Many-to-Many Shortest Paths
Using Highway Hierarchies

Sebastian Knopp **Peter Sanders**

Dominik Schultes **Frank Schulz** **Dorothea Wagner**

Universität Karlsruhe (TH) – Algorithmik I/II

PTV AG, Karlsruhe

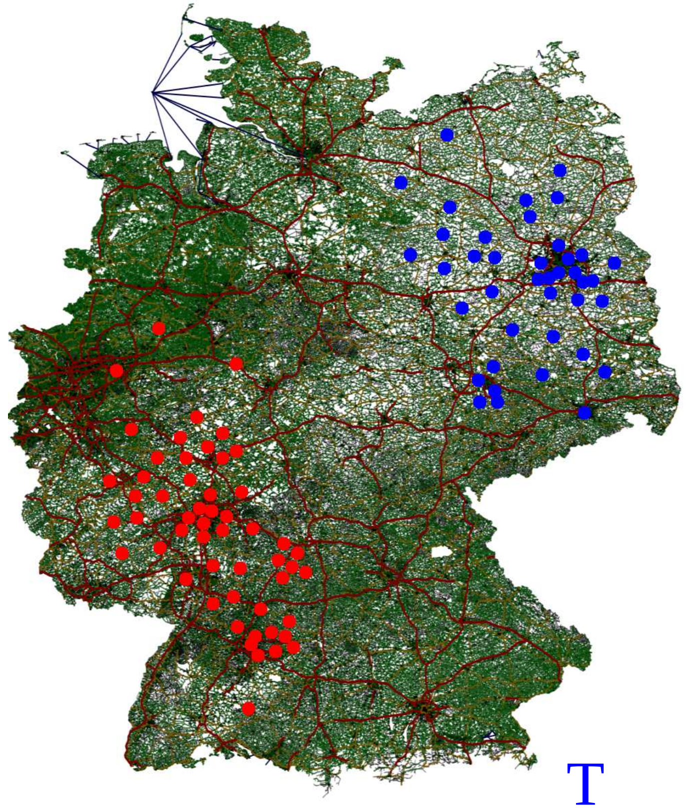
`http://algo2.iti.uka.de/schultes/hwy/`

New Orleans, January 6, 2007

Many-to-Many Shortest Path Problem

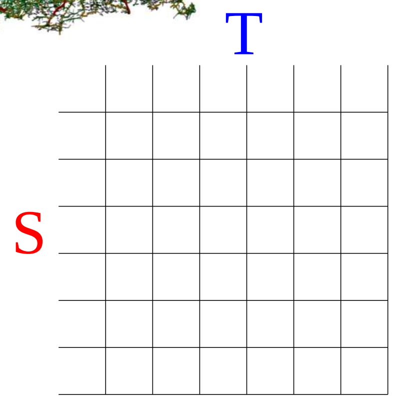
Given:

- graph $G = (V, E)$
- set of source nodes $S \subseteq V$
- set of target nodes $T \subseteq V$



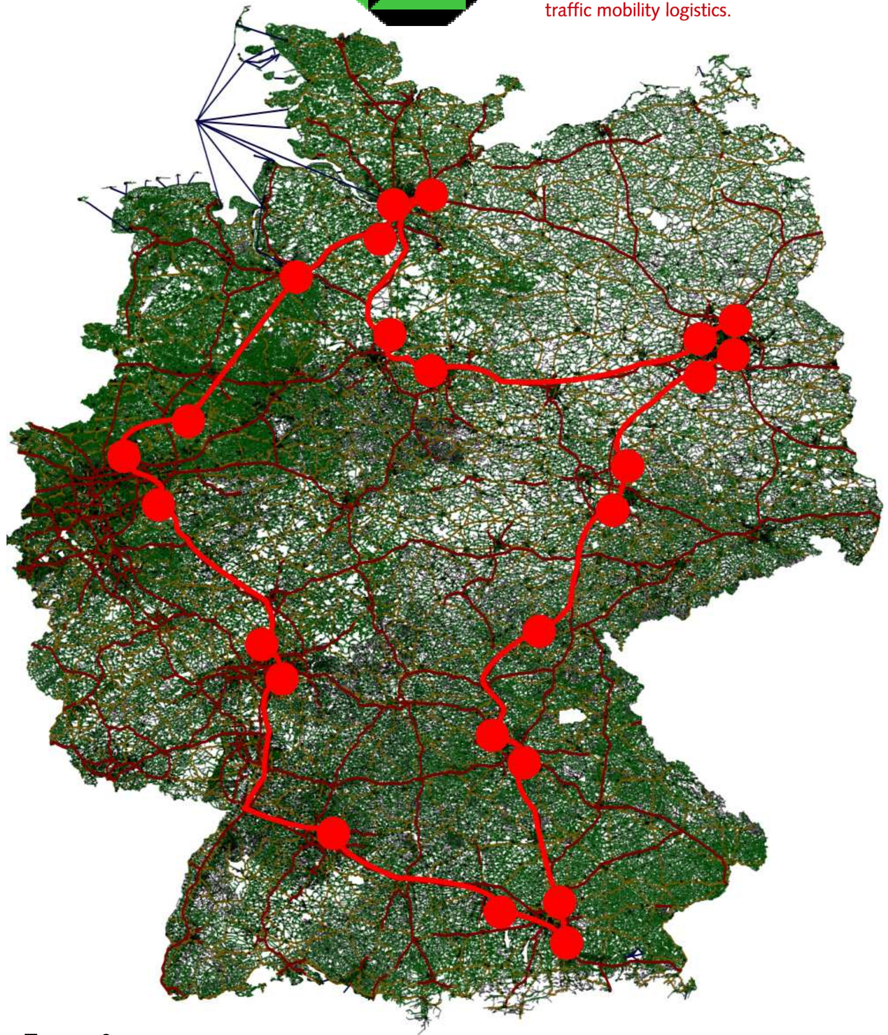
Task: compute $|S| \times |T|$ distance table

containing the shortest path distances



Here: concentrate on road networks

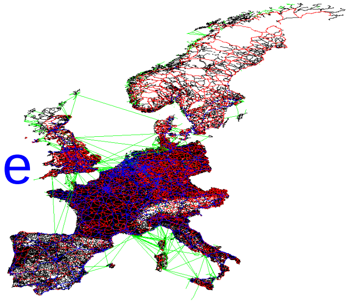
Applications



- **Logistics**
  - vehicle routing problem
  - input for traveling salesman solver
- **Preprocessing for Point-to-Point Techniques**
  - Precomputed Cluster Distances [MaueSandersMatijevic2006]
  - Transit Node Routing [next talk]

Simple Solutions

Example: $10\,000 \times 10\,000$ table
in Western Europe



- apply $\underbrace{\text{SSSP algorithm}}_{\text{(e.g. Dijkstra)}}$ $|S|$ times $\approx 10\,000 \times 10\,\text{s} \approx$ one day

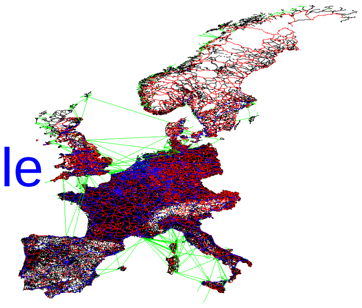
- apply $\underbrace{\text{P2P algorithm}}_{\text{(e.g. highway hierarchies}^{1})}$ $|S| \times |T|$ times $\approx 10\,000^2 \times 1\,\text{ms} \approx$ one day

---

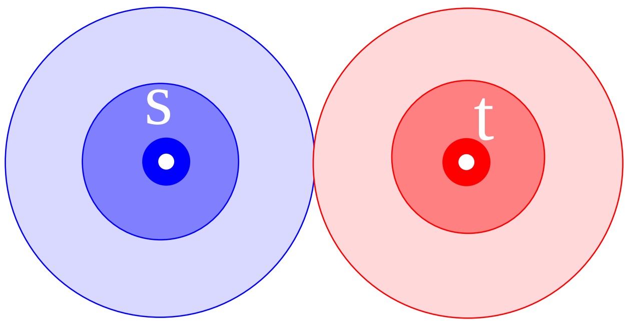
[1] requires about 15 minutes preprocessing time

Our Solution

Example: $10\,000 \times 10\,000$ table in Western Europe



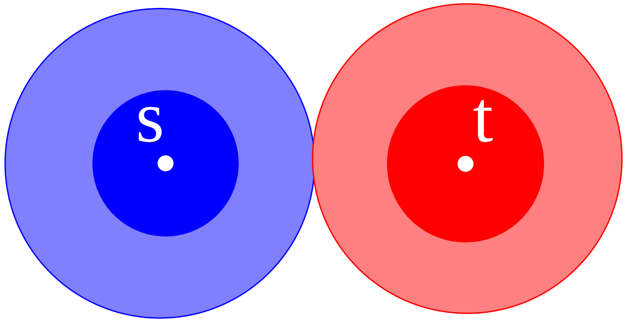
- many-to-many algorithm based on highway hierarchies[1] $\approx$ one minute



[1] requires about 15 minutes preprocessing time

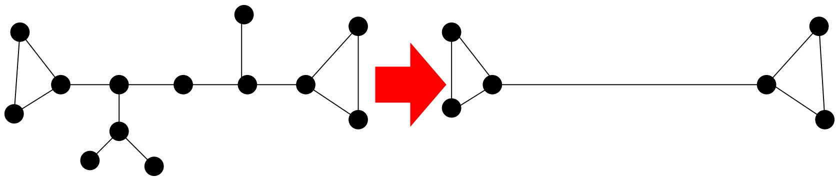
Highway Hierarchies[2]

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

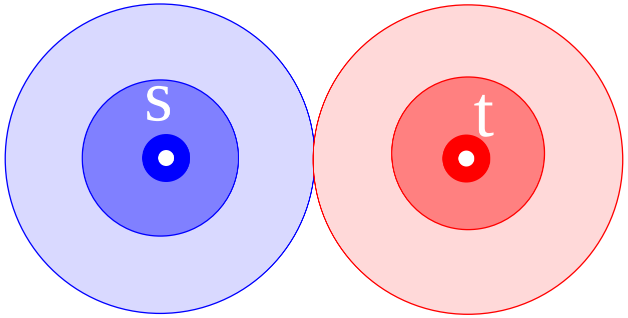


= minimal graph that preserves all shortest paths

- contract network, e.g.,



- iterate ⇝ highway hierarchy



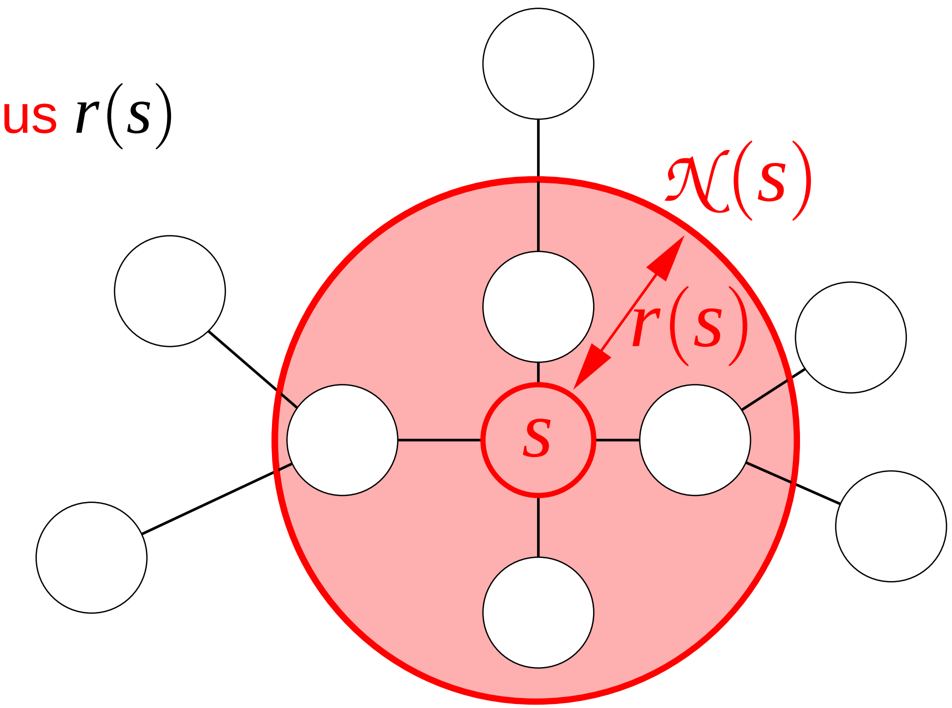
---

[2] presented at ESA 2005 and ESA 2006

Local Area

- choose neighbourhood radius $r(s)$

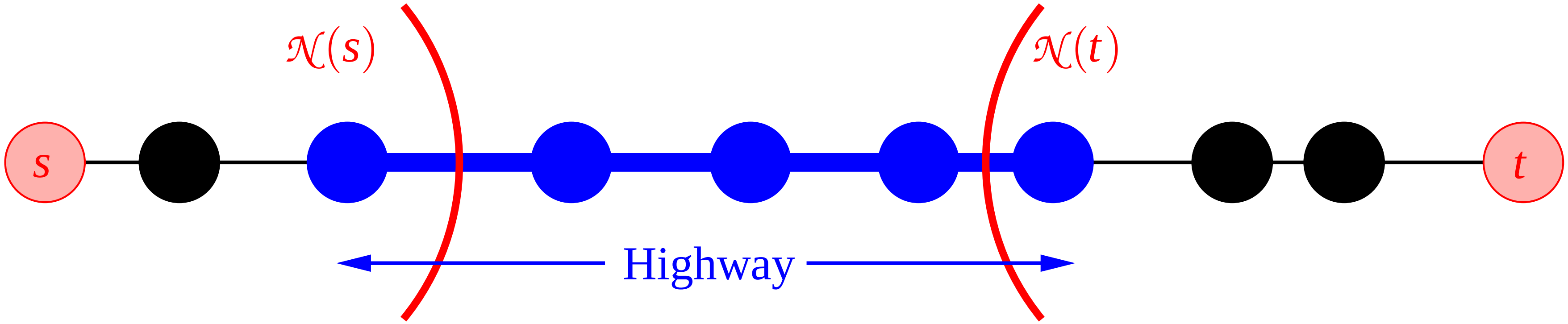
  (by a heuristic)



- define neighbourhood of s

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

Highway Network



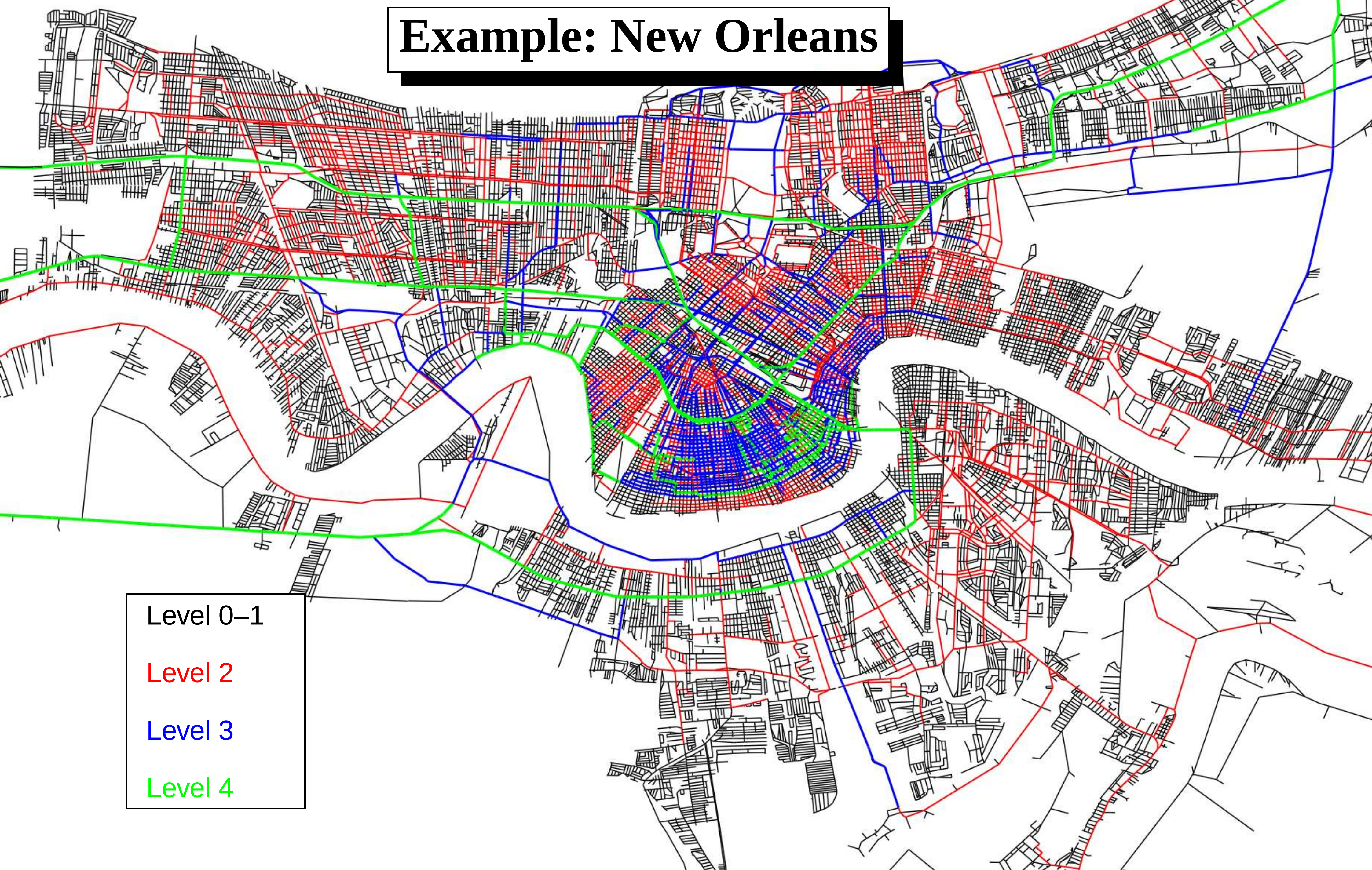
Edge (u,v) belongs to highway network *iff* there are nodes s and t s.t.

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

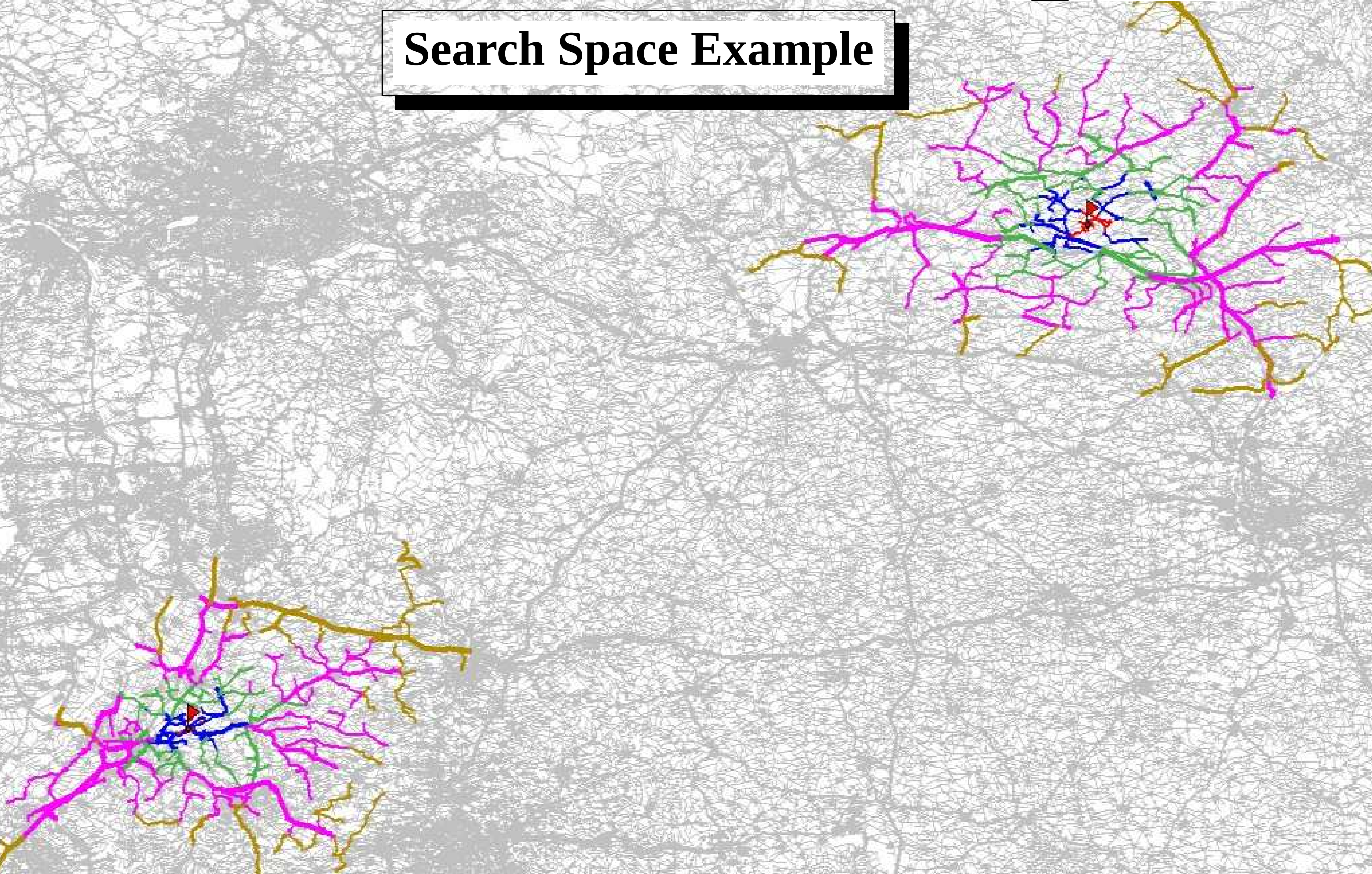
and

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

Example: New Orleans



Search Space Example



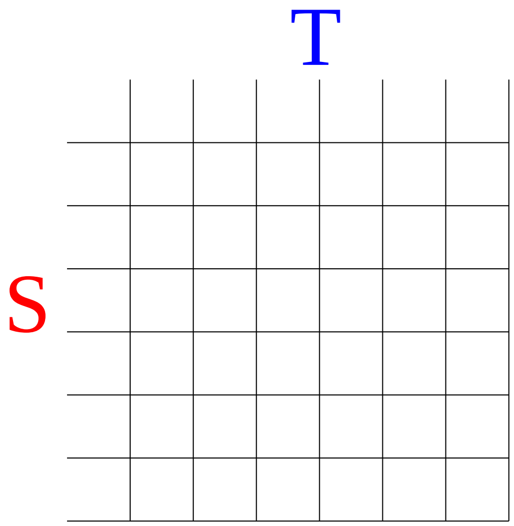
Main Idea

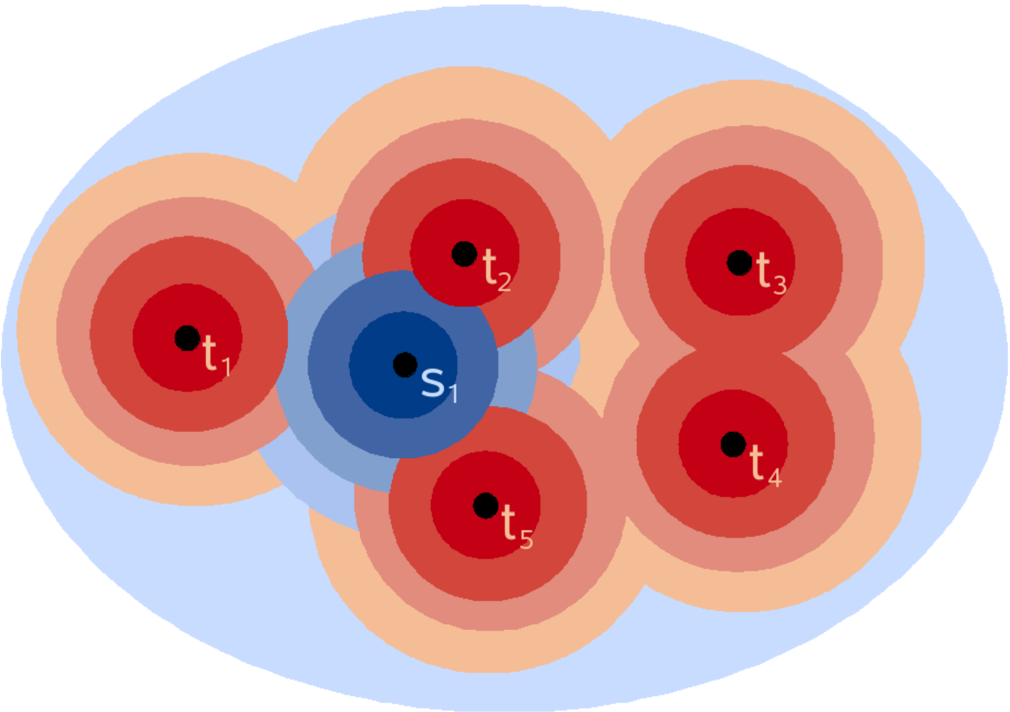
- instead of $|S| \times |T|$ bidirectional highway queries
- perform $|S| + |T|$ unidirectional highway queries

Algorithm

- maintain an $|S| \times |T|$ table D of tentative distances

  (initialize all entries to ∞)



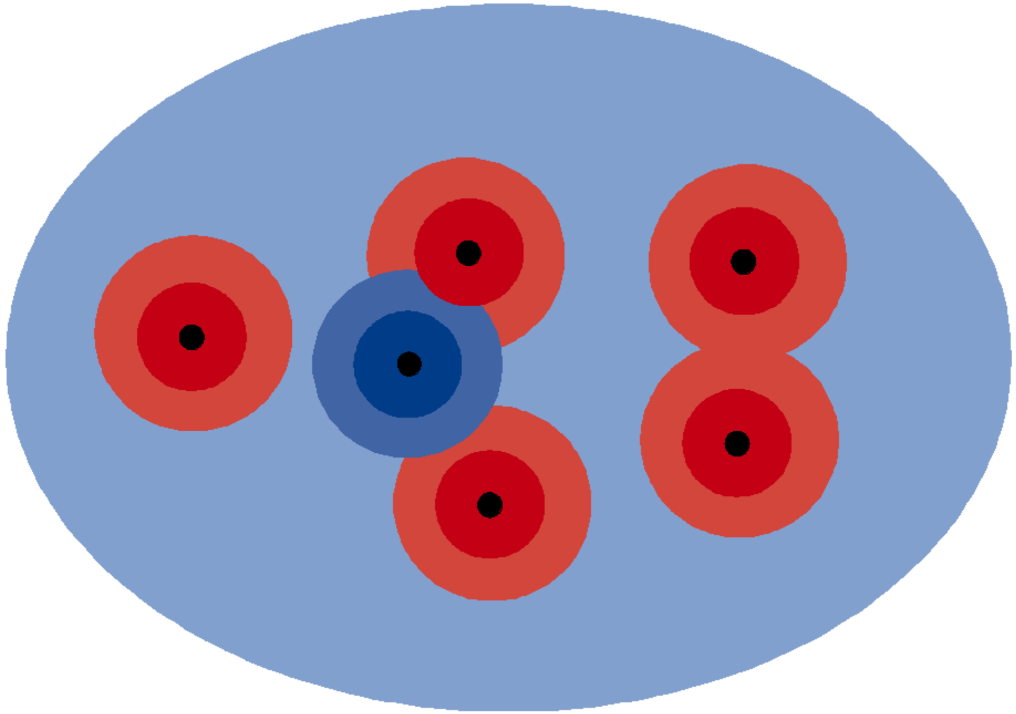


- for each $t \in T$, perform backward search up to the top level,
  store search space entries $(t, u, d(u,t))$
- arrange search spaces: create a bucket for each u
- for each $s \in S$, perform forward search up to and including the top level,
  at each node u, scan all entries $(t, u, d(u,t))$ and
  compute $d(s,u) + d(u,t)$, update $D[s,t]$

Asymmetry

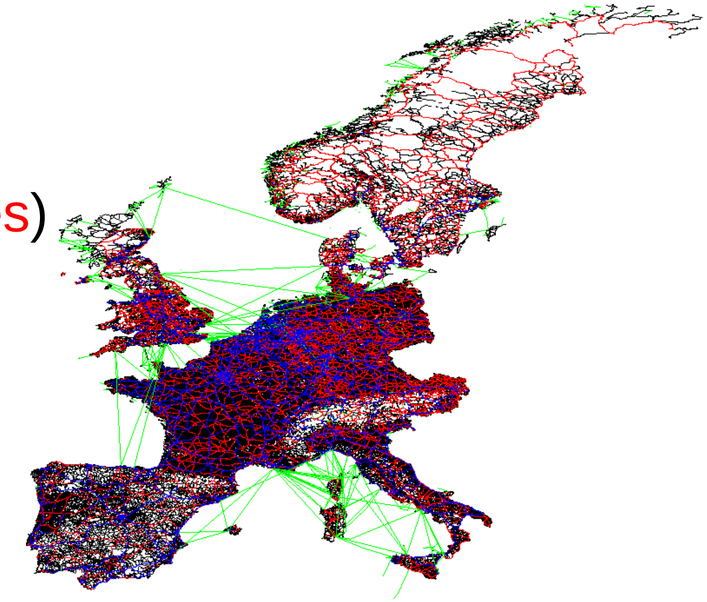
for large distance tables, most time spent on bucket scanning

Solution: use less levels ⇝ strengthen the asymmetry



- backward search spaces get smaller ⇝ less bucket entries
- forward search spaces get bigger

Experiments



Input:

- Western European road network (18 million nodes)
- random source/target node sets

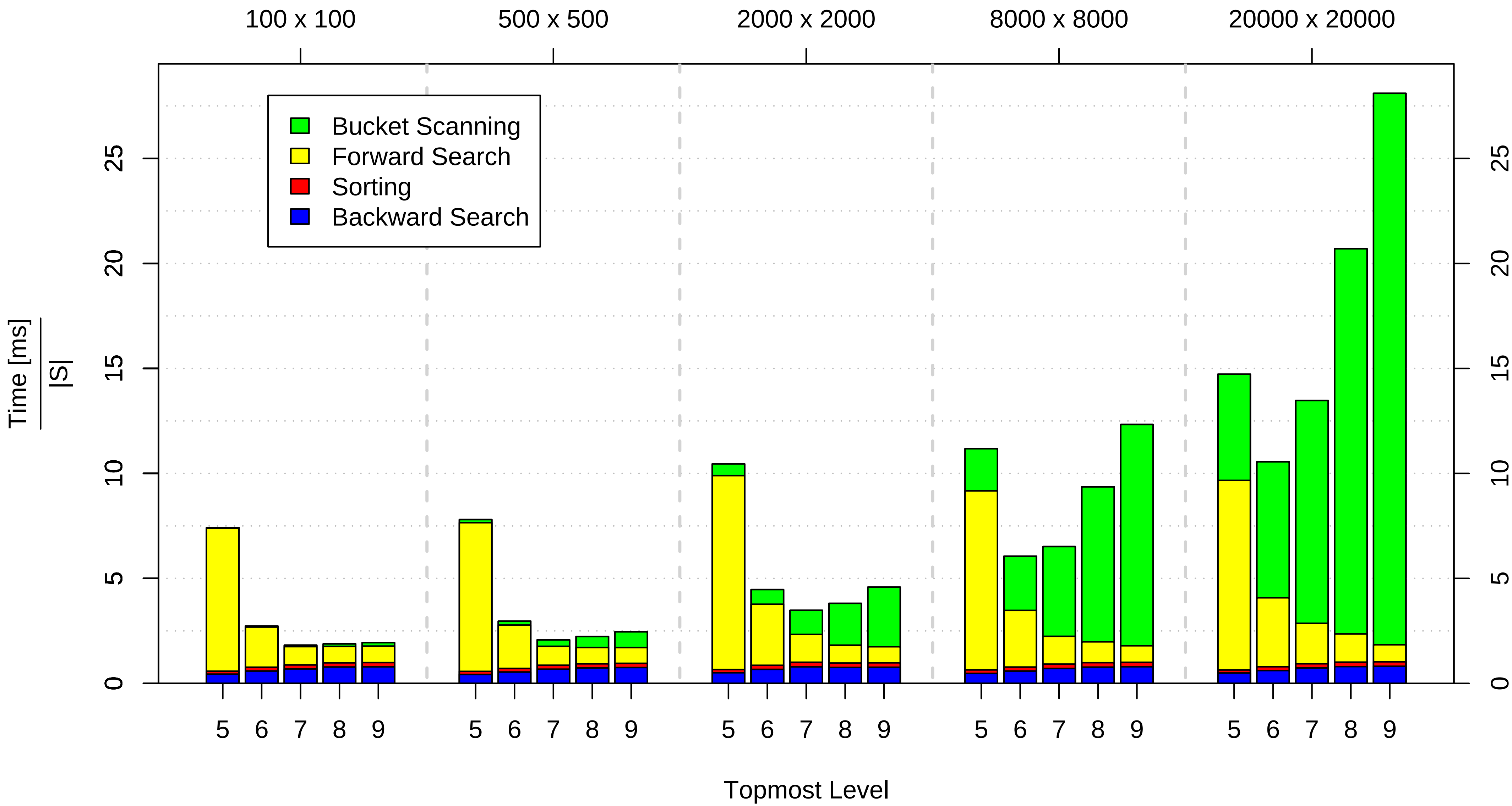
Results:

table size	time	speedup (↔ DIJKSTRA)
1 000 × 1 000	2.5 s	4 680
10 000 × 10 000	58 s	2 017

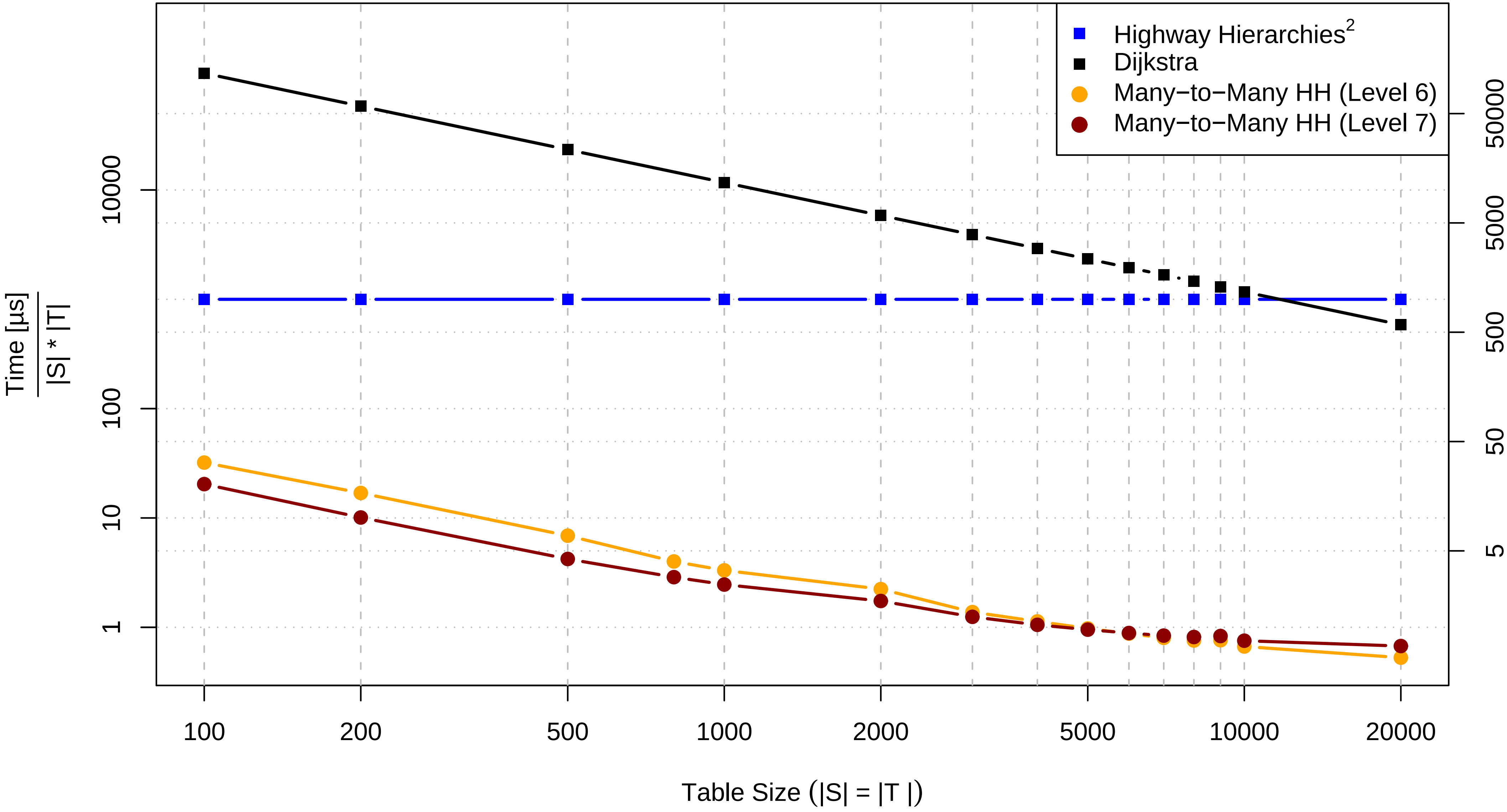
Break Even Point (w.r.t. preprocessing costs): table size 100 × 100

Real-World Instances: similar performance

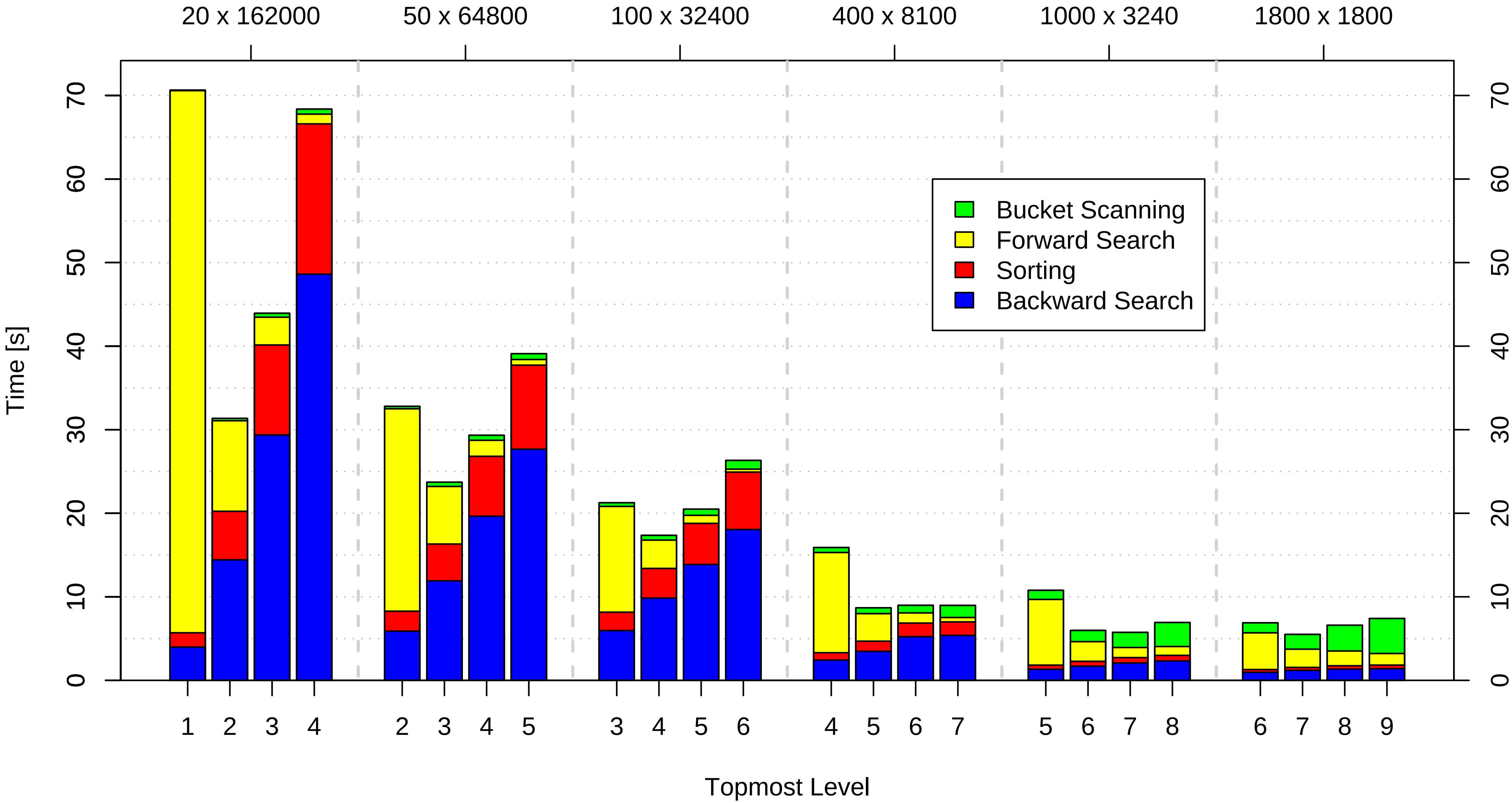
Symmetric Instances



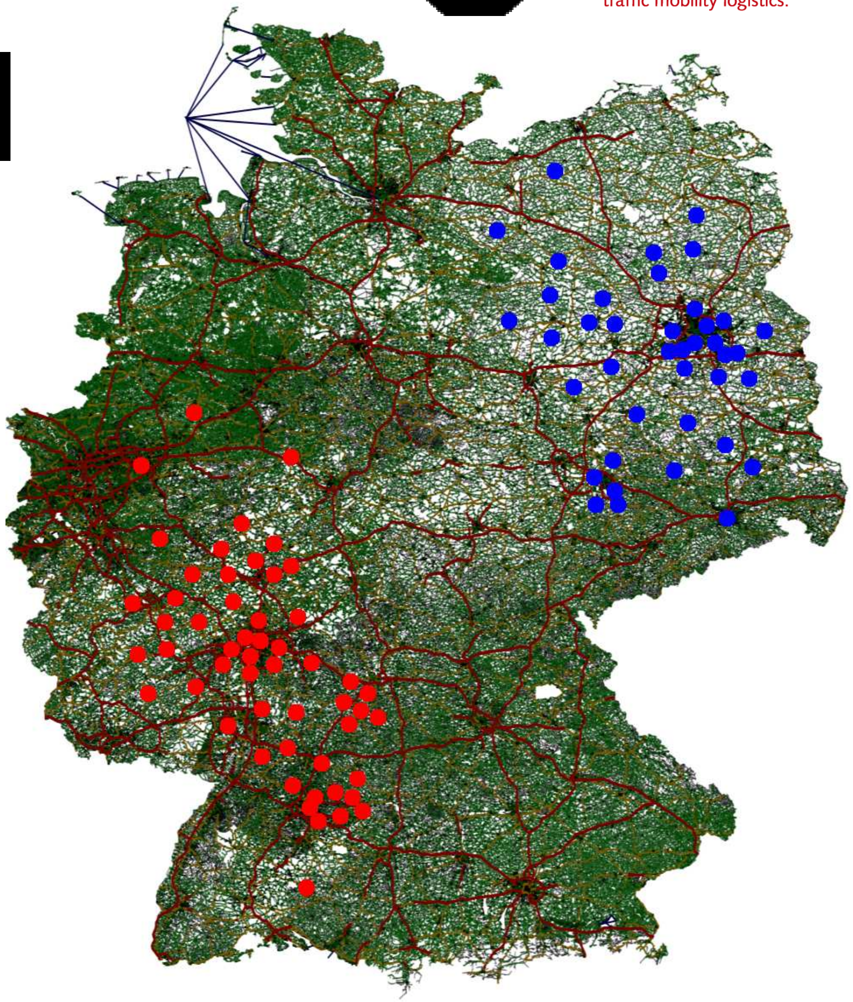
Comparisons



Asymmetric Instances

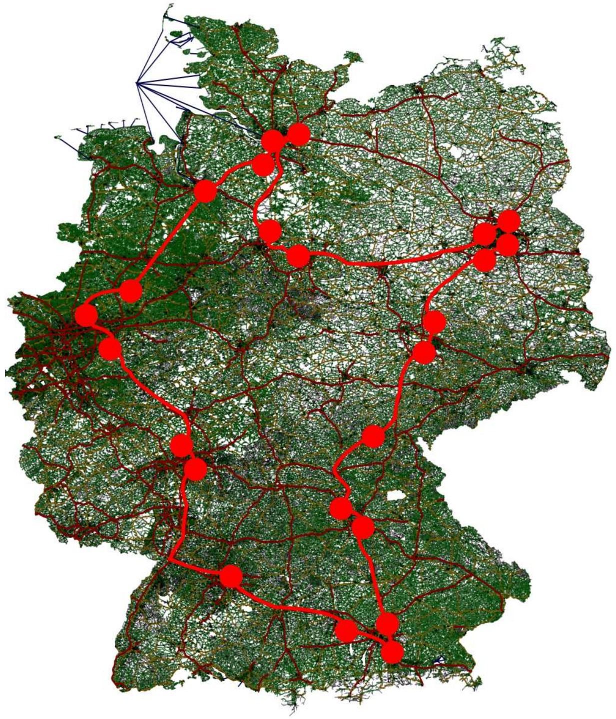


Summary



- very efficient solution to the many-to-many shortest path problem
- requires little preprocessing time ≈ 15 minutes
- computes 10 000 × 10 000 table in ≈ 1 minute (0.6 μs per entry)

Additional Issues



- outputting paths
- incremental computation
- parallelization

Future Work

- adapt preprocessing to specific source/target node sets
- approach can be generalized to other
  - non-goal-directed
  - bidirectional

  speedup techniques