Engineering Route Planning Algorithms

Peter Sanders **Dominik Schultes**

Institut für Theoretische Informatik – Algorithmik II

Universität Karlsruhe (TH)

in cooperation with

Holger Bast, Daniel Delling, Stefan Funke, Sebastian Knopp, Domagoj Matijevic,

Frank Schulz, Dorothea Wagner

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

Oberwolfach, May 2007

Outline

Second Part: Highlighting Aspects of

Algorithm Engineering

Engineering

Route Planning **Algorithms**

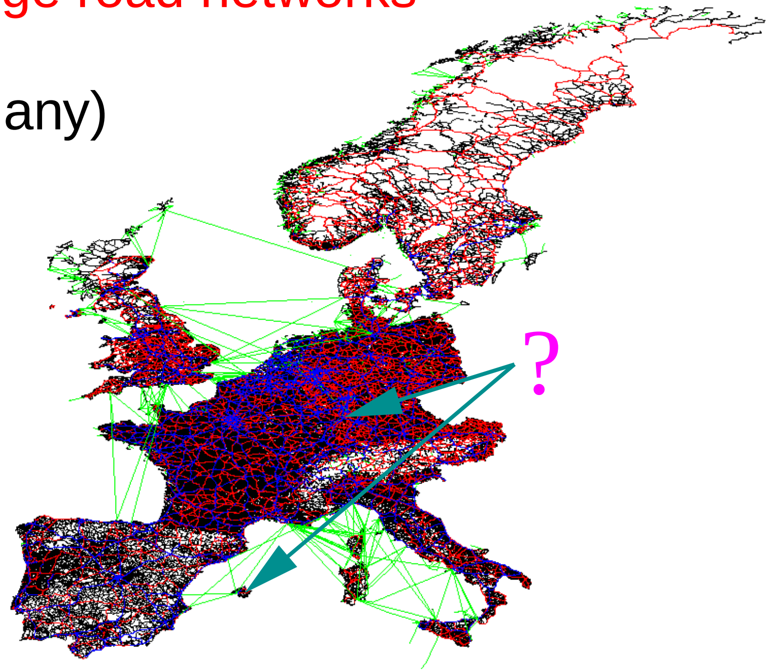
First Part: Overview on our

Route Planning Techniques

Route Planning

Goals:

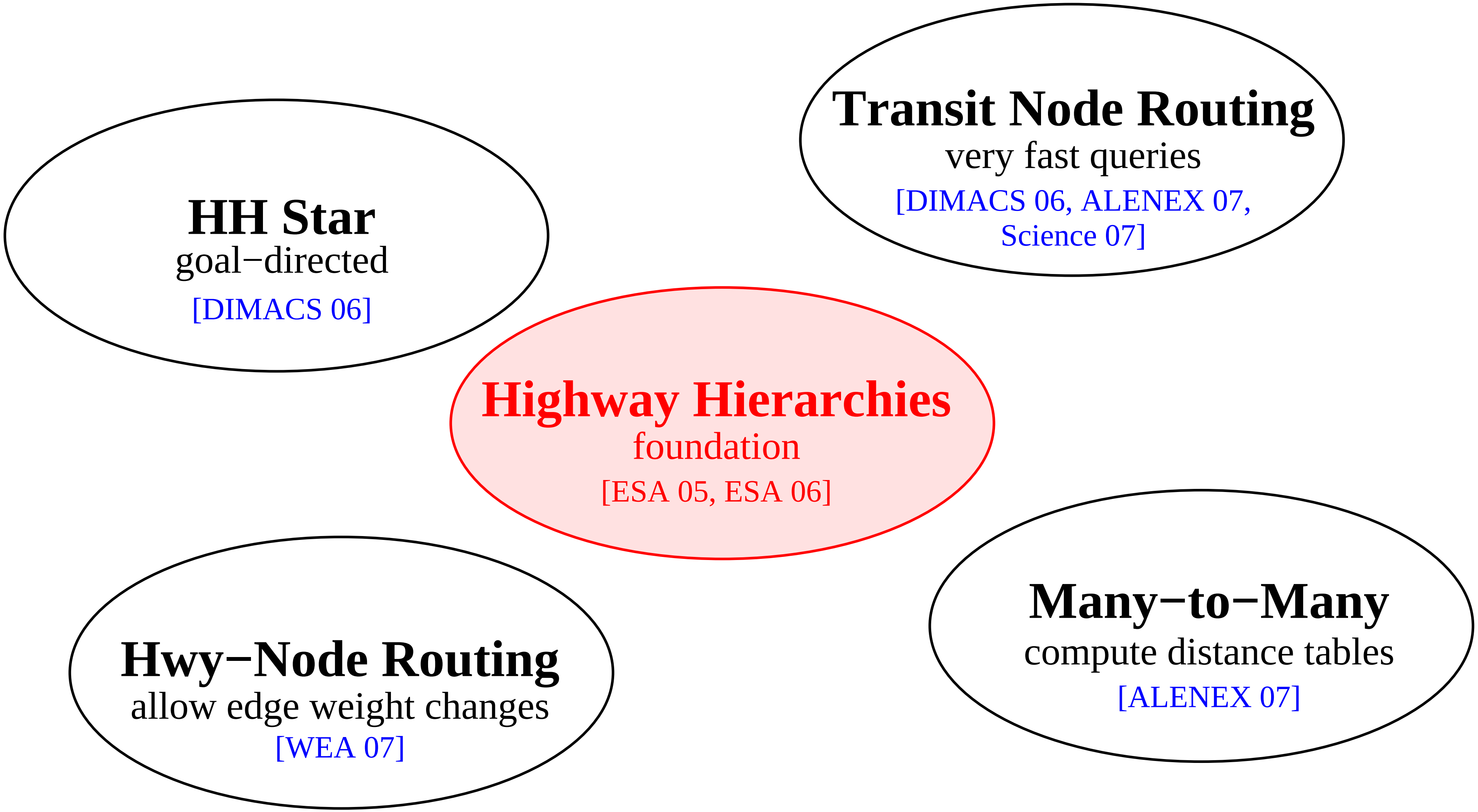
- exact shortest (i.e. fastest) paths in large road networks
- fast queries (point-to-point, many-to-many)
- fast preprocessing
- low space consumption
- fast update operations



Applications:

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

Overview

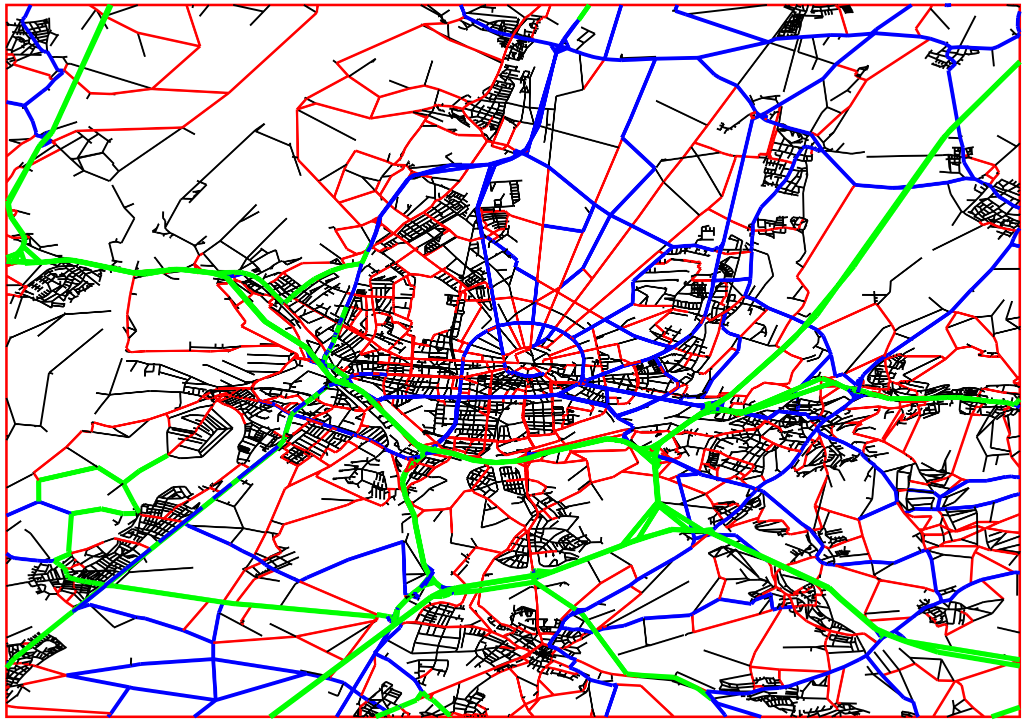


Transit Node Routing
very fast queries
[DIMACS 06, ALENEX 07, Science 07]
HH Star
goal−directed
[DIMACS 06]
Highway Hierarchies
foundation
[ESA 05, ESA 06]
Many−to−Many
compute distance tables
[ALENEX 07]
Hwy−Node Routing
allow edge weight changes
[WEA 07]

Highway Hierarchies

Construction: iteratively alternate between

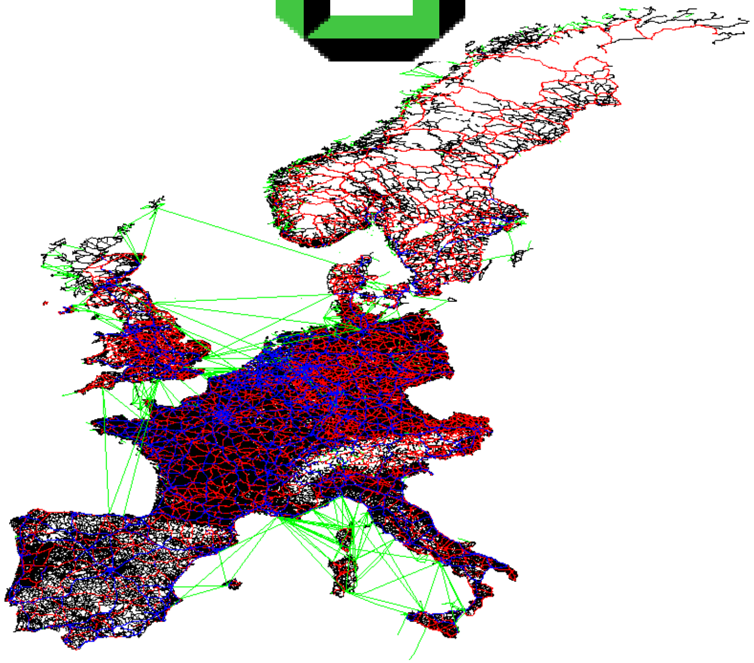
- removal of low degree nodes
- removal of edges that only appear on shortest paths close to source or target



yields a hierarchy of highway networks

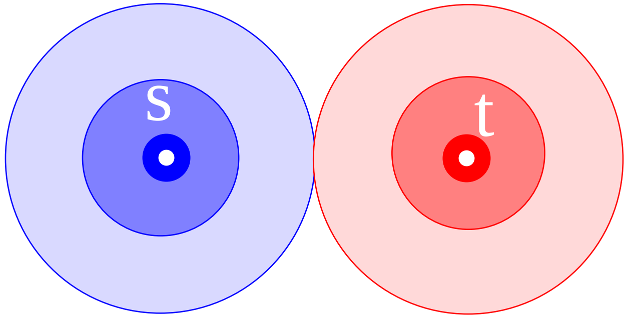
in a sense, classify roads / junctions by 'importance'

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)

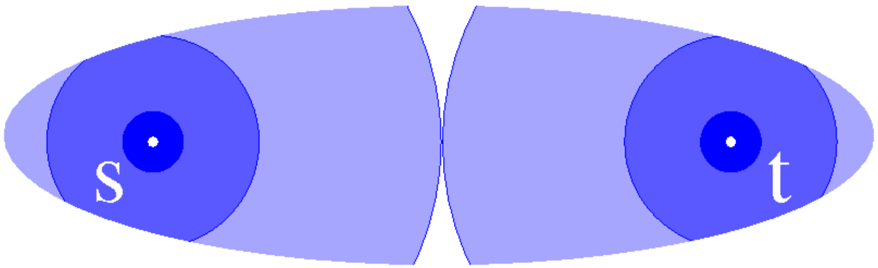
  can be reduced to 17 bytes/node



Highway Hierarchies Star

[joint work with D. Delling, D. Wagner]

- 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



Transit Node Routing

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

First Observation:

For long-distance travel: leave current location

via one of only a few 'important' traffic junctions,

called **access points** [in Europe $\approx$ 10]

($\rightsquigarrow$ we can afford to store all access points for each node)

Second Observation:

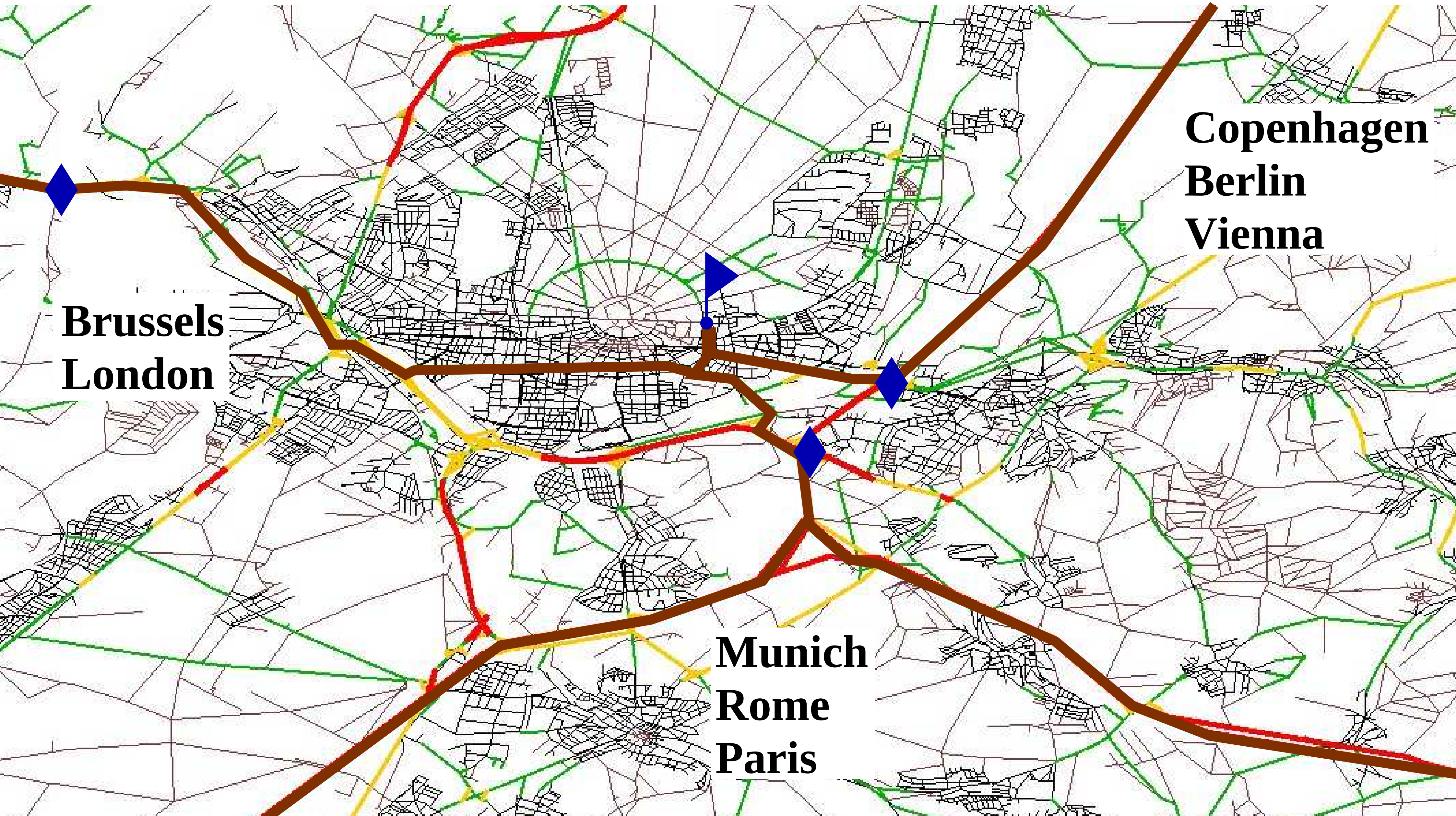
Each access point is relevant for several nodes. $\rightsquigarrow$

union of the access points of all nodes is small,

called **transit node set** [in Europe $\approx$ 10 000]

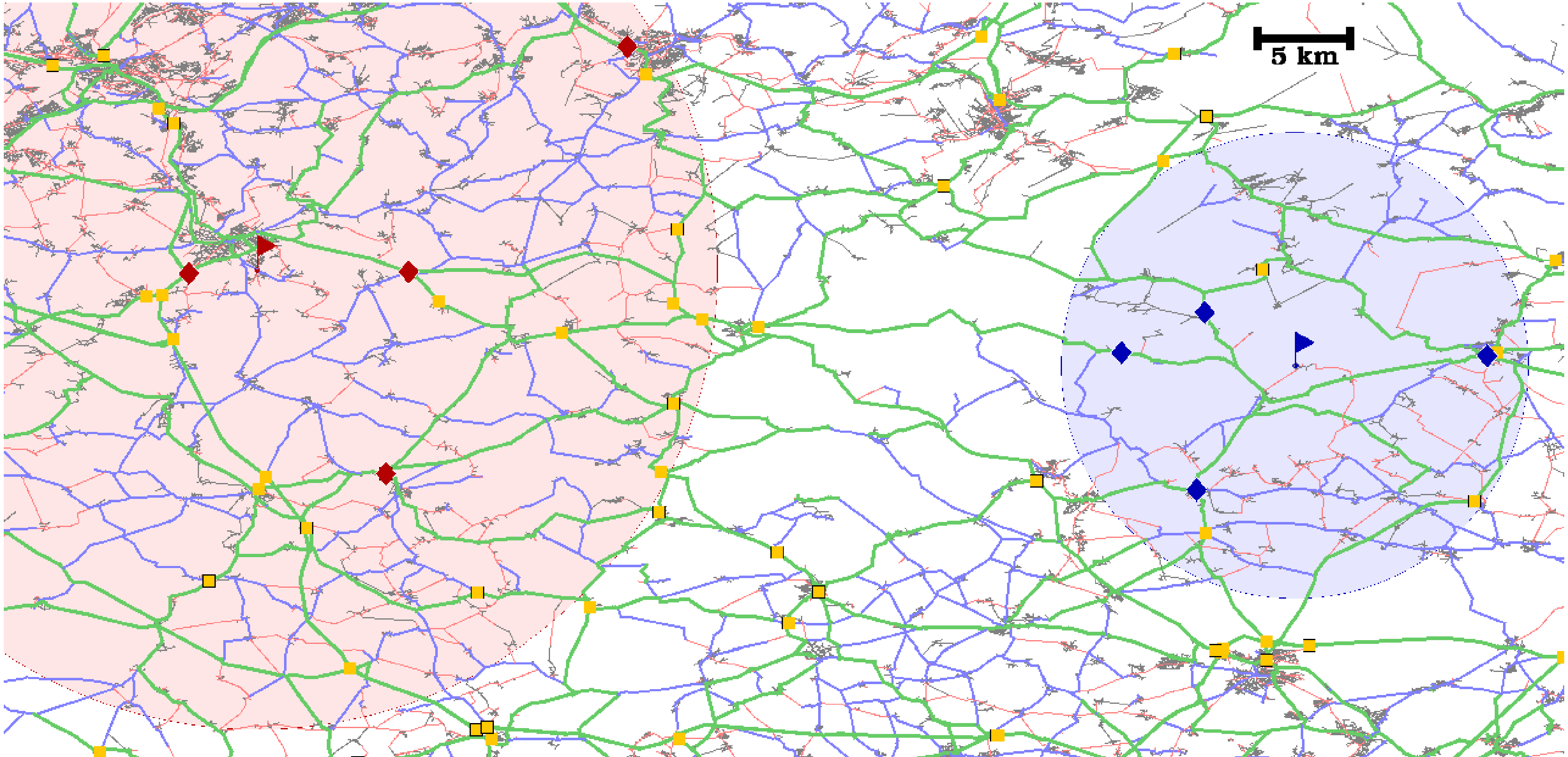
($\rightsquigarrow$ we can afford to store the distances between all transit node pairs)

Transit Node Routing



Transit Node Routing

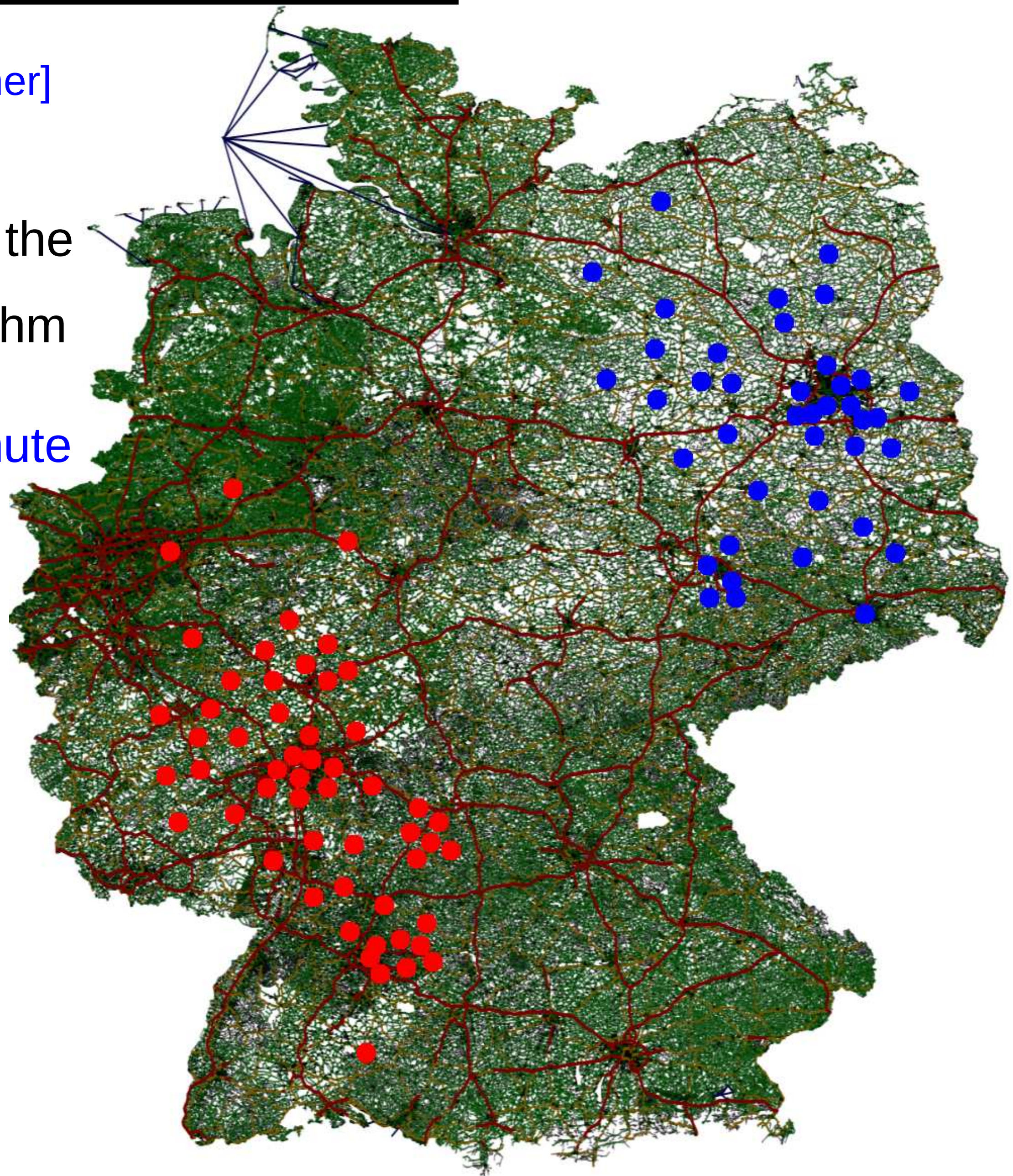
- uses highway hierarchies to classify nodes by 'importance'
- *very* fast queries (down to 6 μs, 1 000 000 times faster than DIJKSTRA)
- more preprocessing time (2:44 h) and space (251 bytes/node) needed



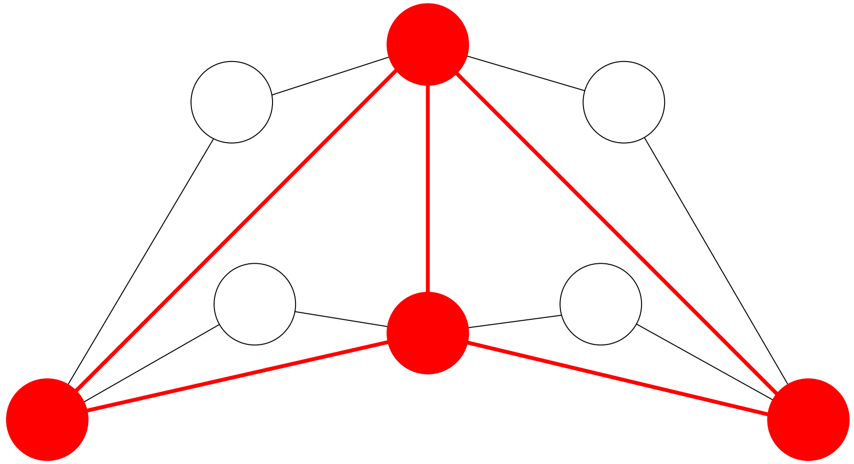
Many-to-Many Shortest Paths

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

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



Static Highway-Node Routing



- extend ideas from
  - multi-level overlay graphs
  - highway hierarchies
  - transit node routing
- uses highway hierarchies to classify nodes by ‘importance’
- preprocessing: 19 min
- memory overhead: 8 bytes/node
- query time: 1.1 ms

Dynamic Highway-Node Routing

- change entire cost function

  typically < 2 minutes

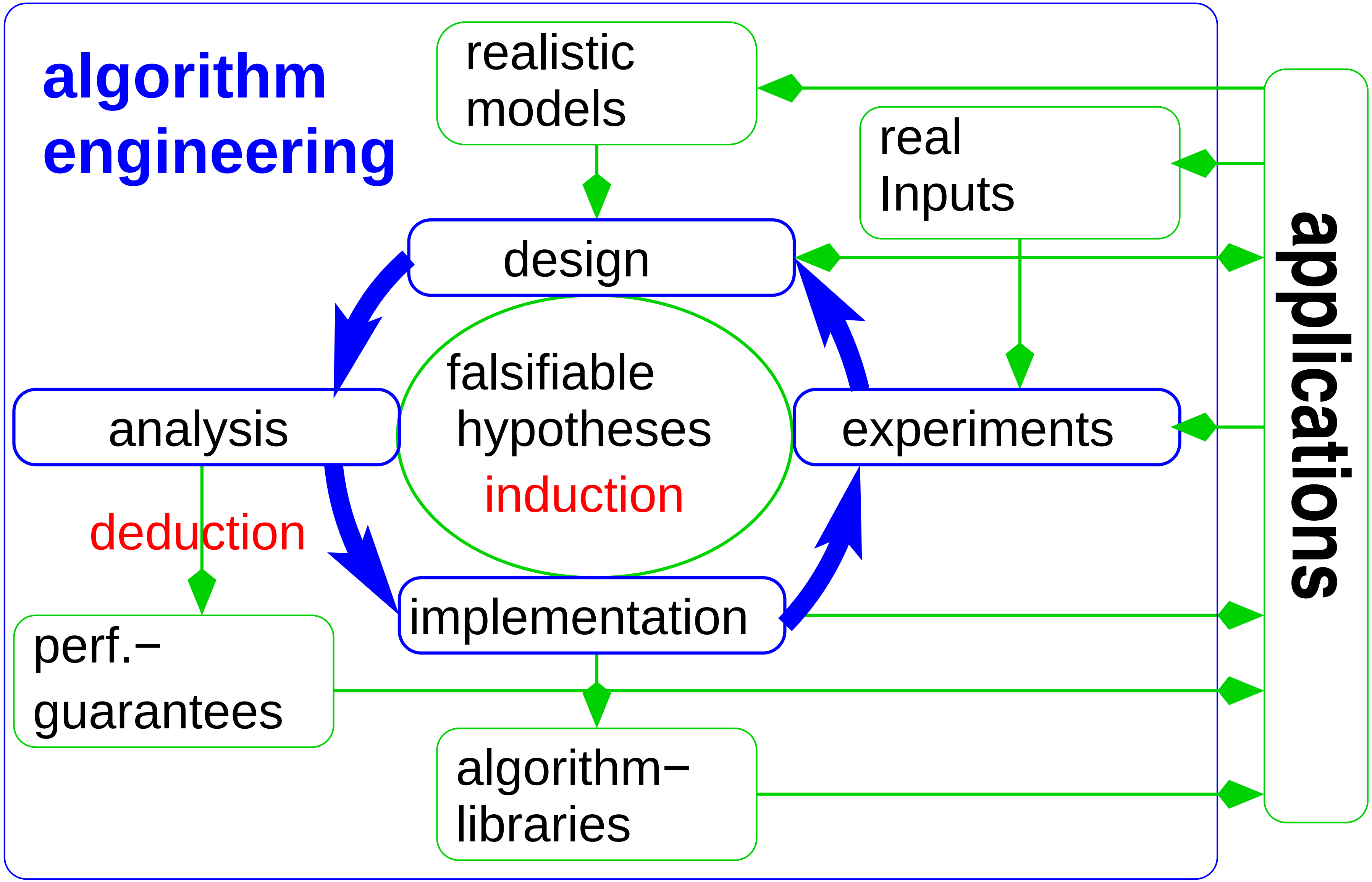
- change a few edge weights
  - update data structures

    2 – 40 ms per changed edge

  OR

  - perform prudent query

    e.g., 47.5 ms if 100 motorway edges have been changed

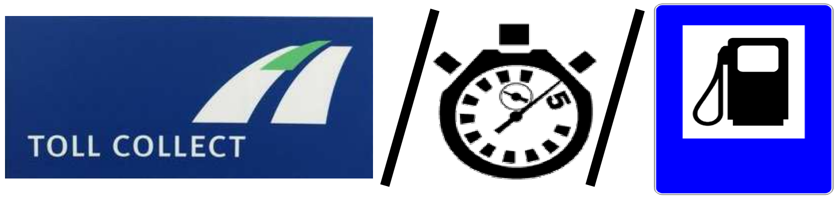

algorithm
engineering
realistic
models
real
Inputs
design
falsifiable
hypotheses
induction
analysis
experiments
deduction
implementation
perf.−
guarantees
algorithm−
libraries
applications

Models

Application:

- structure of a road network is ('almost') static
  $\rightsquigarrow$ allow preprocessing
- edge weights may change unexpectedly

- time-dependent edge weights
- point-to-point, many-to-many
- multi-objective



Machine:

- memory hierarchy
- parallel

Analysis

Correctness:

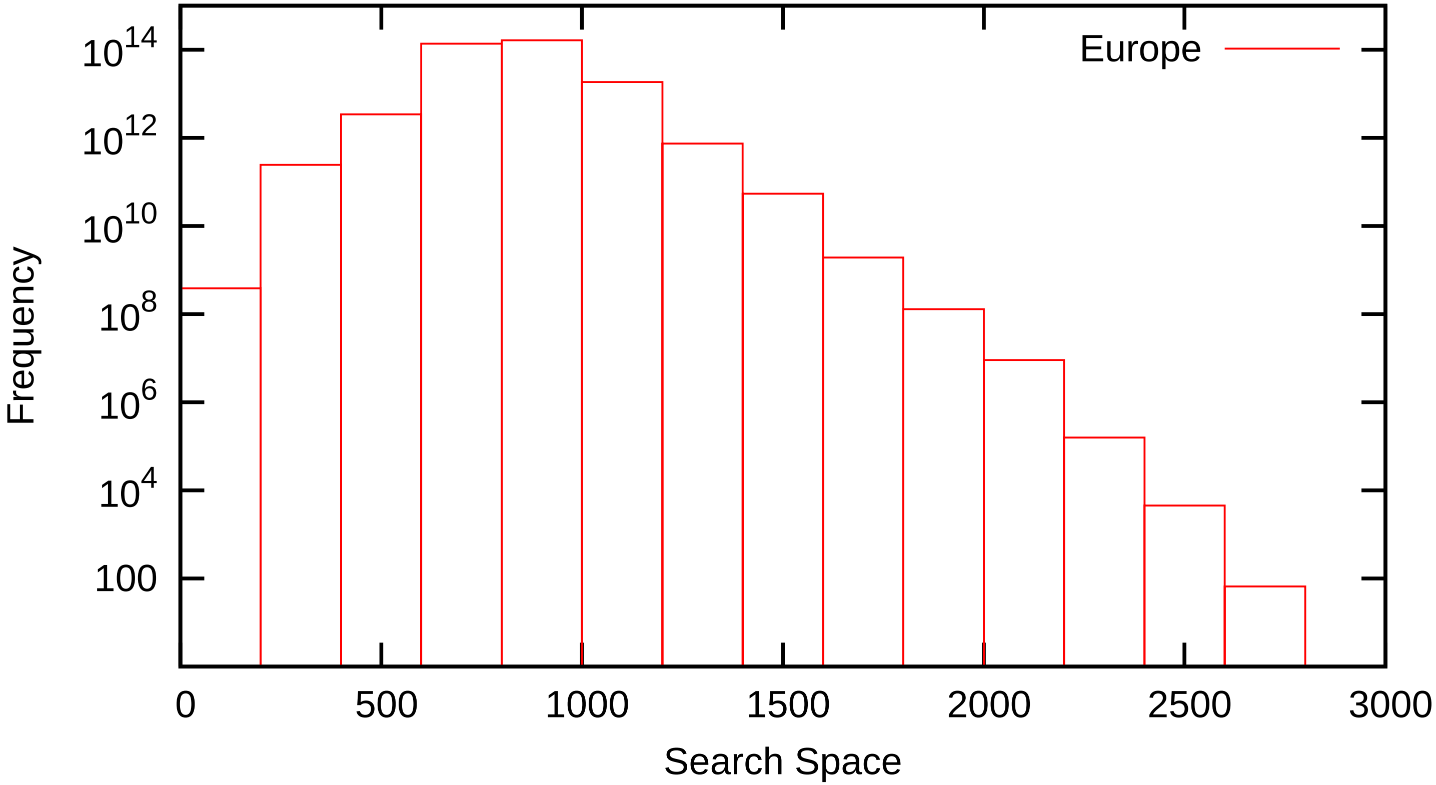
- for TNR and HNR: probably not too difficult
- for HH: surprisingly difficult (ambigious shortest paths)

Worst-Case Bounds:

- performance relies on ‘certain’ graph properties: specify them
- derive worst-case bounds for graphs with the specified properties

Analysis

Per-Instance Worst-Case Guarantees:



histogram of (upper bounds on) the search space sizes of all possible n^2 queries

can be computed using a linear number of queries

Implementation

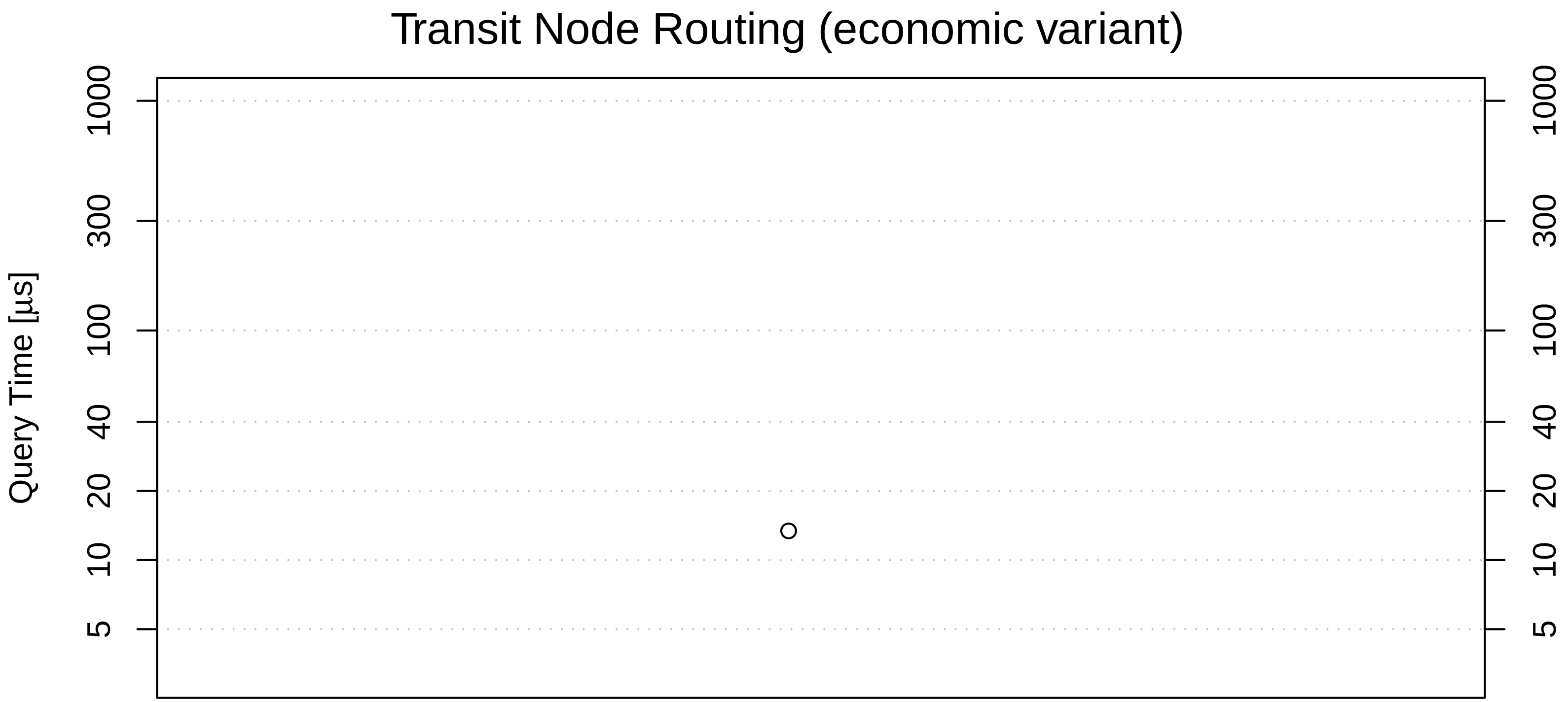
[covers *all* mentioned route planning techniques]

- quite complex (≈ 18 000 lines of code (w/o tools))
- C++ template mechanism

  (currently, 23 different instantiations of our Dijkstra template class)
- standard template library and ‘home-made’ data structures
  - provide only the required functionality
  - can efficiently handle large data sets
- thorough checking: asserts, naive reference implementations
- visualisation

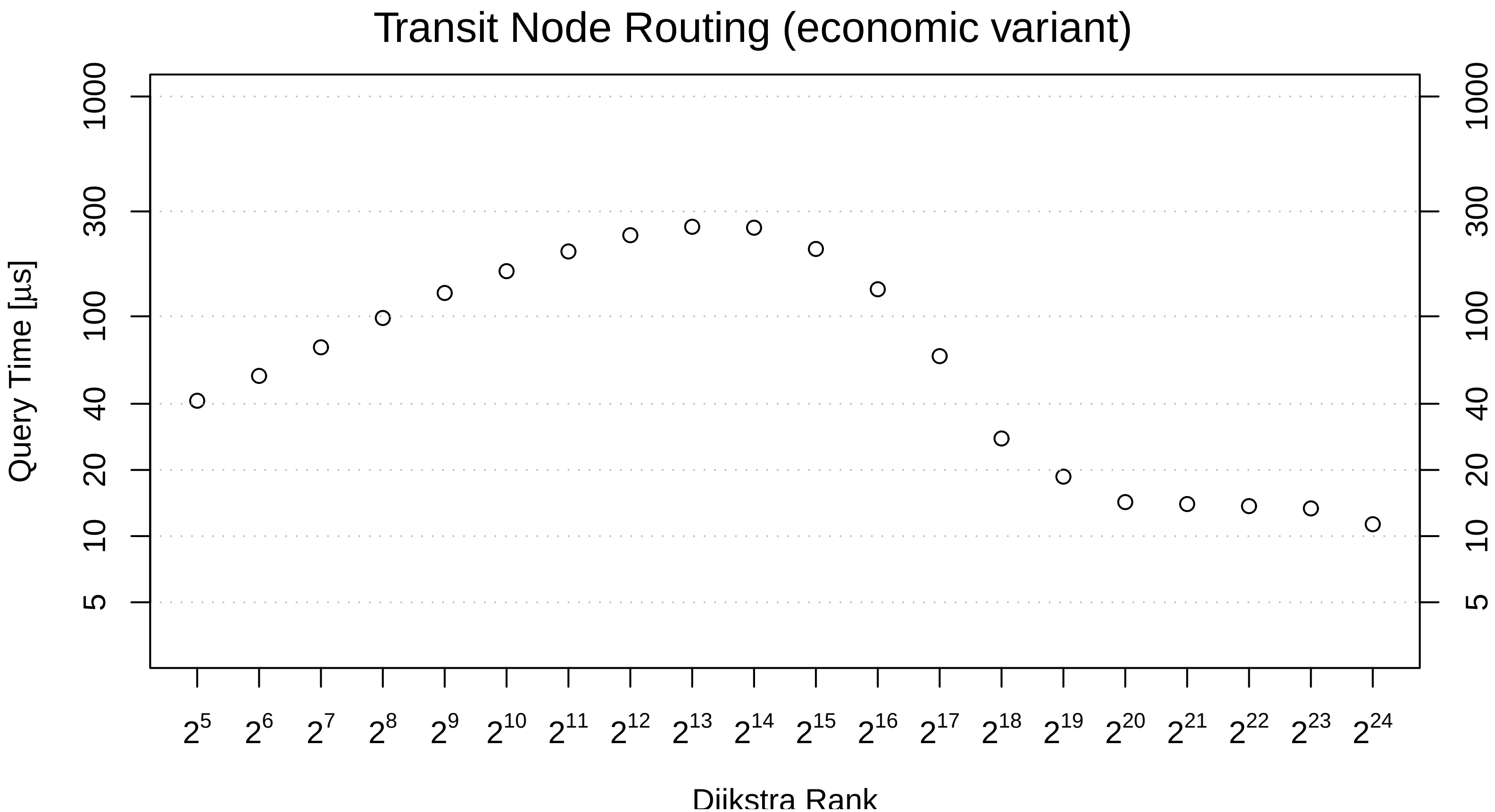
Experiments

- s-t-pairs uniformly at random $\longleftrightarrow$ queries in real applications
- average value $\longleftrightarrow$ variance?



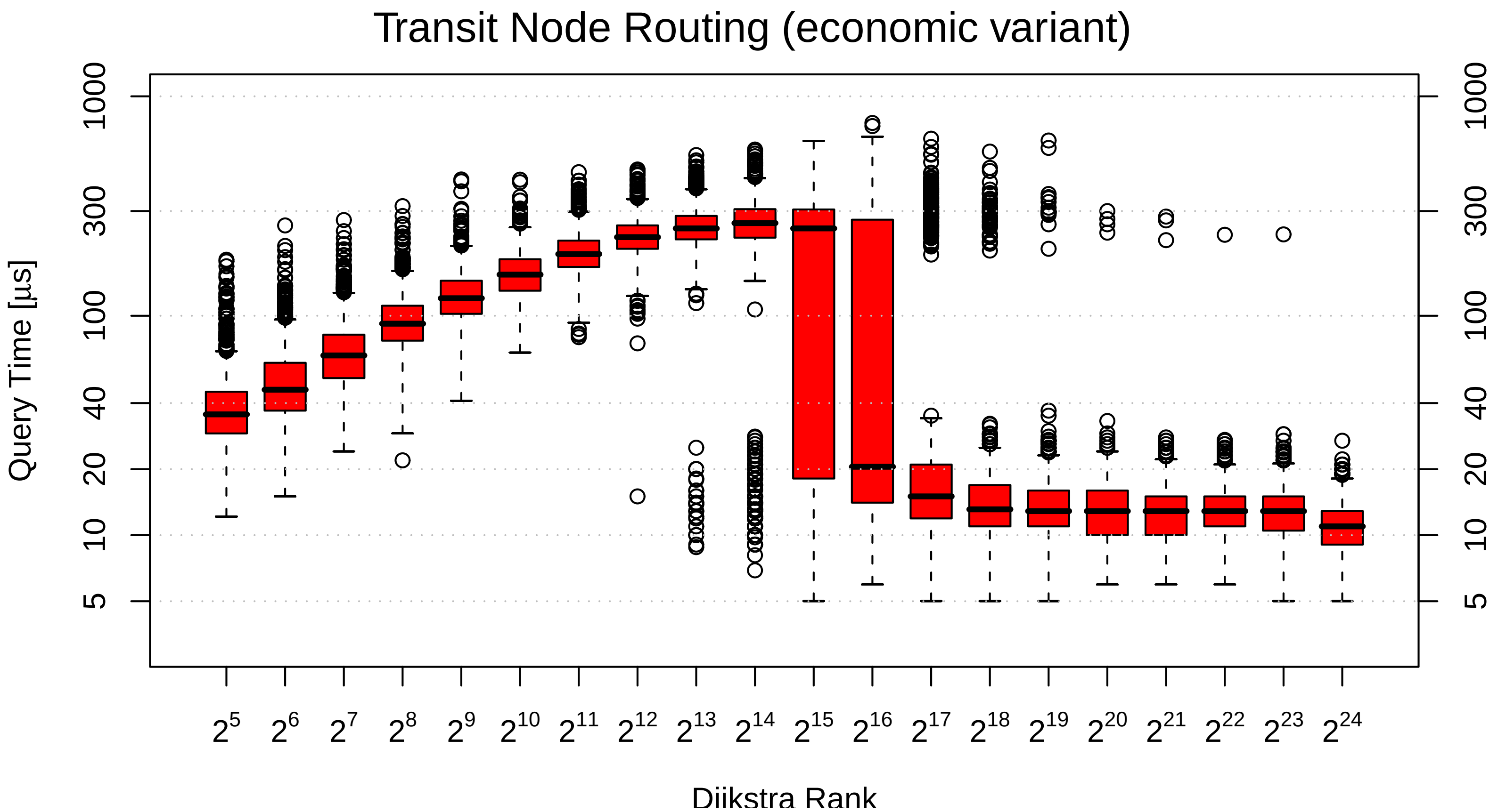
Experiments

- consider different localities!
- average value $\longleftrightarrow$ variance?



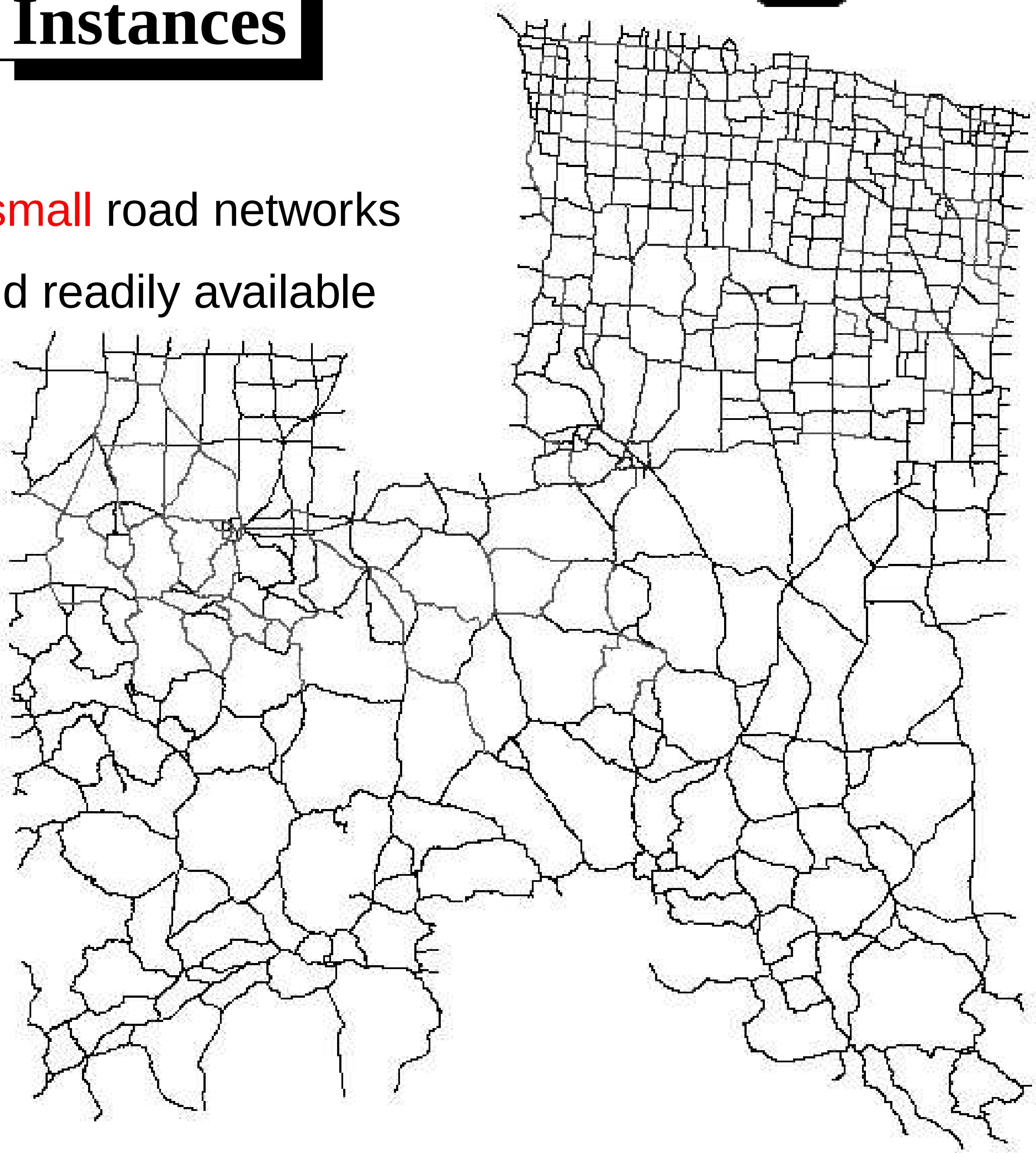
Experiments

- consider different localities!
- plot complete spectrum!



Instances

- before 2005: only very small road networks
  publicly and readily available

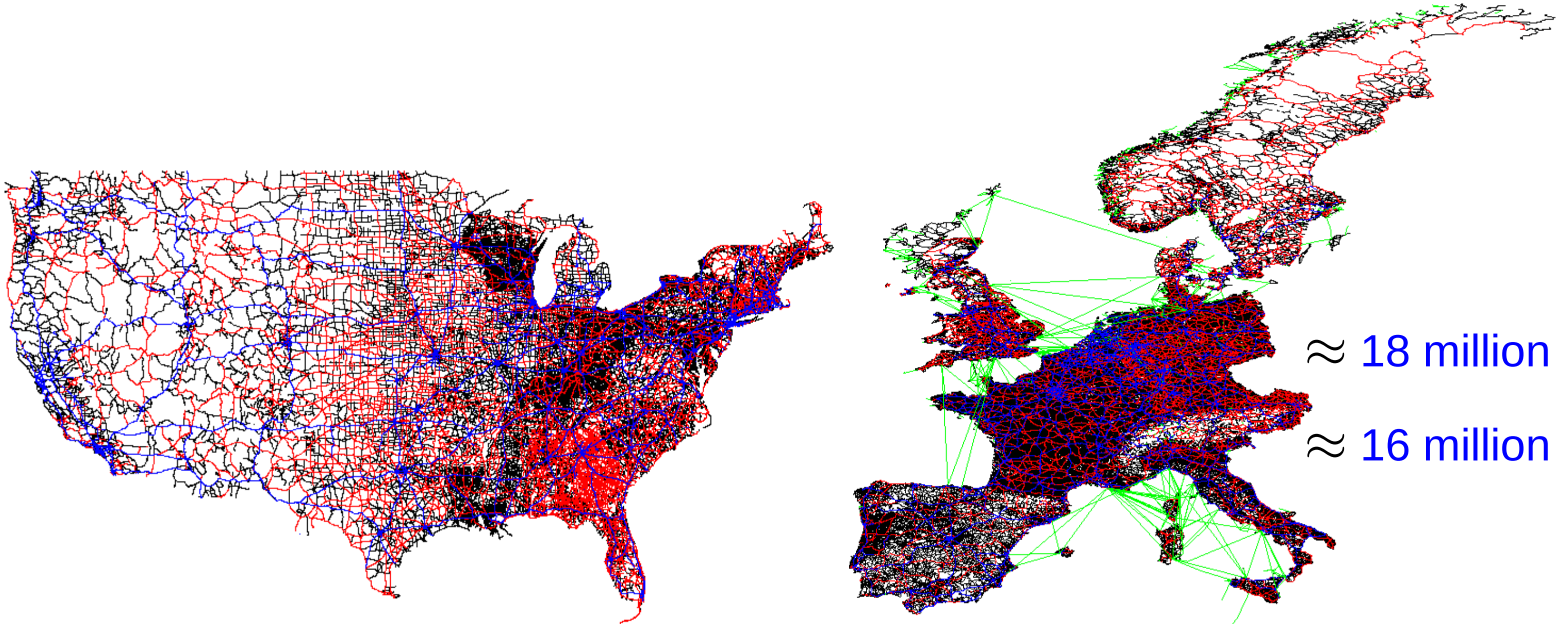


$\approx$ 200 000 nodes, **but** only

$\approx$ 1 000 'degree > 2' nodes

Instances

- in 2005: US and Western European road networks obtained
  - composed from a public source (US Census Bureau)
  - provided by a company (PTV AG) for scientific use



$\approx$ 18 million nodes

$\approx$ 16 million 'degree > 2' nodes

- now: widely spread (e.g., DIMACS Implementation Challenge)

Instances

Open Issues:

- turn penalties

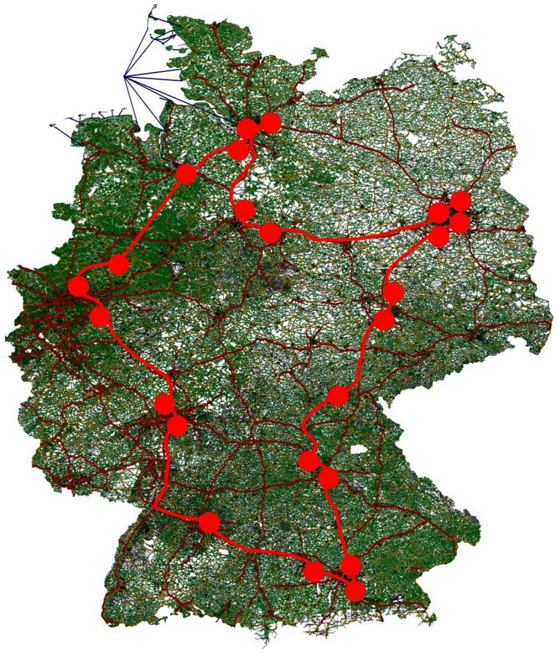
- real source-target pairs

  (we have some many-to-many instances)

- real traffic reports (edge weight changes)
- time-dependent edge weights (not only for motorways!)
- other graph types

Applications

- single point-to-point queries
  - mobile navigation system (built-in, PDA, mobile phone, . . .)
  - internet route planning service
- massive amount of point-to-point queries
  - traffic simulations
- many-to-many queries
  - logistics optimisation
  - ride sharing



promising contacts to various companies—more to come?