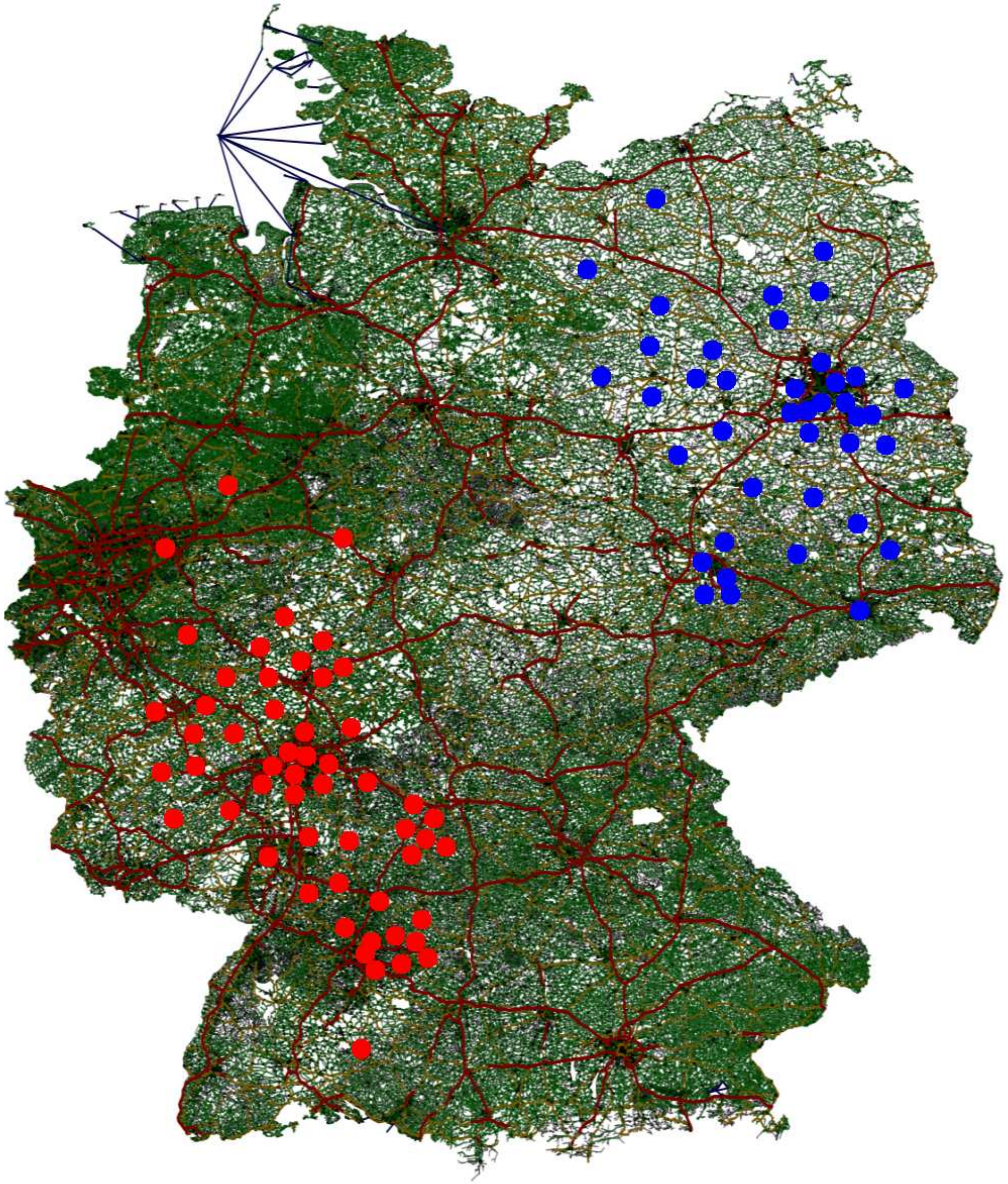
Many-to-Many Shortest Paths

joint work with S. Knopp, F. Schulz, D. Wagner

[ALENEX 07]

- efficient many-to-many variant of the highway hierarchies query algorithm
- 10 000 $\times$ 10 000 table in one minute





Transit-Node Routing

[DIMACS Challenge 06, ALENEX 07, Science 07]

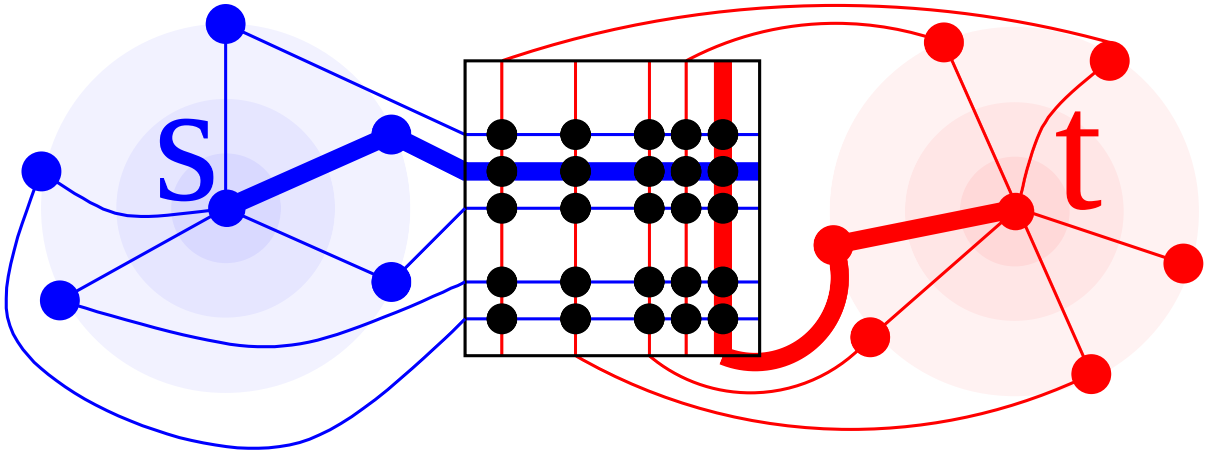
joint work with H. Bast, S. Funke, D. Matijevic

- *very* fast queries

  (down to 6 μs, 1 000 000 times faster than DIJKSTRA)

- winner of the 9th DIMACS Implementation Challenge

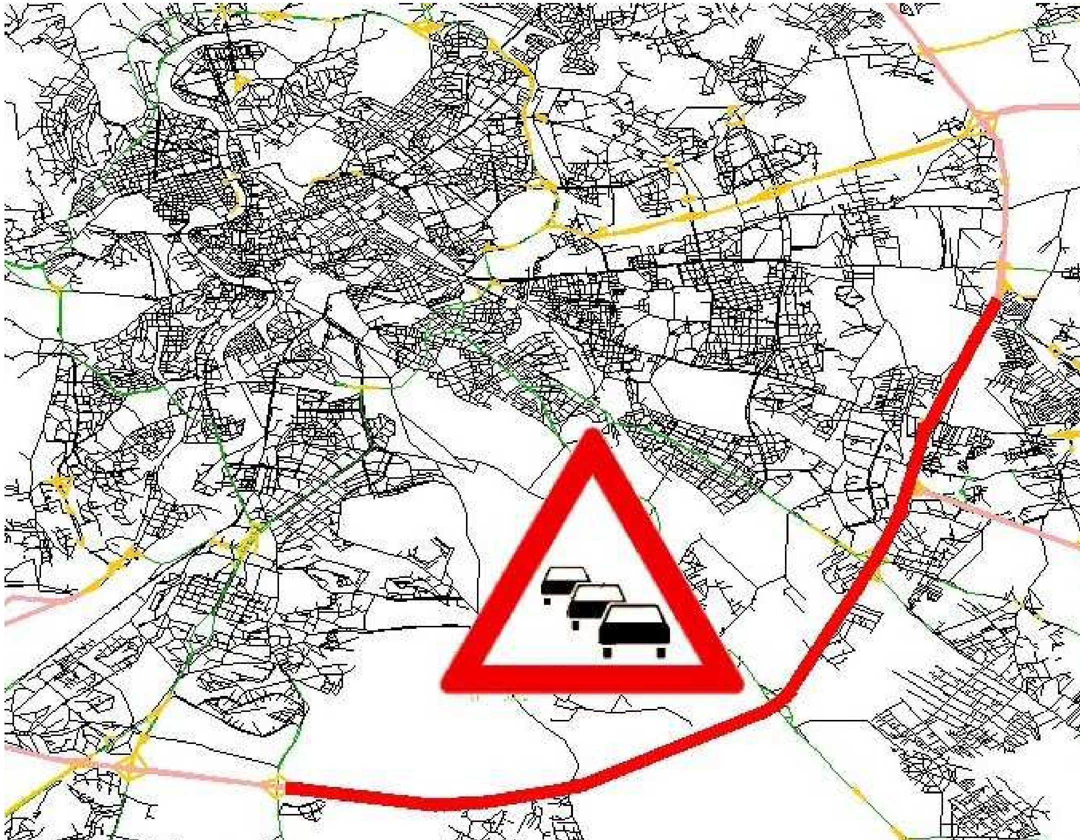
- more preprocessing time (2:44 h) and space (251 bytes/node) needed



Highway-Node Routing

[WEA 07]

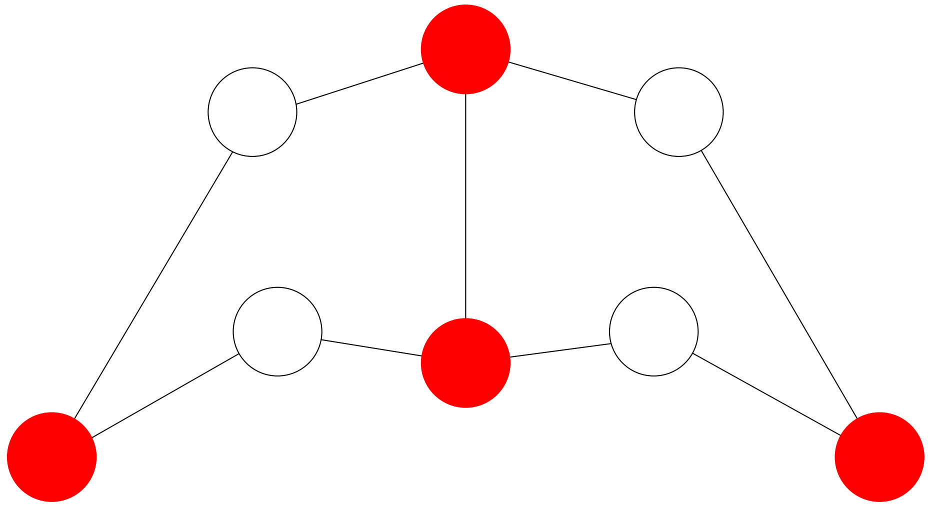
- performance similar to highway hierarchies
- outstandingly low space requirements
- conceptually very simple
- handles dynamic scenarios



Overlay Graph

[Holzer, Schulz, Wagner, Weihe, Zaroliagis 2000–2007]

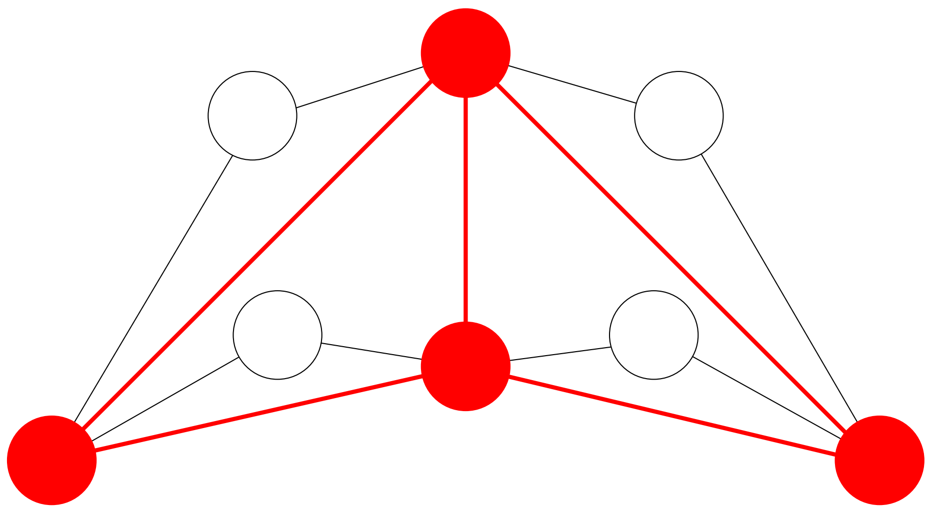
- graph $G = (V, E)$ is given
- select node subset $S \subseteq V$



Overlay Graph

[Holzer, Schulz, Wagner, Weihe, Zaroliagis 2000–2007]

- graph $G = (V, E)$ is given
- select node subset $S \subseteq V$

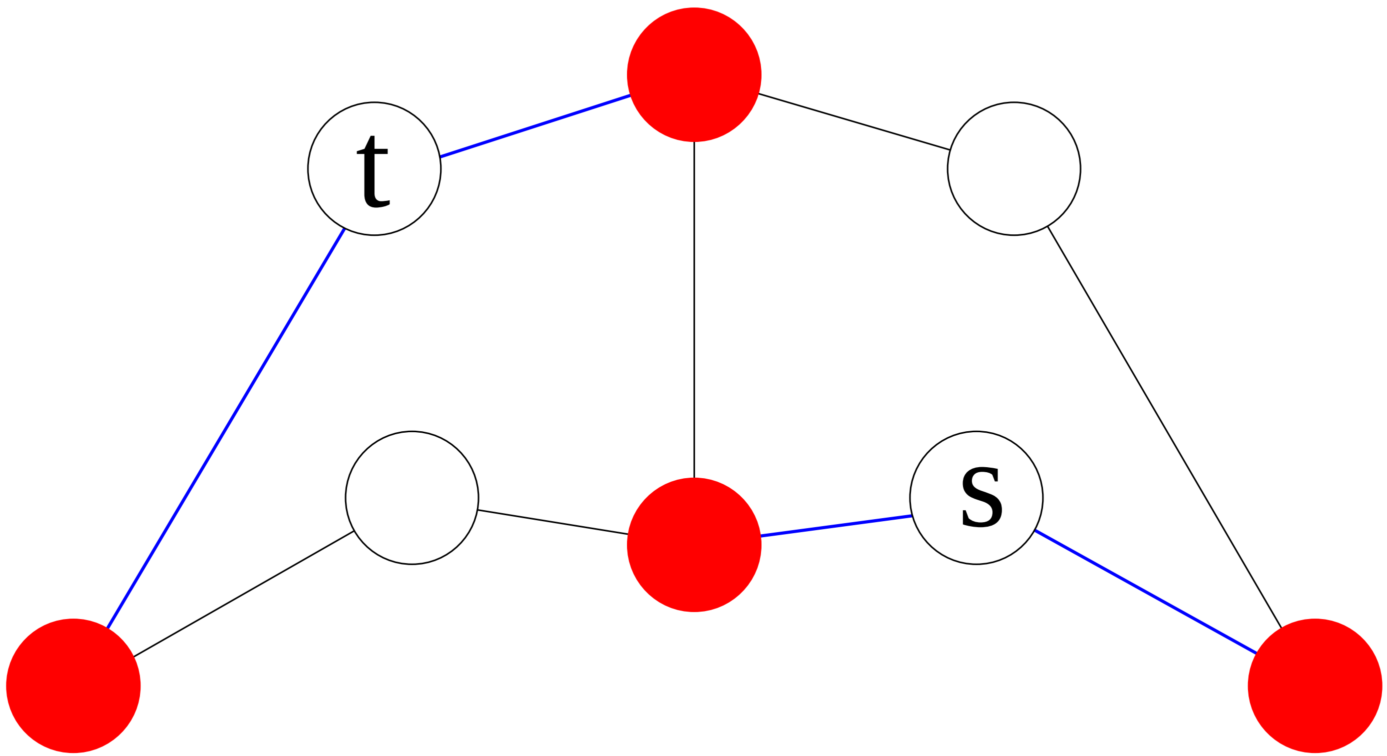


- overlay graph $G' := (S, E')$ where

$E' := \{(s,t) \in S \times S \mid$ no inner node of the shortest s-t-path belongs to $S\}$

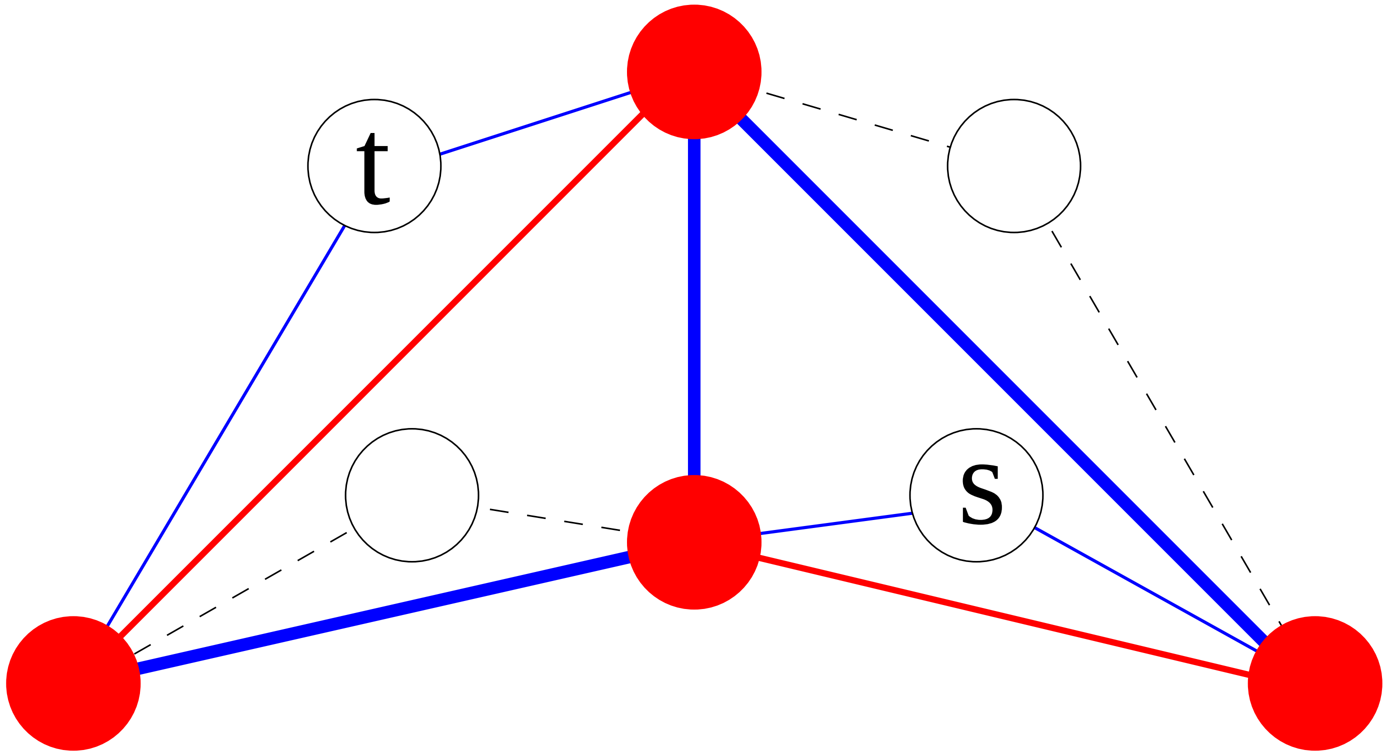
Query

- bidirectional
- perform search in G till search trees are covered by nodes in S



Query

- bidirectional
- perform search in G till search trees are covered by nodes in S
- continue search only in G'



Static Highway-Node Routing

- extend ideas from
  - multi-level overlay graphs [HolzerSchulzWagnerWeiheZaroliagis00–07]
  - highway hierarchies [SS05–06]
  - transit node routing [BastFunkeMatijevicSS06–07]
- use highway hierarchies to classify nodes by 'importance'

  i.e., select node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$

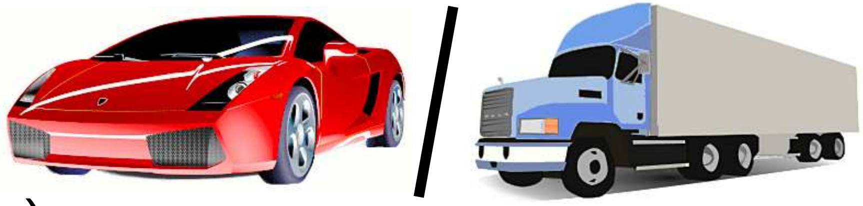
  (crucial distinction from previous separator-based approach)
- construct multi-level overlay graph
- perform multi-level query

Static Highway-Node Routing

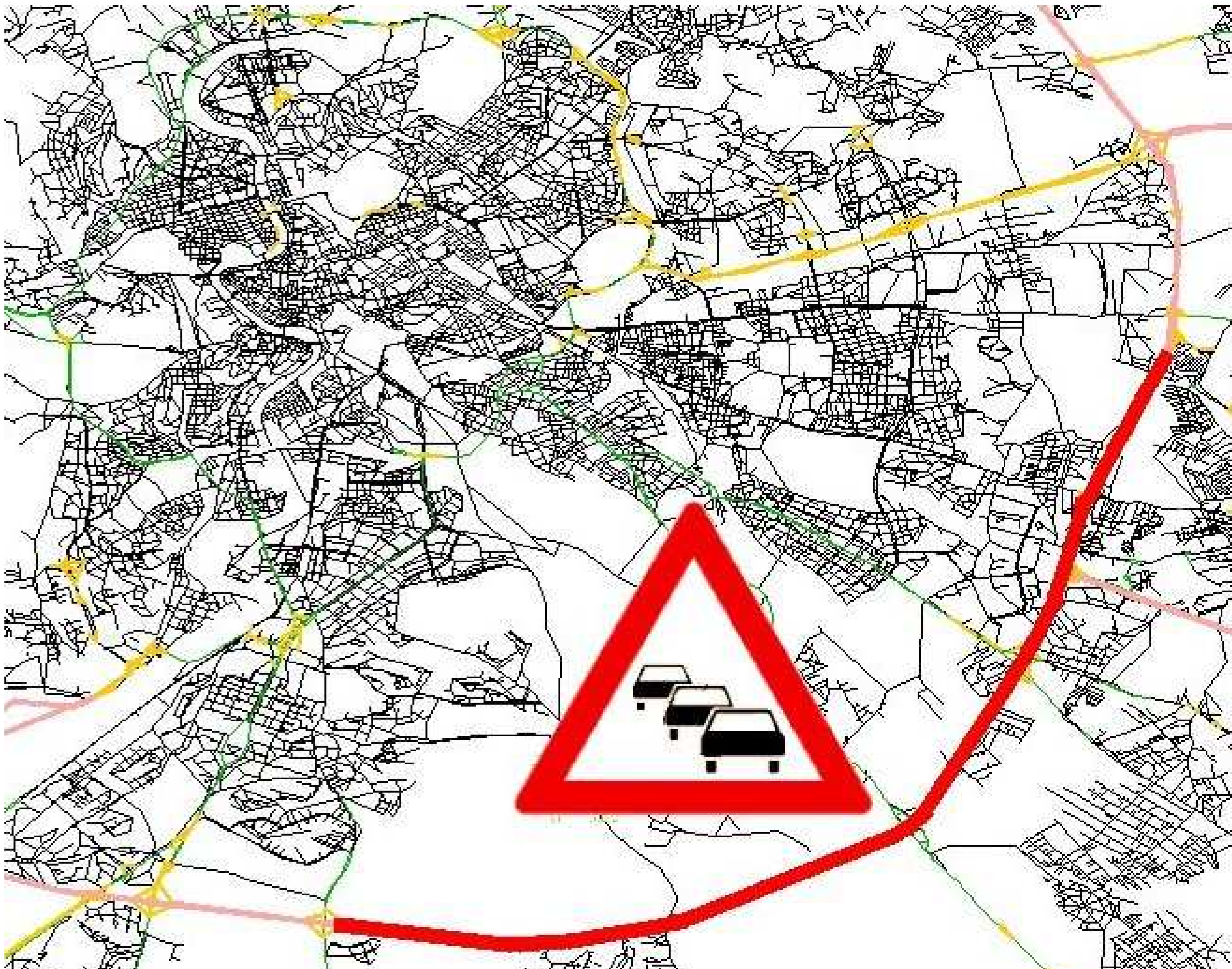
- extend ideas from
  - multi-level overlay graphs [HolzerSchulzWagnerWeiheZaroliagis00–07]
  - highway hierarchies [SS05–06]
  - transit node routing [BastFunkeMatijevicSS06–07]
- use highway hierarchies to classify nodes by ‘importance’
  i.e., select node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$ 16 min
  (crucial distinction from previous separator-based approach)
- construct multi-level overlay graph 3 min, 8 bytes/node
- perform multi-level query 1.1 ms

(experiments with a European road network with ≈ 18 million nodes)

Dynamic Scenarios



- change entire cost function
  (e.g., use different speed profile)



- change a few edge weights
  (e.g., due to a traffic jam)

Dynamic Highway-Node Routing

change entire cost function

- keep the node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$
- recompute the overlay graphs

speed profile	default	fast car	slow car	slow truck	distance
constr. [min]	1:40	1:41	1:39	1:36	3:56
query [ms]	1.17	1.20	1.28	1.50	35.62
#settled nodes	1 414	1 444	1 507	1 667	7 057

Dynamic Highway-Node Routing

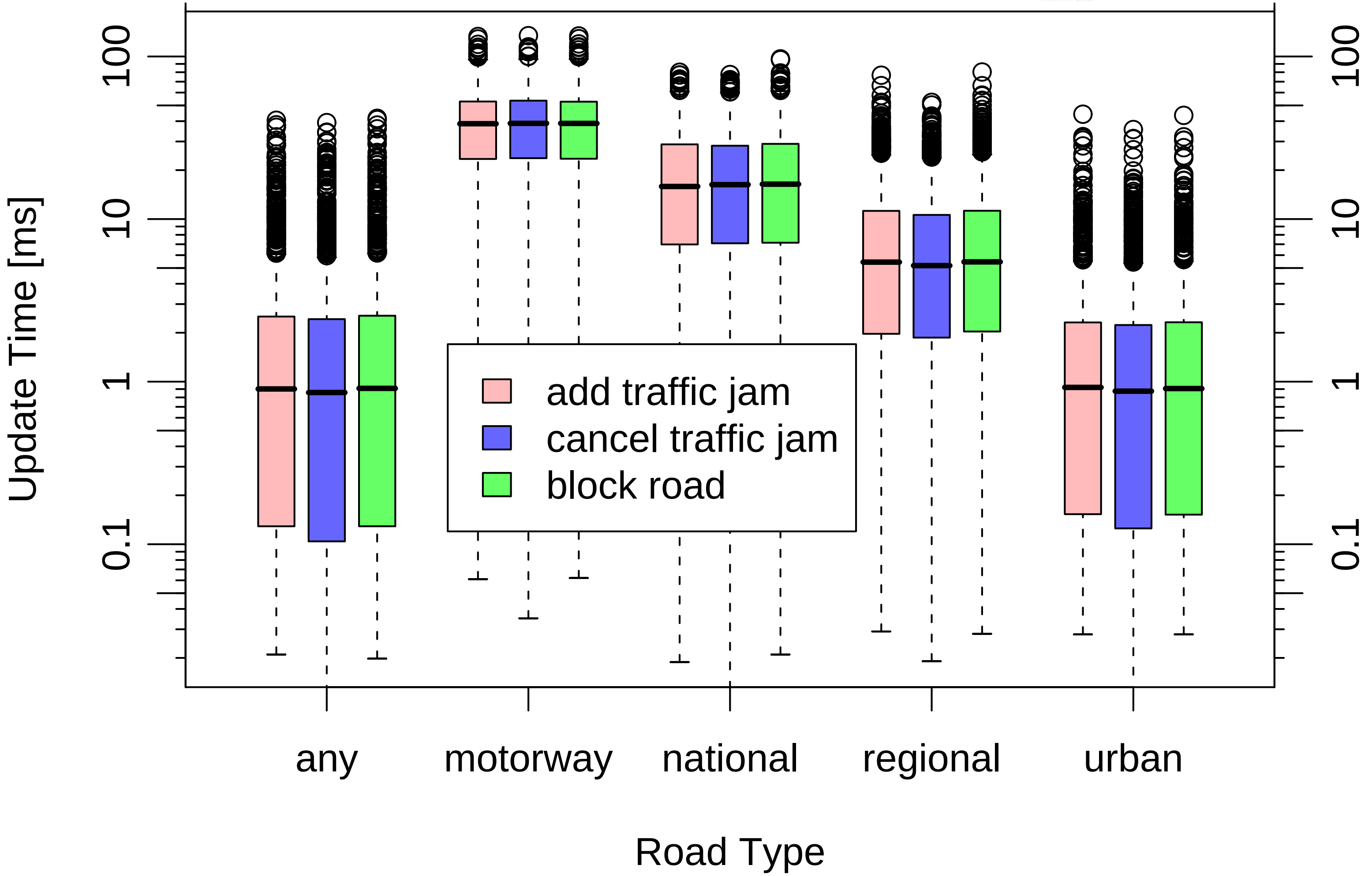
change a few edge weights

- **server scenario:** if something changes,
  - update the preprocessed data structures
  - answer many subsequent queries very fast

- **mobile scenario:** if something changes,
  - it does not pay to update the data structures
  - perform single ‘prudent’ query that takes changed situation into account

Dynamic Highway-Node Routing

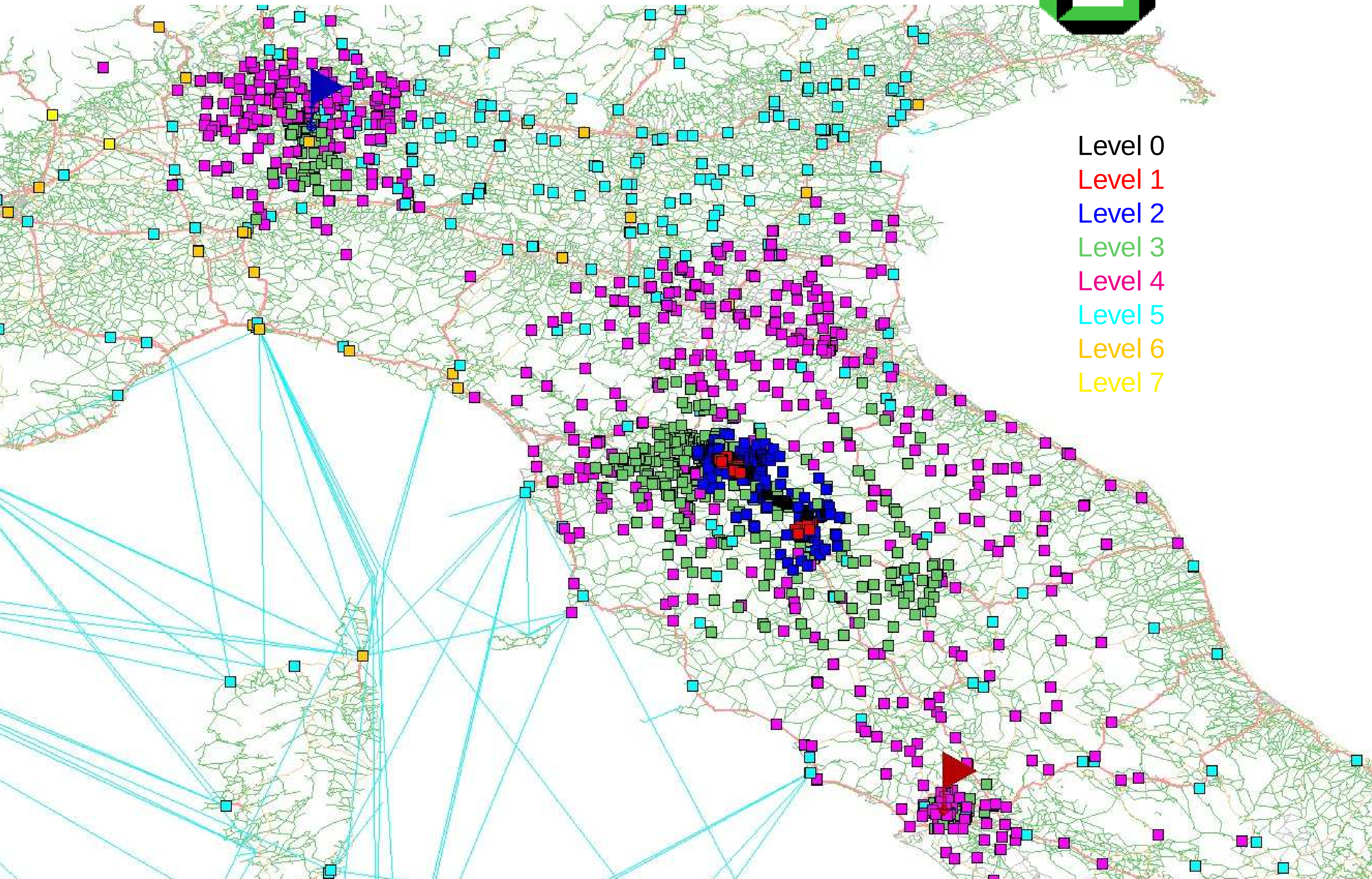
change a few edge weights, server scenario



Dynamic Highway-Node Routing

change a few edge weights, mobile scenario

- keep the node sets $S_1 \supseteq S_2 \supseteq S_3 \ldots$
- keep the overlay graphs
- use auxiliary data to determine for each node u a reliable level $r(u)$
- during a query, at node u
  - do not use edges that have been created in some level $> r(u)$
  - instead, downgrade the search to level $r(u)$


Level 0
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7

Summary

Highway Hierarchies: fast queries, fast preprocessing, low space,

few tuning parameters,

basis for many-to-many and transit-node / highway-node routing.

Many-to-Many: huge distance tables are tractable,

subroutine for transit-node routing.

Transit-Node Routing: fastest routing so far.

Highway-Node Routing: ‘simpler’ highway hierarchies,

fast queries, very low space,

efficiently dynamizable.