Accurate High-Performance Route Planning

Peter Sanders **Dominik Schultes**

Institut für Theoretische Informatik – Algorithmik II

Universität Karlsruhe (TH)

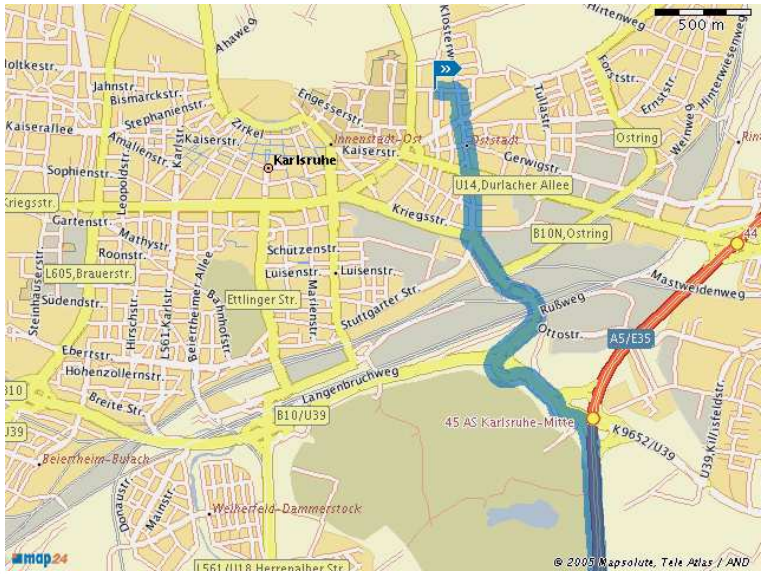
`http://algo2.iti.uka.de/schultes/hwy/`

Zurich, May 9, 2006

How do I get there from here ?

Applications

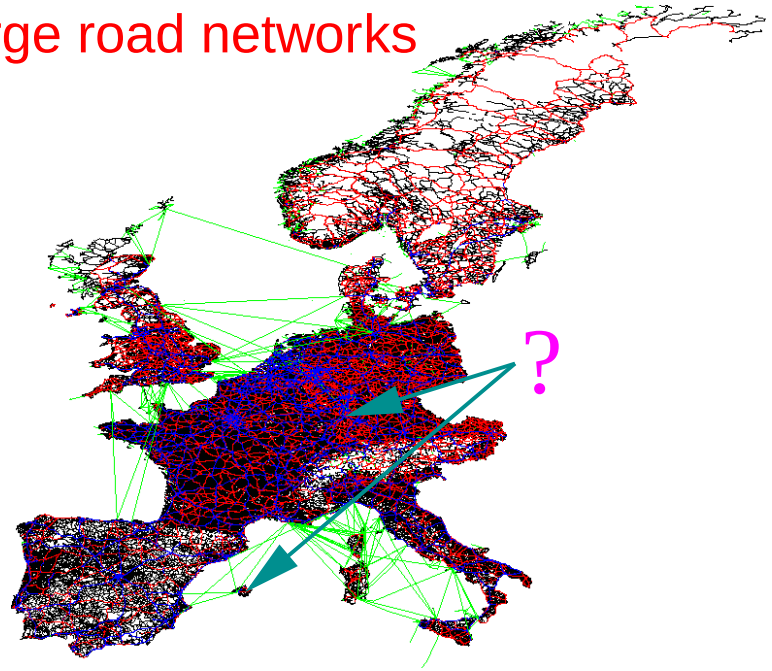
- route planning systems
  in the internet
  (e.g. `www.map24.de`)
- car navigation systems
- . . .



Goals

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

  i.e., optimised not only for long paths

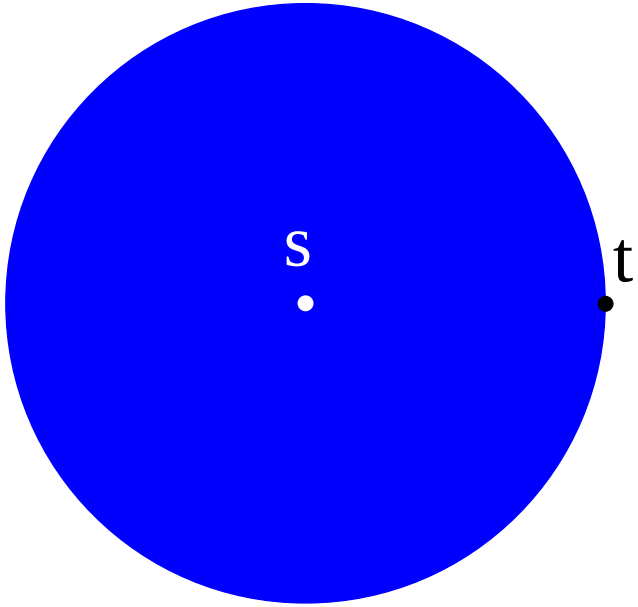


Related Work

method	query	prepr.	space	scale	source
basic A^*	$-$	$++$	$++$	$+$	[Hart et al. 68]
bidirected	$-$	$++$	$++$	$+$	[Pohl 71]
heuristic hwy hier.	$+$	$++$	$+$	$+$	[commercial]
separator hierarchies	o	?	$-$	$-$	[several groups 02]
geometric containers	$++$	$--$	$+$	$+$	[Wagner et al. 03]
bitvectors	$++$	$-$	o	$-$	[Lauther. . . 04]
landmarks	$+$	$++$	$-$	$-$	[Goldberg et al. 04]
landmarks $+$ reaches	$++$	o	o	o	[Goldberg et al. 06]
highway hierarchies	$++$	$+$	$+$	$+$	here

DIJKSTRA's Algorithm

Dijkstra



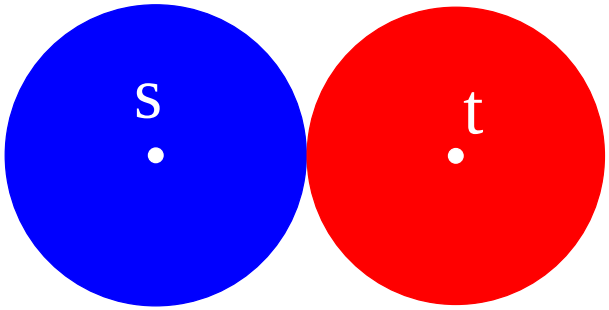
not practicable

for large road networks

(e.g. Western Europe:

≈ 18 000 000 nodes)

bidirectional Dijkstra



improves the running time,

but still too slow

Naive Route Planning

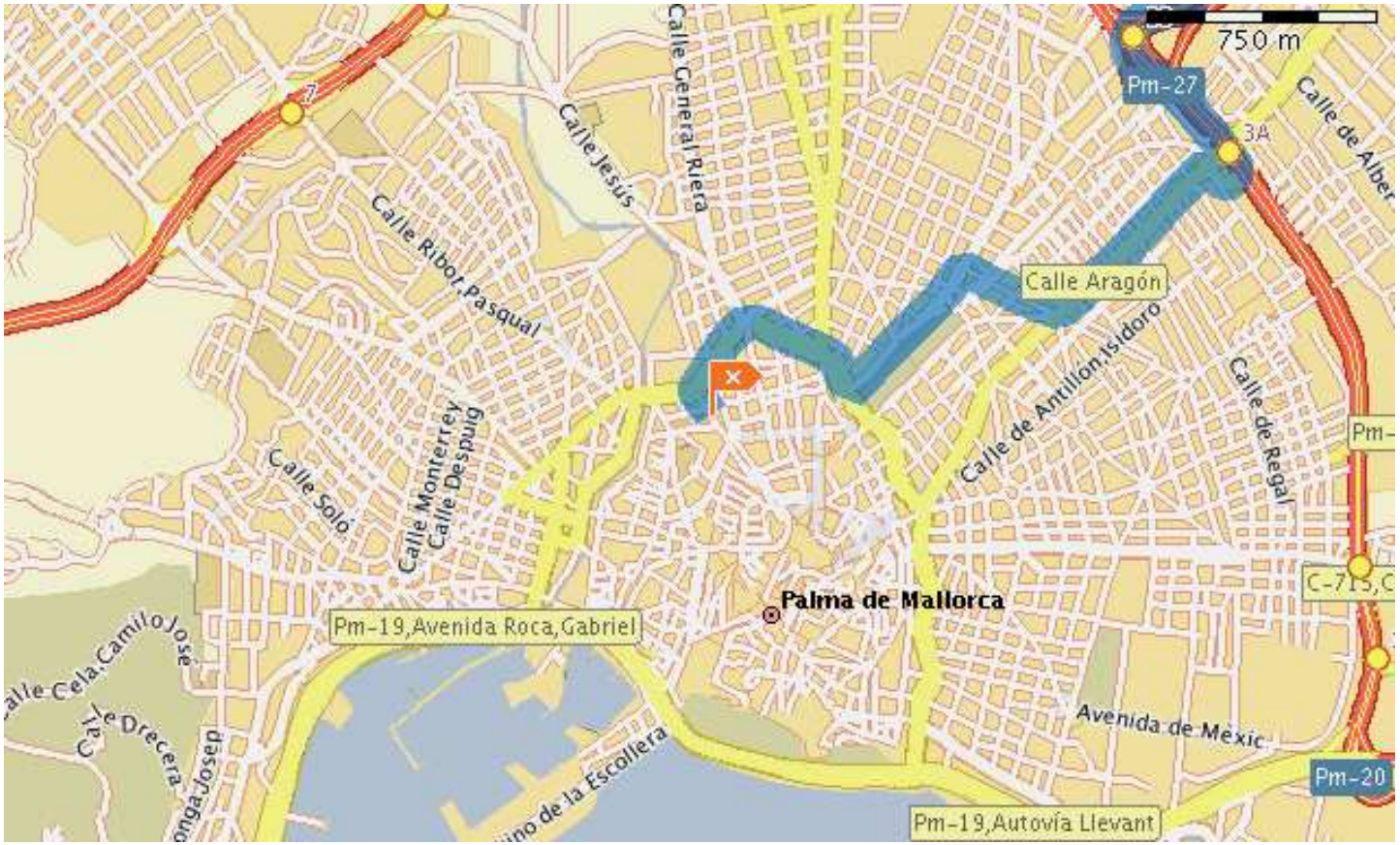
1. Look for the next reasonable motorway

Naive Route Planning

1. Look for the next reasonable motorway
2. Drive on motorways to a location close to the target

Naive Route Planning

1. Look for the next reasonable motorway
2. Drive on motorways to a location close to the target
3. Search the target starting from the motorway exit



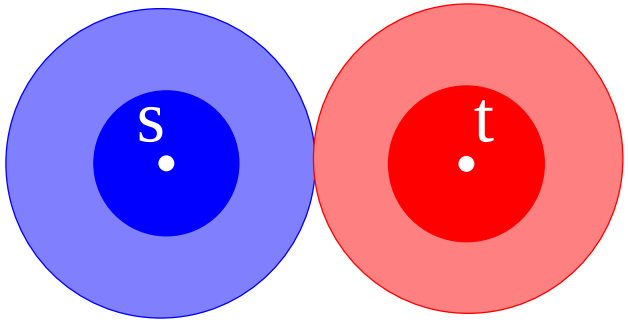
Commercial Systems

1. Search from the source and target node (‘bidirectional’)
   within a certain radius (e.g. 20 km),
   consider all roads
2. Continue the search within a larger radius (e.g. 100 km),
   consider only national roads and motorways
3. Continue the search,
   consider only motorways

fast, but not exact

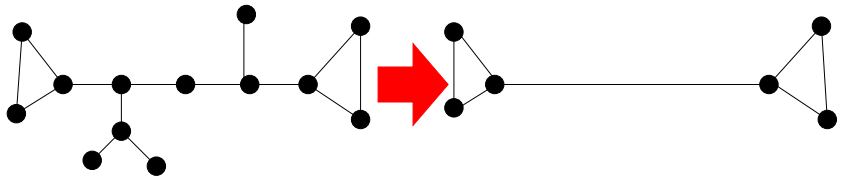
Exact Highway Hierarchies

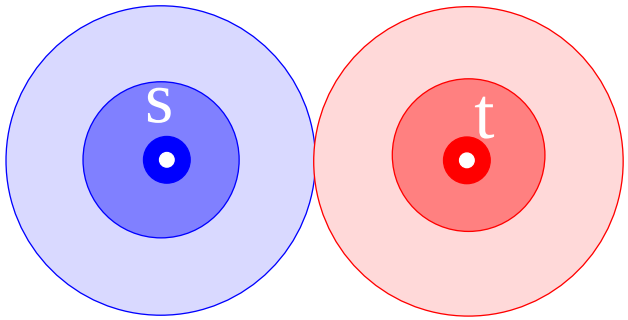
- complete search within a local area
- search in a (thinner) highway network



= minimal graph that preserves all shortest paths

- contract network, e.g.,
- iterate $\rightsquigarrow$ highway hierarchy





A Meaning of "Local"

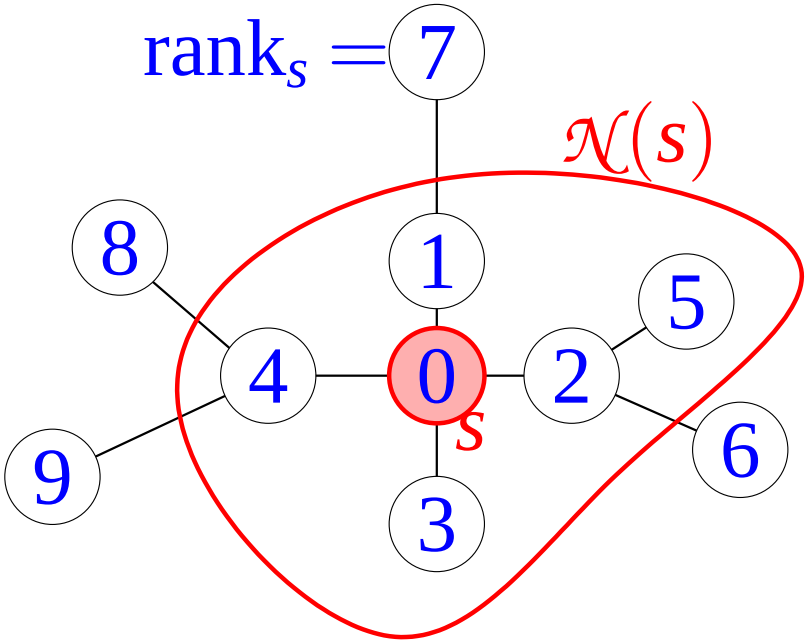
- choose neighbourhood radius $r(s)$

  e.g. distance to the H-closest node for a fixed parameter H

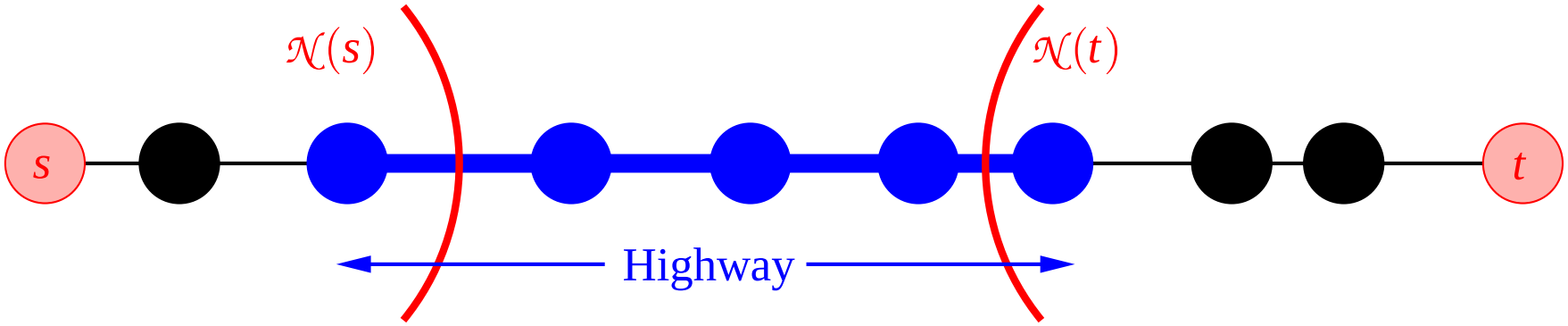
- define neighbourhood of s:

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

- example for $H = 5$



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

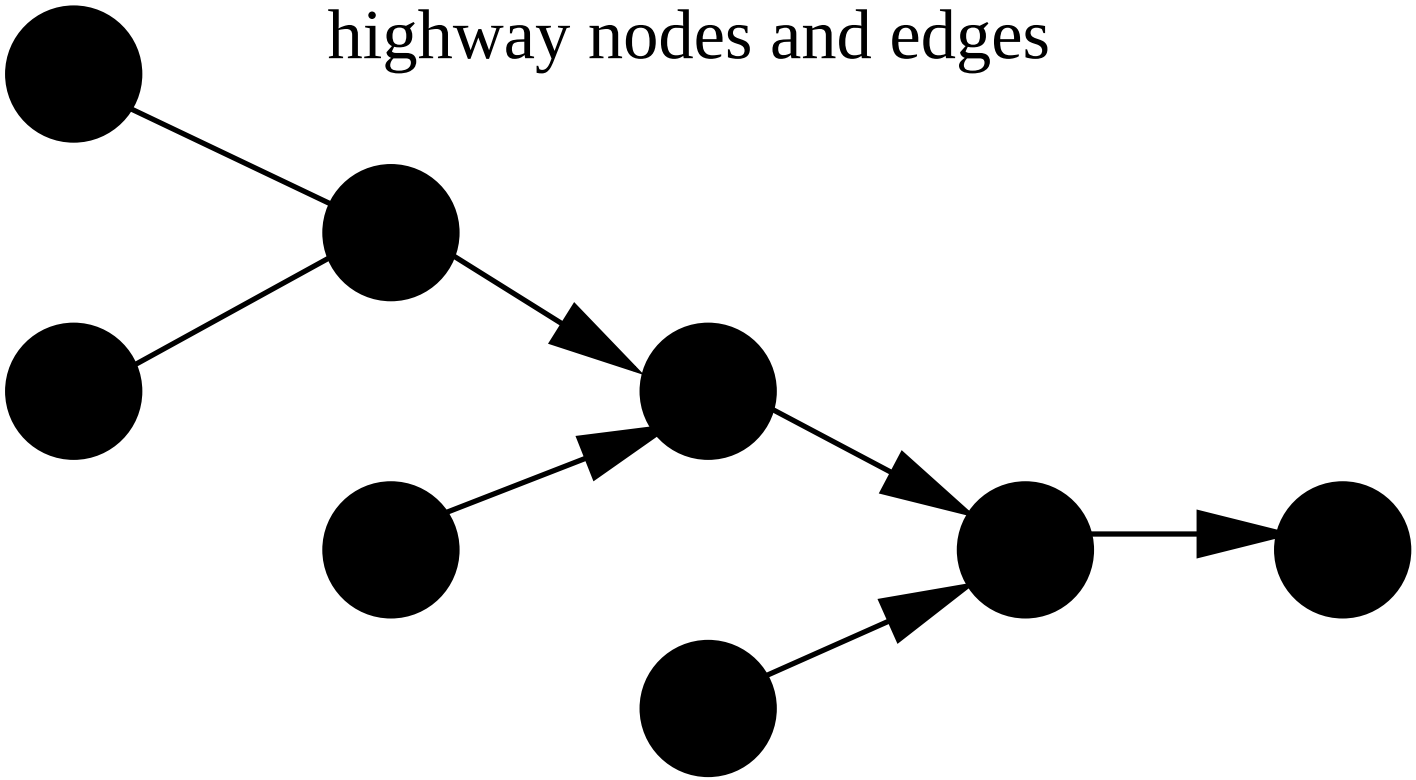
- $v \notin \mathcal{N}(s)$

and

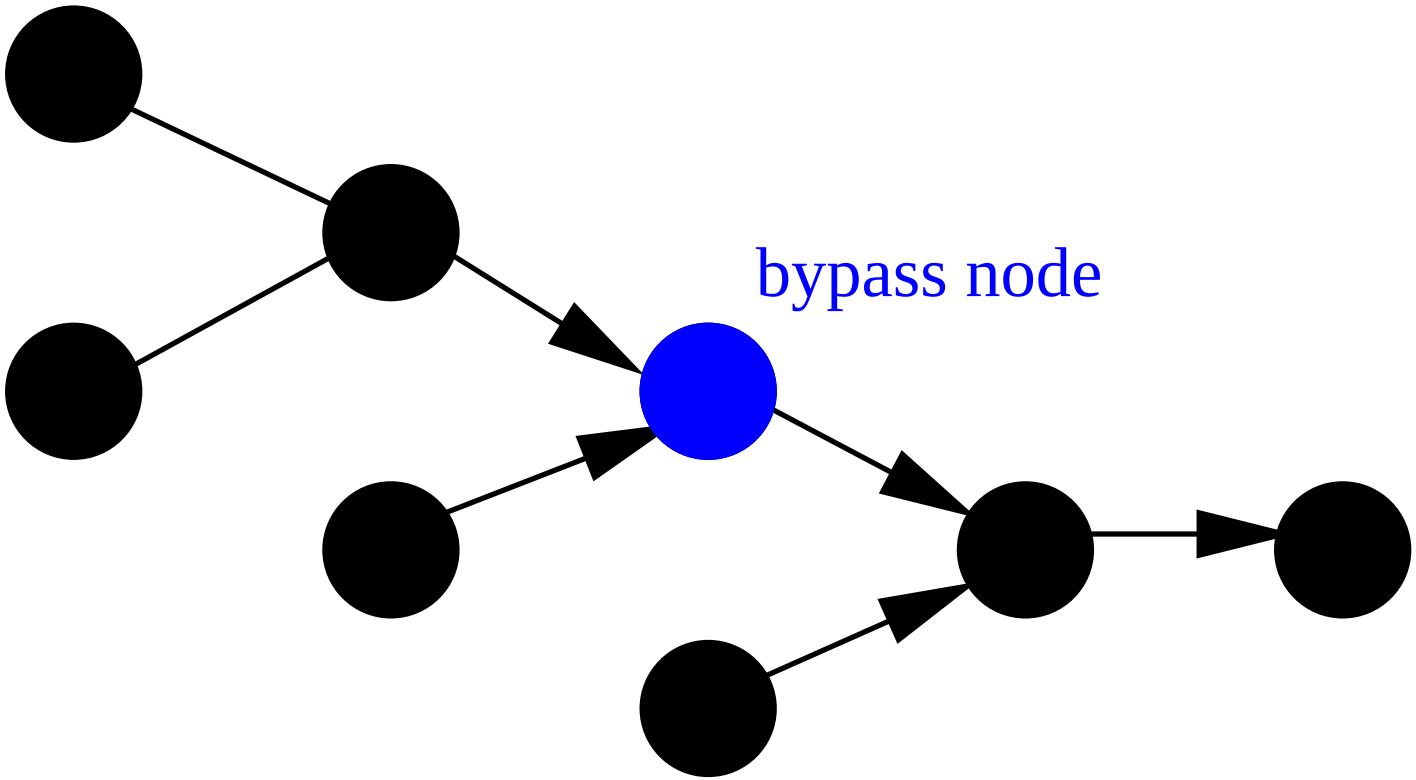
- $u \notin \mathcal{N}(t)$

Contraction

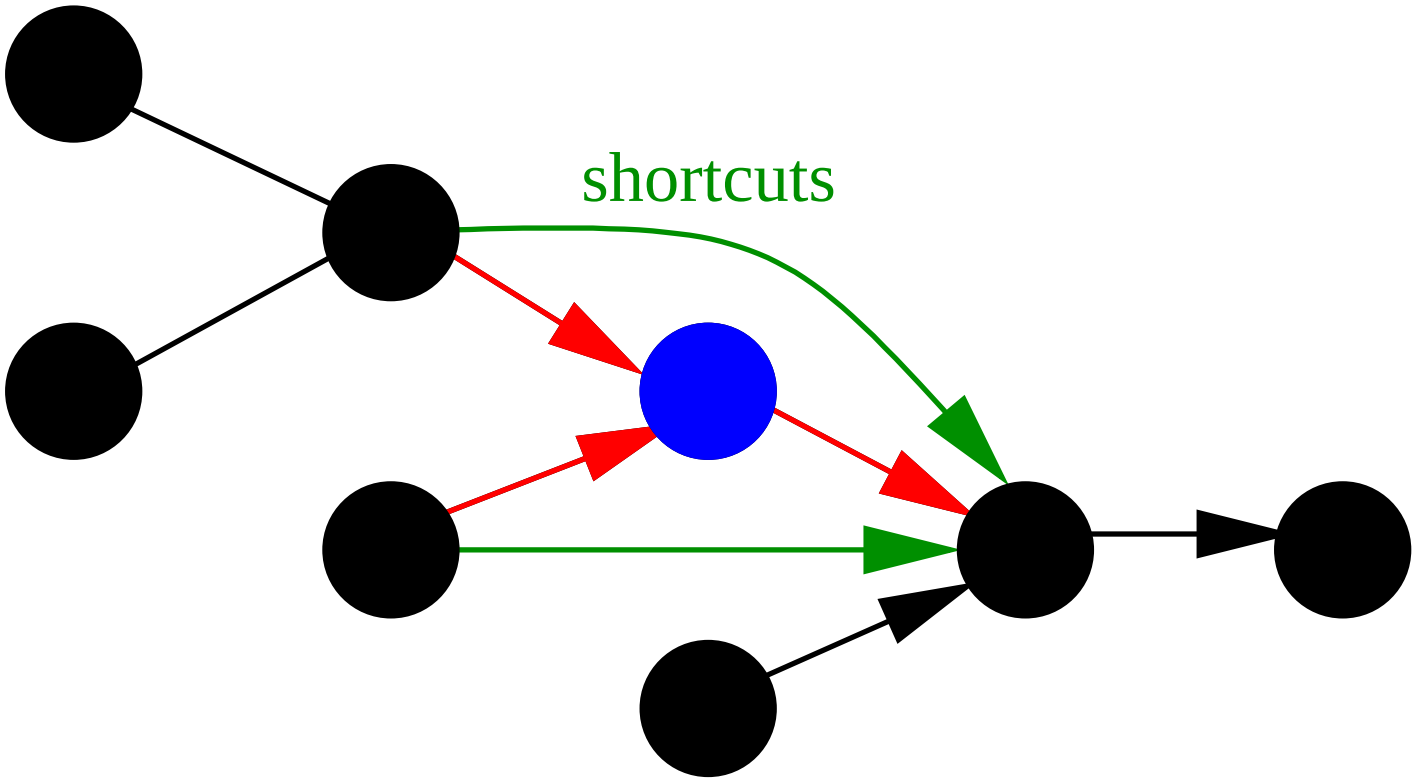
highway nodes and edges



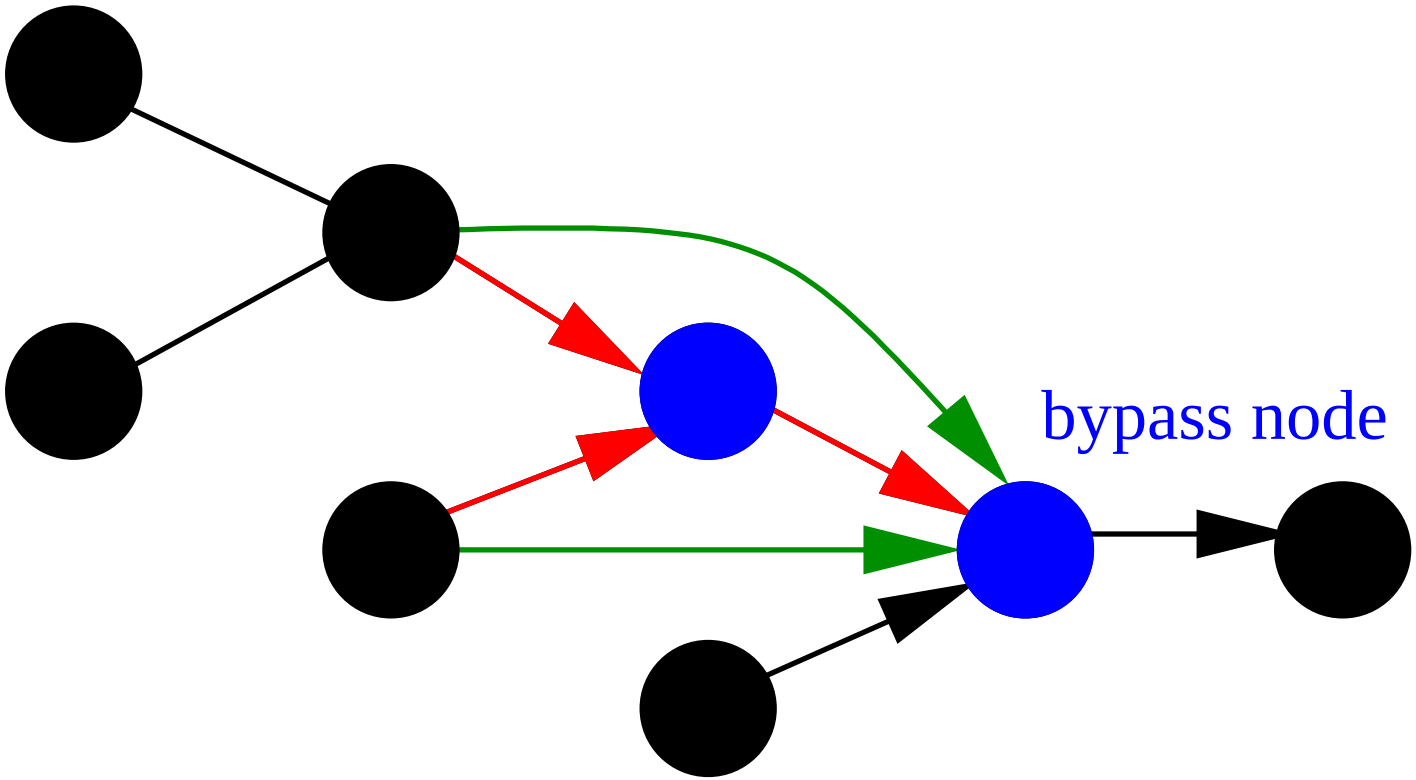
Contraction



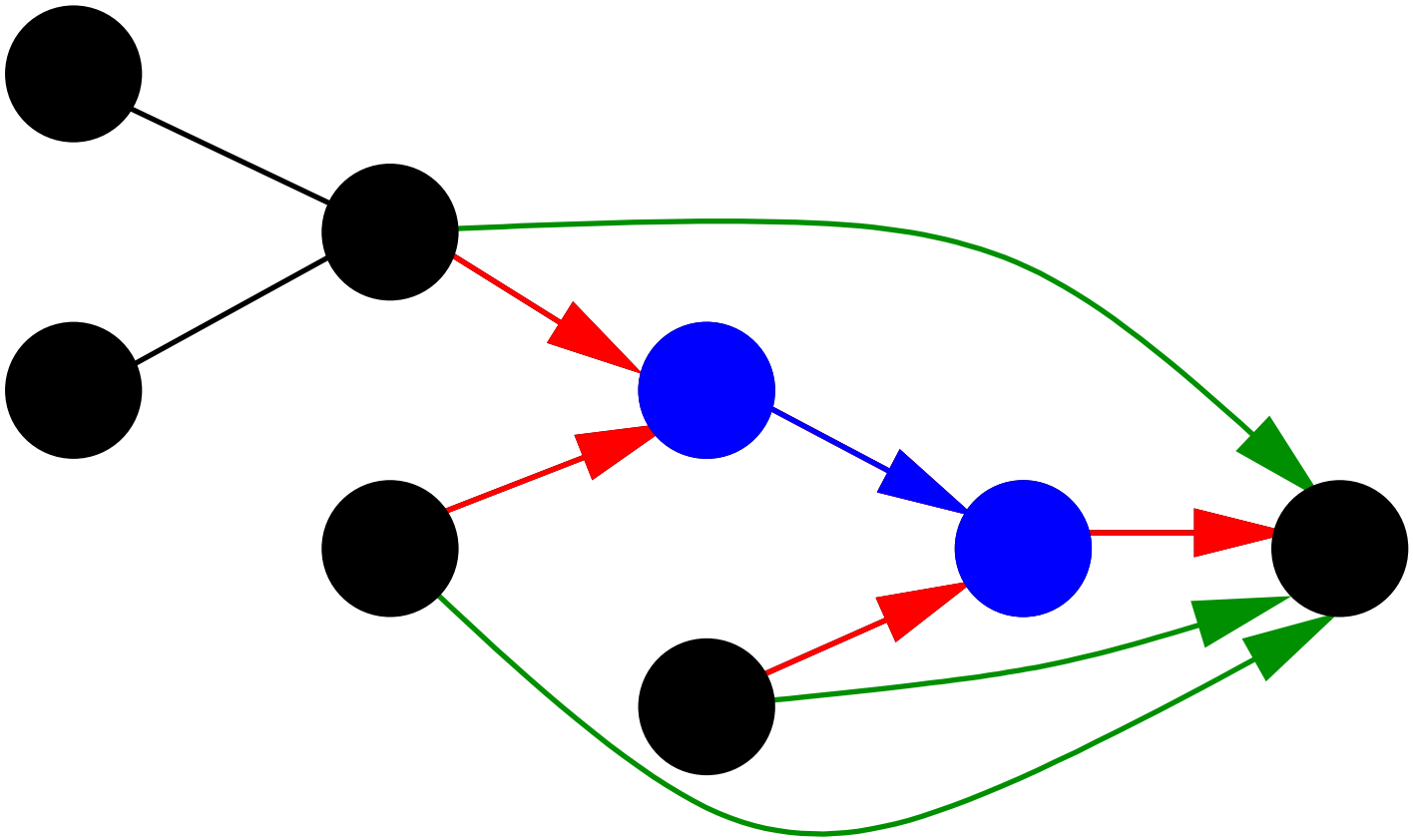
Contraction



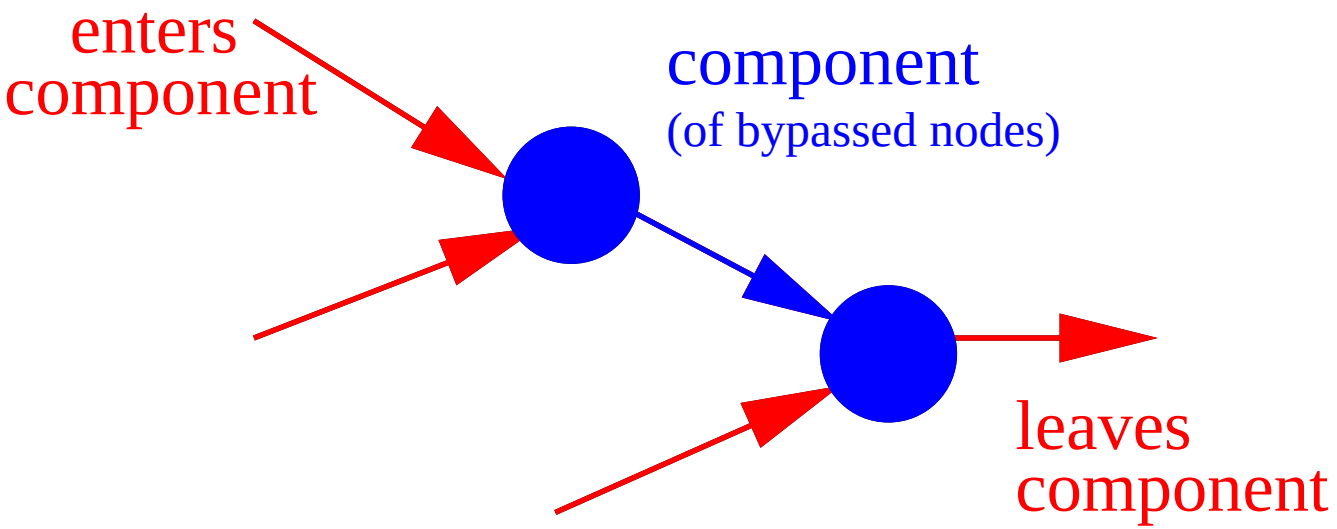
Contraction



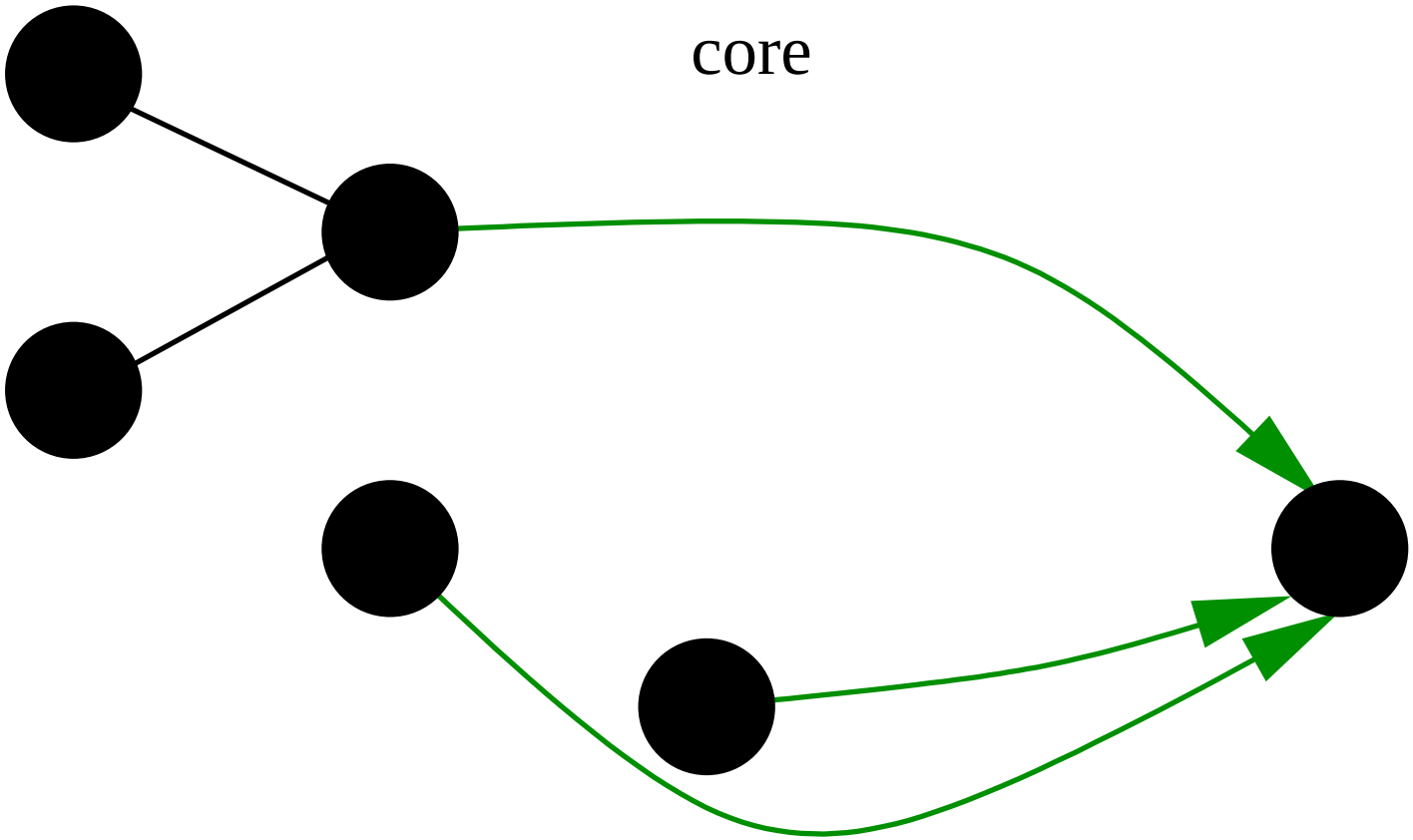
Contraction



Contraction



Contraction



Contraction

Which nodes should be bypassed?

Use some heuristic taking into account

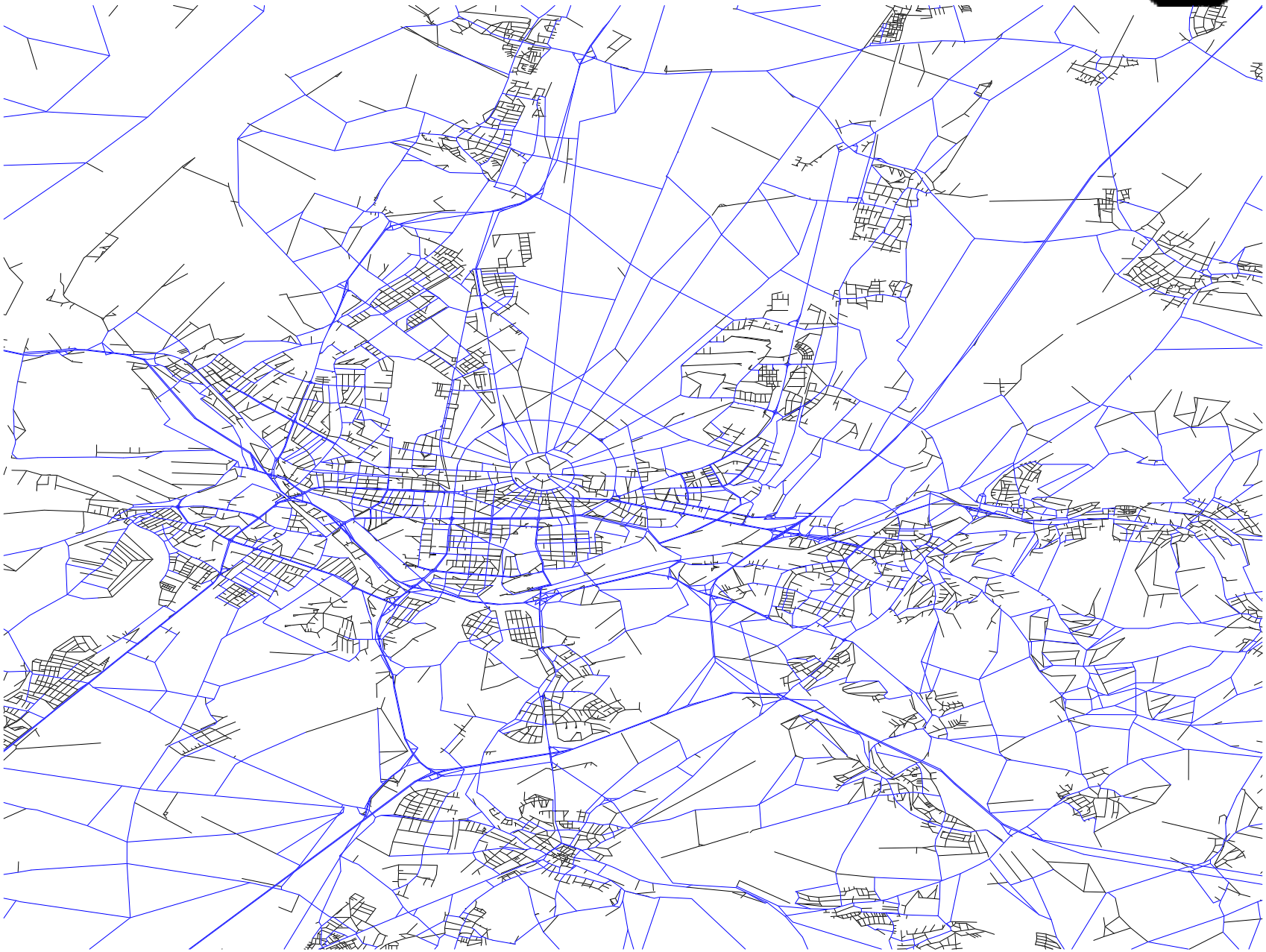
- the number of shortcuts that would be created and
- the degree of the node.

Construction

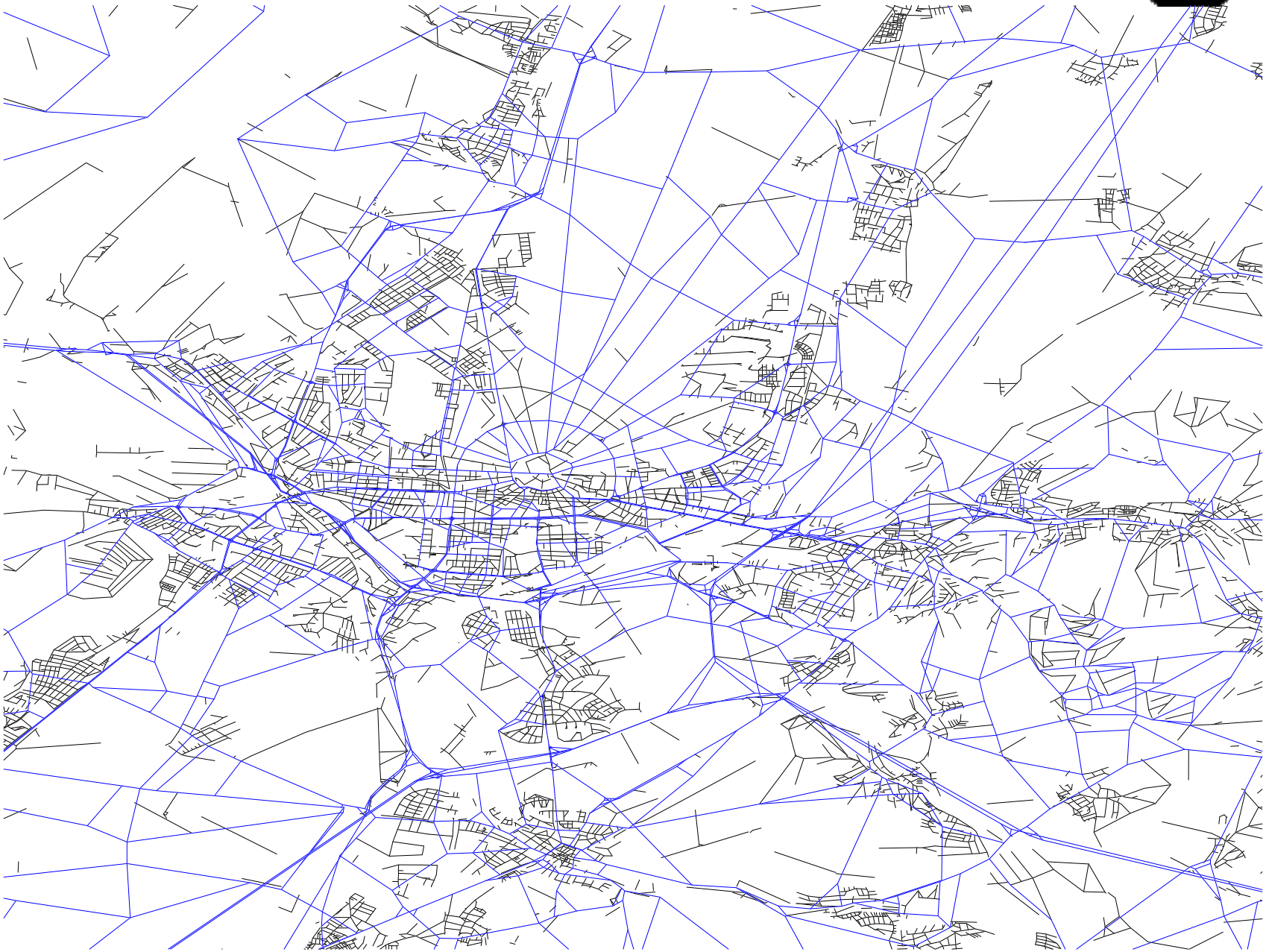
Example: Western Europe, bounding box around Karlsruhe

Complete Road Network

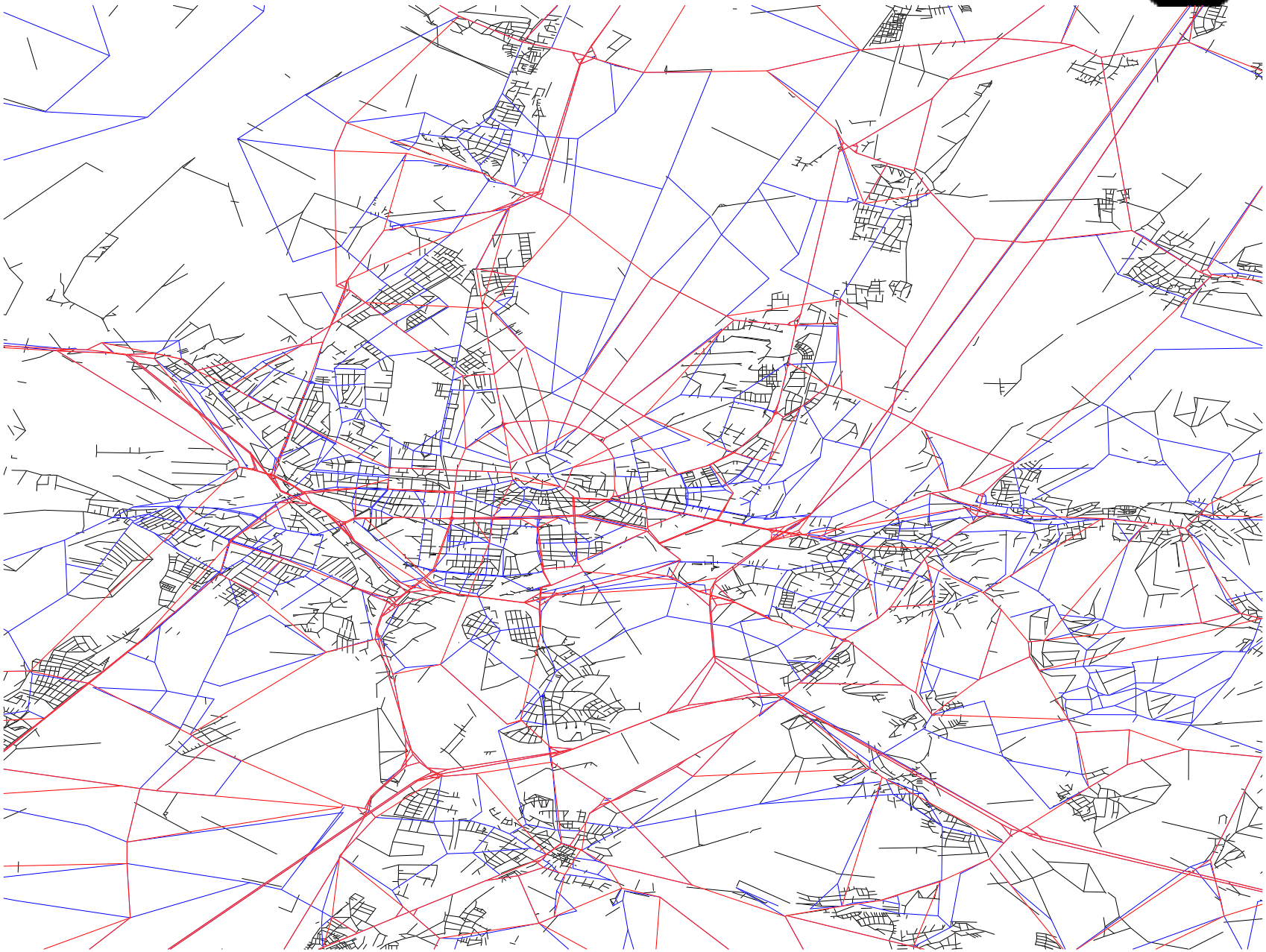
Highway Network



Contracted Network (Core)



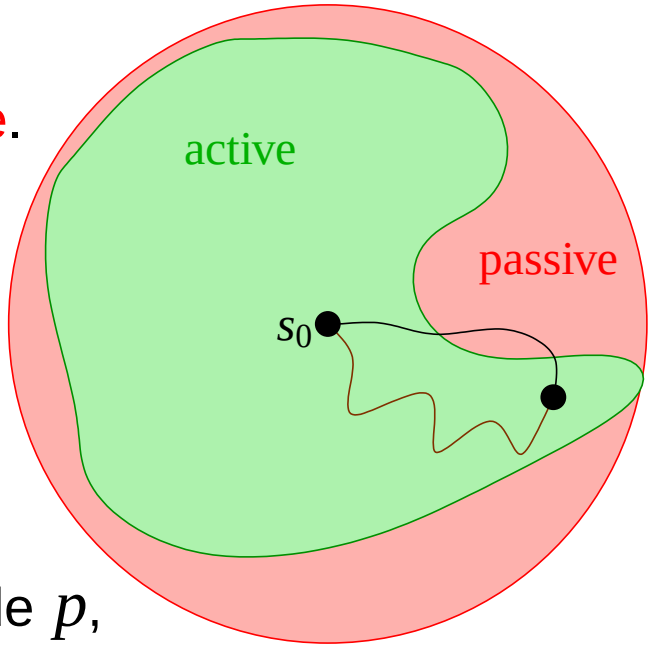
Highway Hierarchy: Level 1 and Level 2



Fast Construction

Phase 1: *Construction of Partial Shortest Path Trees*

For each node s_0, perform an SSSP search from s_0.

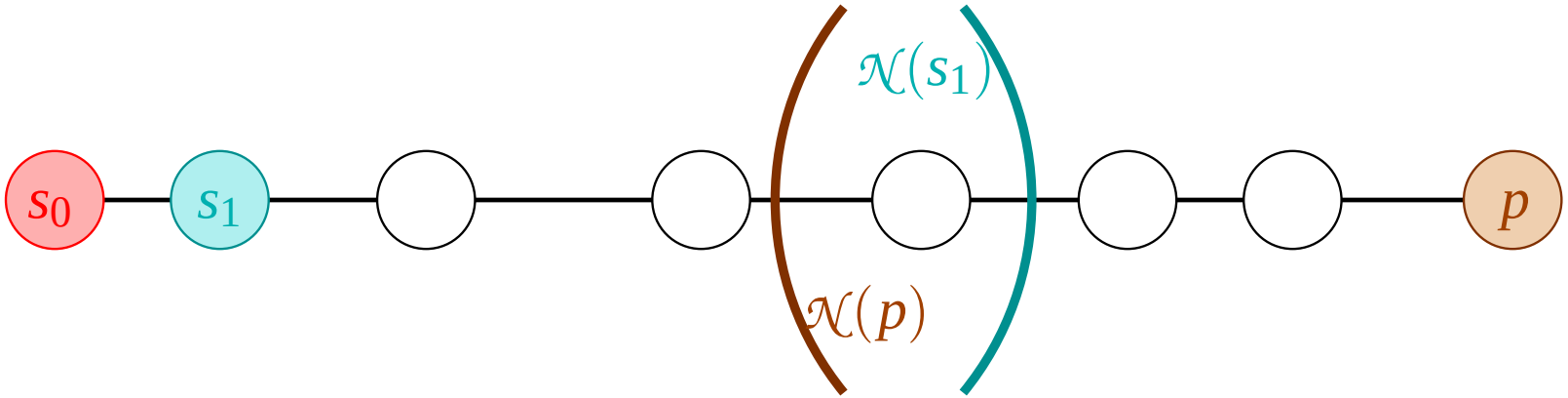


- A node's state is either active or passive.
- s_0 is active.
- A node inherits the state of its parent in the shortest path tree.
- If the abort condition is fulfilled for a node p, p's state is set to passive.

The search is aborted when all queued nodes are passive.

Fast Construction

Abort Condition:



p is set to passive *iff*

$$|\mathcal{N}(s_1) \cap \mathcal{N}(p)| \leq 1$$

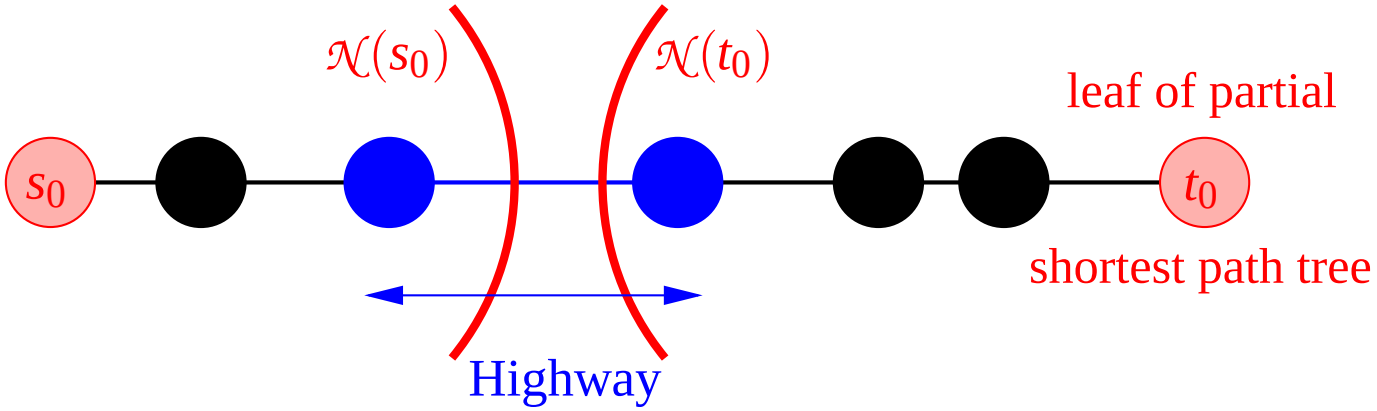
Fast Construction, Phase 2

Theorem:

The tree roots and leaves encountered in Phase 1

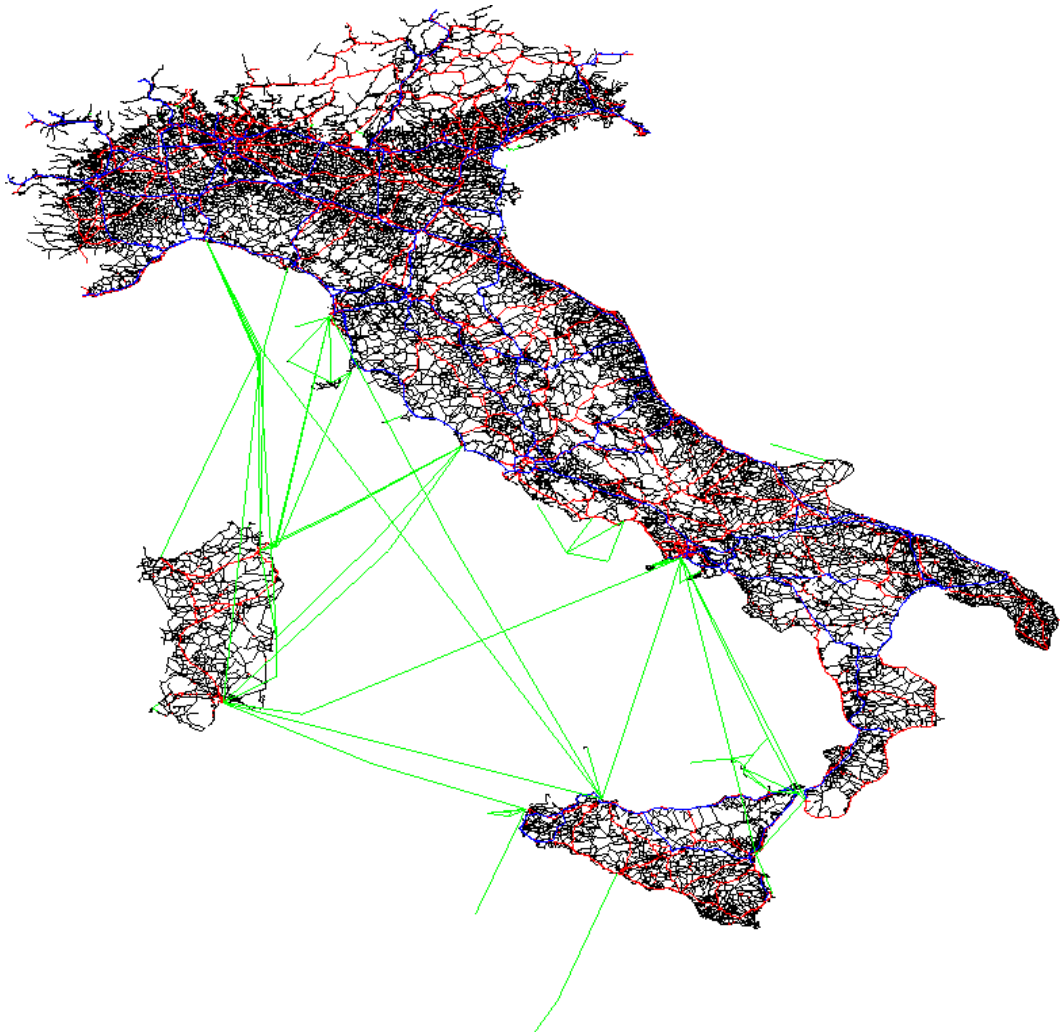
witness all highway edges.

The highway edges can be found in time linear in the tree sizes.



Fast Construction

Problem: very long edges, e.g. ferries



Faster Construction

Solution: An active node v is declared to be a maverick if

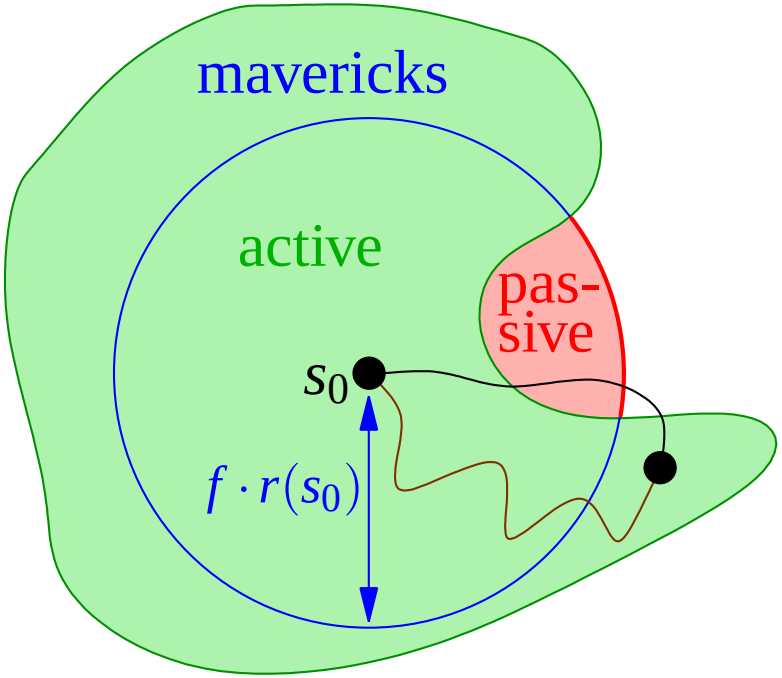
$$d(s_0, v) > f \cdot r(s_0).$$

When all active nodes are mavericks,

the search from passive nodes is

no longer continued.

$\rightsquigarrow$ superset of the highway network



Space Consumption

Choose neighborhood sizes such that levels shrink geometrically

$\rightsquigarrow$ linear space consumption

Arbitrarily Small Constant Factor (not implemented):

- Large $H_0 \rightsquigarrow$ large level-0 radius $\rightsquigarrow$ small higher levels
- No $r(\cdot)$ needed for level-0 search (under certain assumptions)
- Mapping to next level by hash table

Query

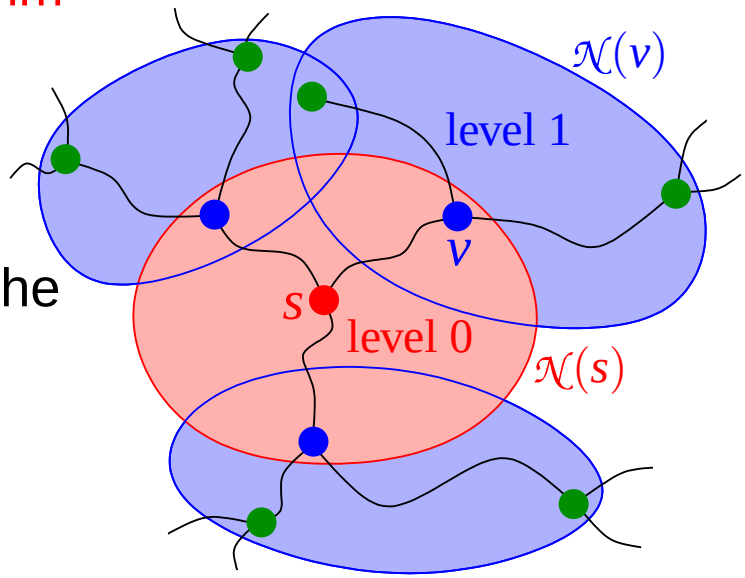
Bidirectional version of Dijkstra's Algorithm

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.



- entrance point to level 0
- entrance point to level 1
- entrance point to level 2

Query (bidir. Dijkstra I)

Operations on two priority queues $\overrightarrow{Q}$ and $\overleftarrow{Q}$:

- void **insert**(nodeID, key)
- void **decreaseKey**(nodeID, key)
- nodeID **deleteMin**()

node u has key $\delta(u)$

- (tentative) distance from the respective source node

Query (bidir. Dijkstra II)

```
query(s,t) {
    →Q.insert(s,0); ←Q.insert(t,0);
    while (→Q ∪ ←Q ≠ ∅) do {
        ⇋ ∈ {→,←};                    //select direction
        u := ⇋Q.deleteMin();
        relaxEdges(⇋,u);
    }
}
```

Query (bidir. Dijkstra III)

relaxEdges$(\overleftrightharpoons, u)$ {

  foreach $e = (u, v) \in \overrightarrow{\overleftrightharpoons E}$ **do** {

    k := $\delta(u)$ + $w(e)$;

    if $v \in \overleftrightharpoons Q$ **then** $\overleftrightharpoons Q$.decreaseKey(v, k); **else** $\overleftrightharpoons Q$.insert(v, k);

  }

}

Query (Hwy I)

Operations on two priority queues $\overrightarrow{Q}$ and $\overleftarrow{Q}$:

- void **insert**(nodeID, key)
- void **decreaseKey**(nodeID, key)
- nodeID **deleteMin**()

node u has key $(\delta(u), \ell(u), \text{gap}(u))$

- (tentative) distance from the respective source node
- search level
- gap to the next neighbourhood border

lexicographical order: <, <, >

Query (Hwy II)

```
query(s,t) {
    Q→.insert(s, (0,0,r0→(s))); Q←.insert(t, (0,0,r0←(t)));
    while (Q→ ∪ Q← ≠ ∅) do {
        ⇋ ∈ {→,←};                    //select direction
        u := Q⇋.deleteMin();
        relaxEdges(⇋, u);
    }
}
```

Query (Hwy III)

relaxEdges($\overleftrightharpoons, u$) {

foreach $e = (u,v) \in \overleftrightharpoons{E}$ **do** {

gap := gap(u);

if gap = ∞ **then** gap := $r^{\overleftrightharpoons}_{\ell(u)}(u)$; // leave component

for ($\ell := \ell(u)$; $w(e) >$ gap; ℓ++, gap := $r^{\overleftrightharpoons}_{\ell}(u)$); // go "upwards"

if $\ell(e) < \ell$ **then continue**; // Restriction 1

if e "enters a component" **then continue**; // Restriction 2

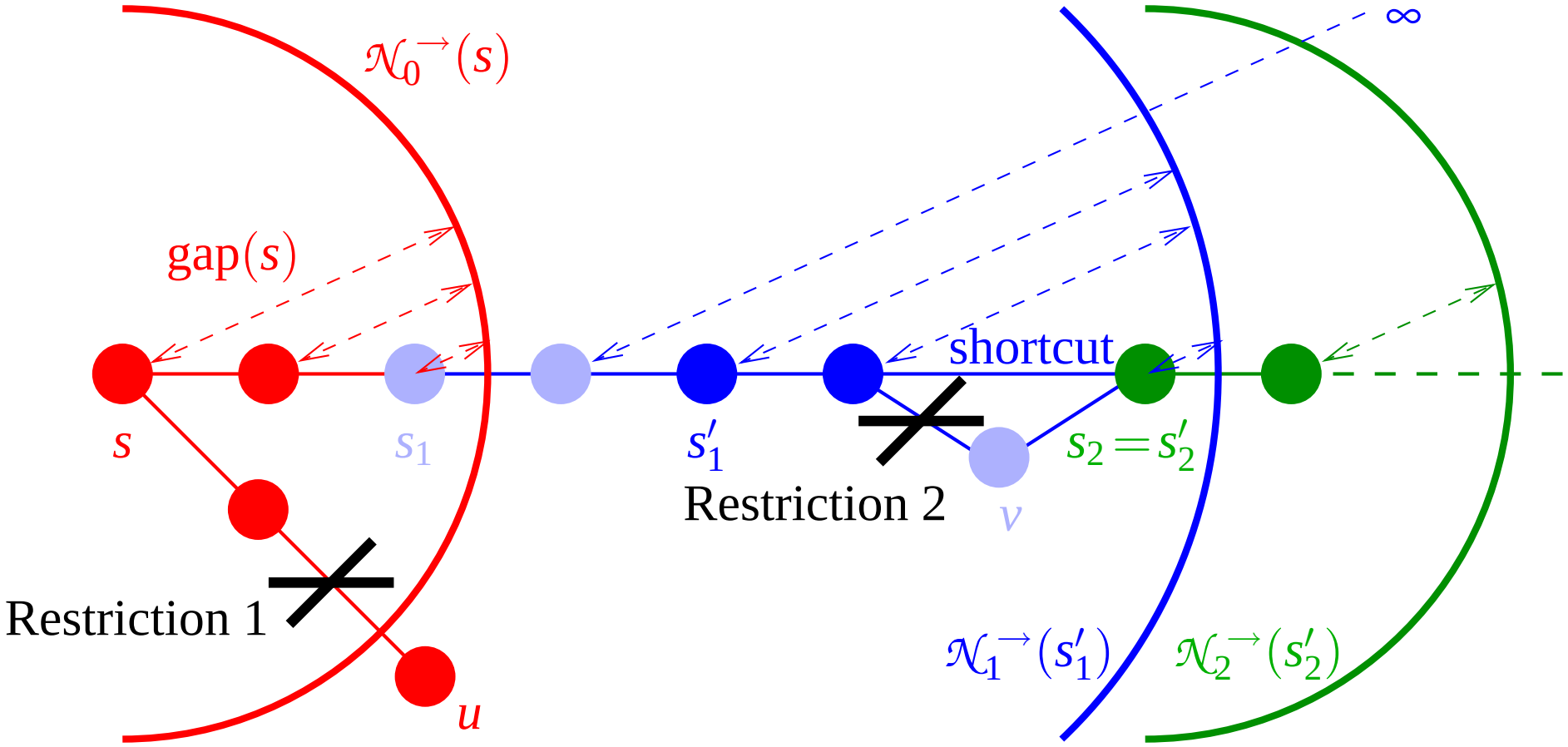
k := ($\delta(u) + w(e)$, ℓ, gap$-w(e)$);

if $v \in \overleftrightharpoons{Q}$ **then** $\overleftrightharpoons{Q}$.decreaseKey($v,k$); **else** $\overleftrightharpoons{Q}$.insert($v,k$);

}

}

Query


$\mathcal{N}_0^{\rightarrow}(s)$
gap(s)
∞
shortcut
s
s_1
s_1'
$s_2 = s_2'$
Restriction 2
v
Restriction 1
u
$\mathcal{N}_1^{\rightarrow}(s_1')$
$\mathcal{N}_2^{\rightarrow}(s_2')$

Query

Theorem:

We still find the shortest path.

Query

Example: from Karlsruhe, Am Fasanengarten 5

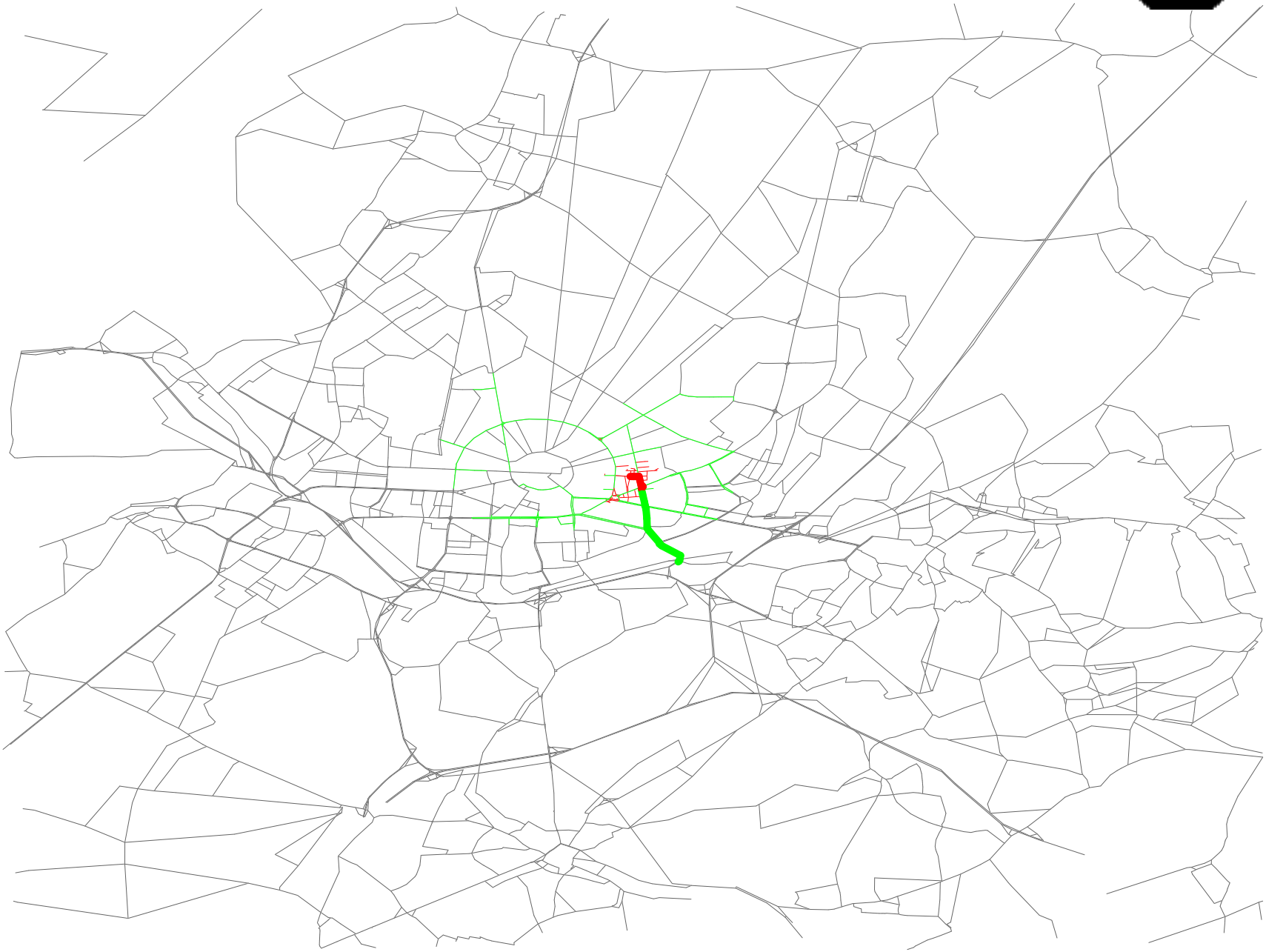
to Palma de Mallorca

Bounding Box: 20 km Level 0

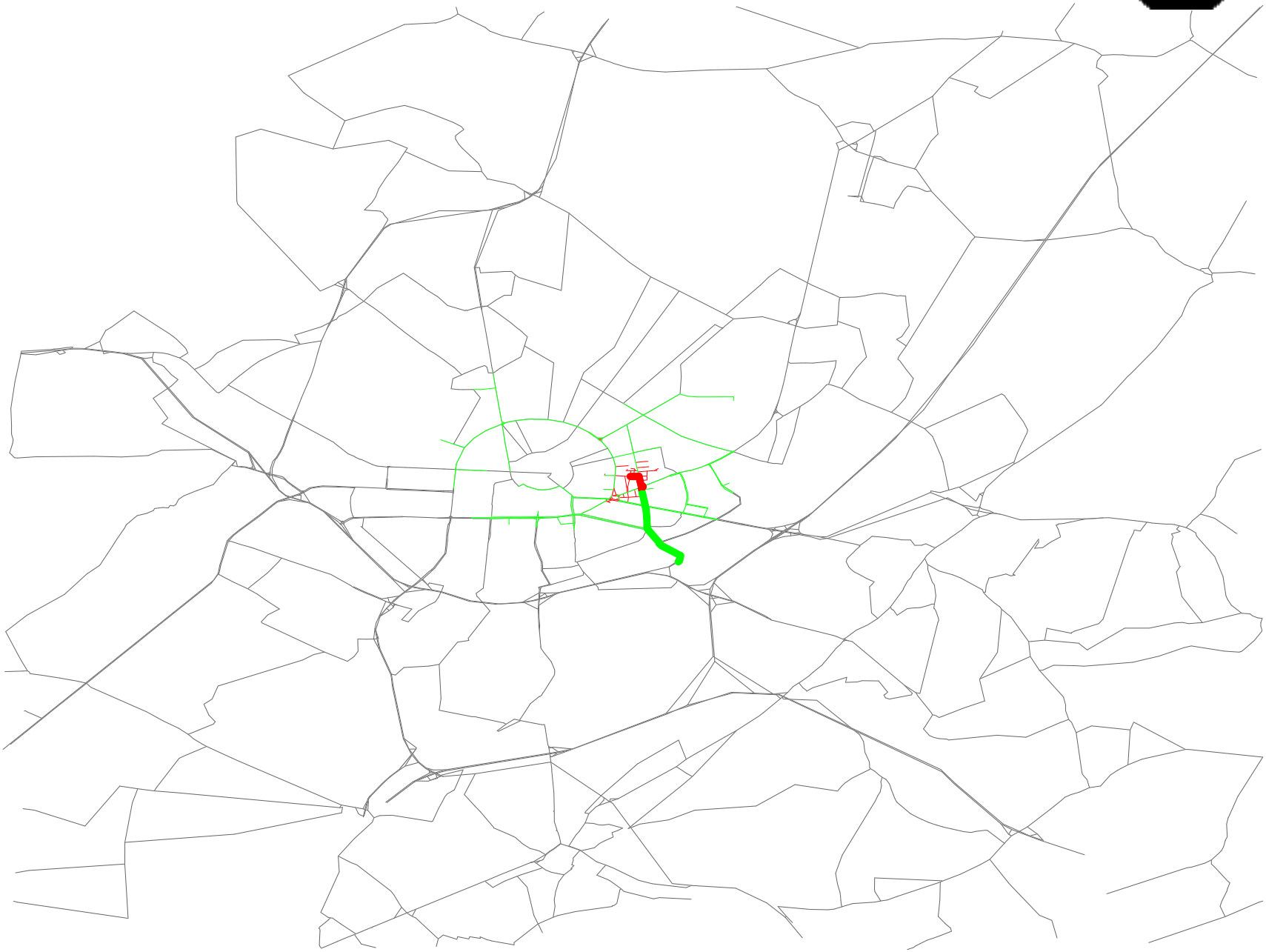
Bounding Box: 20 km Level 0 Search Space

Bounding Box: 20 km Level 1

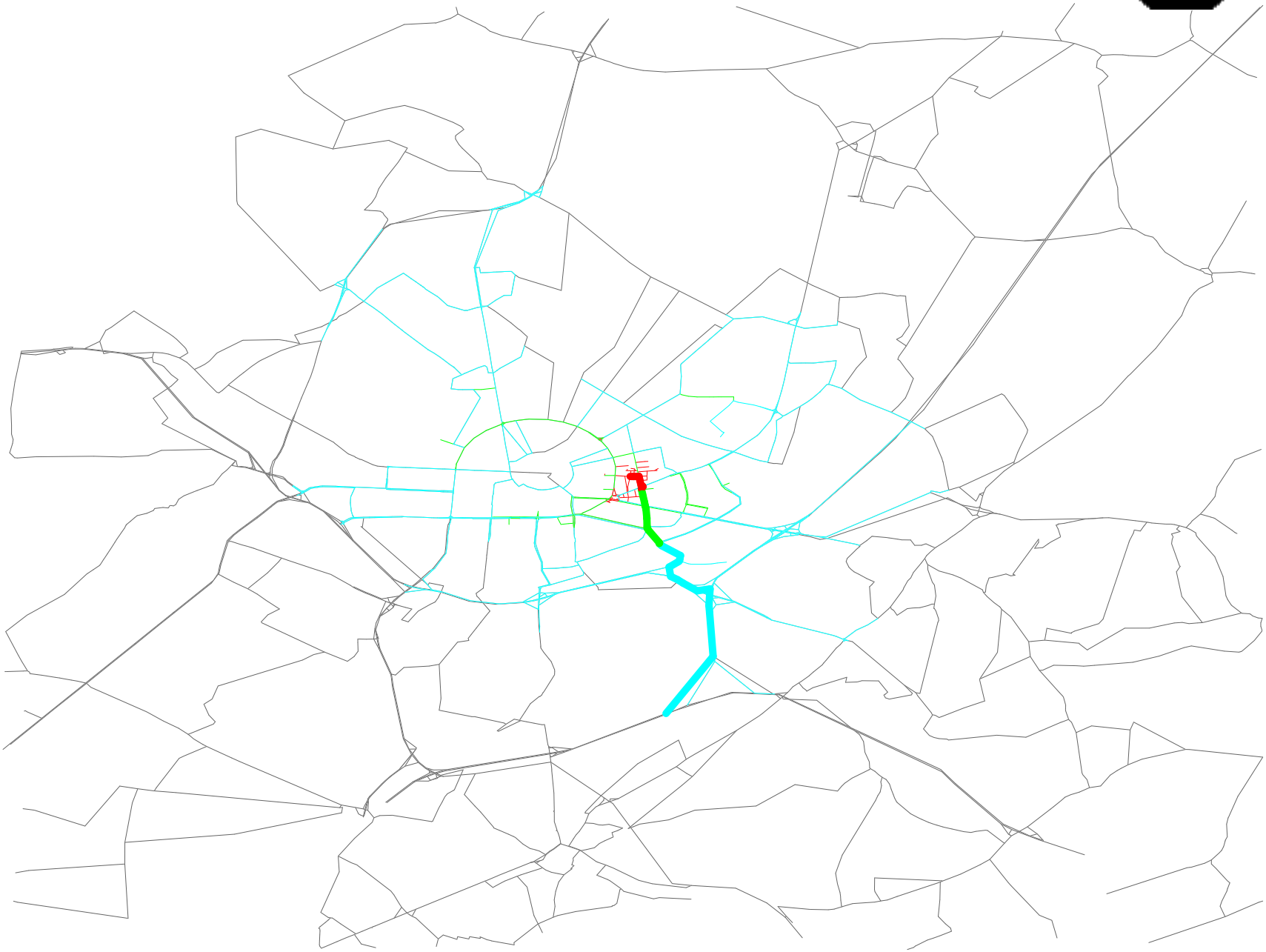
Bounding Box: 20 km Level 1 Search Space



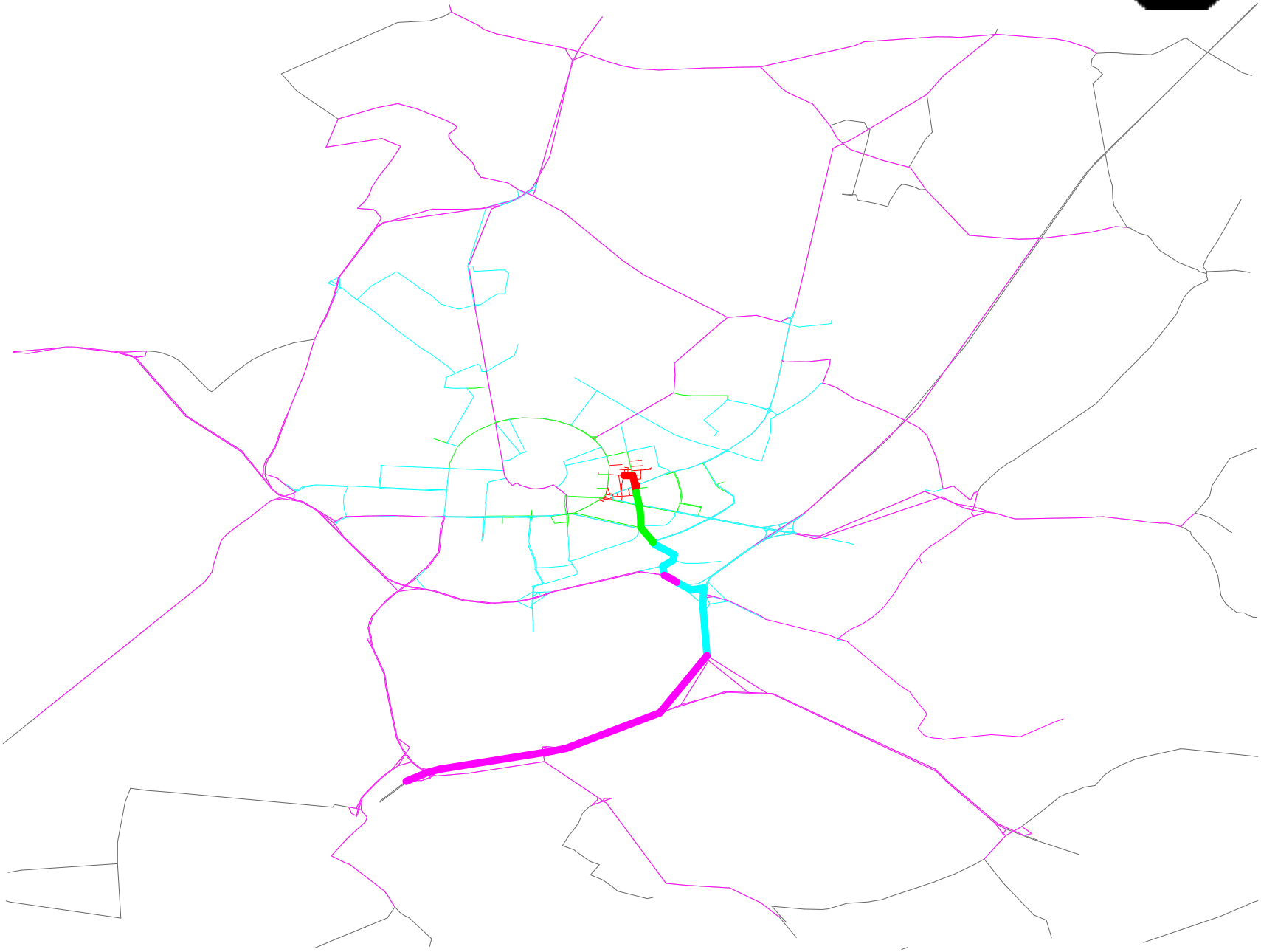
Bounding Box: 20 km Level 2



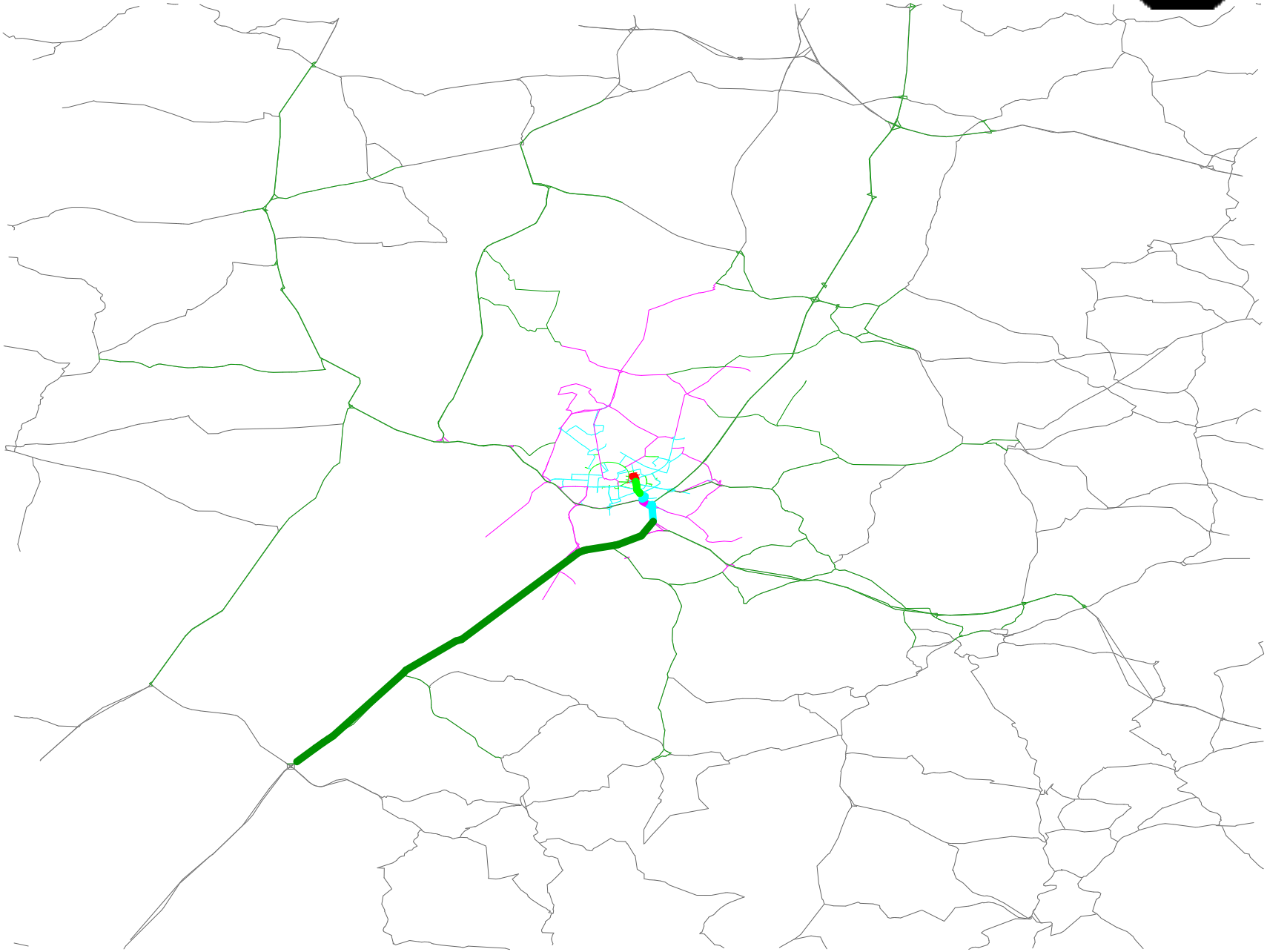
Bounding Box: 20 km Level 2 Search Space



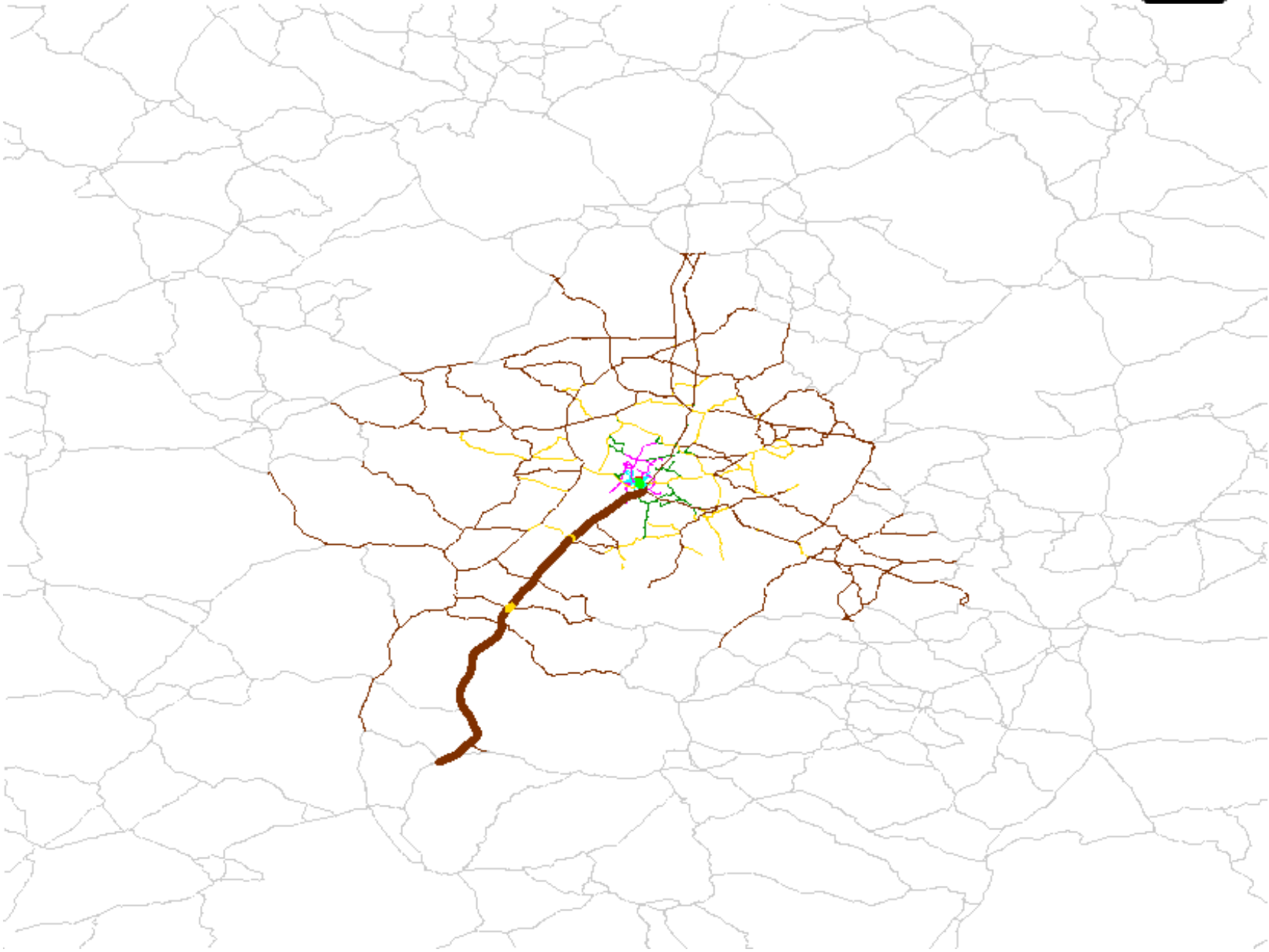
Bounding Box: 20 km Level 3 Search Space



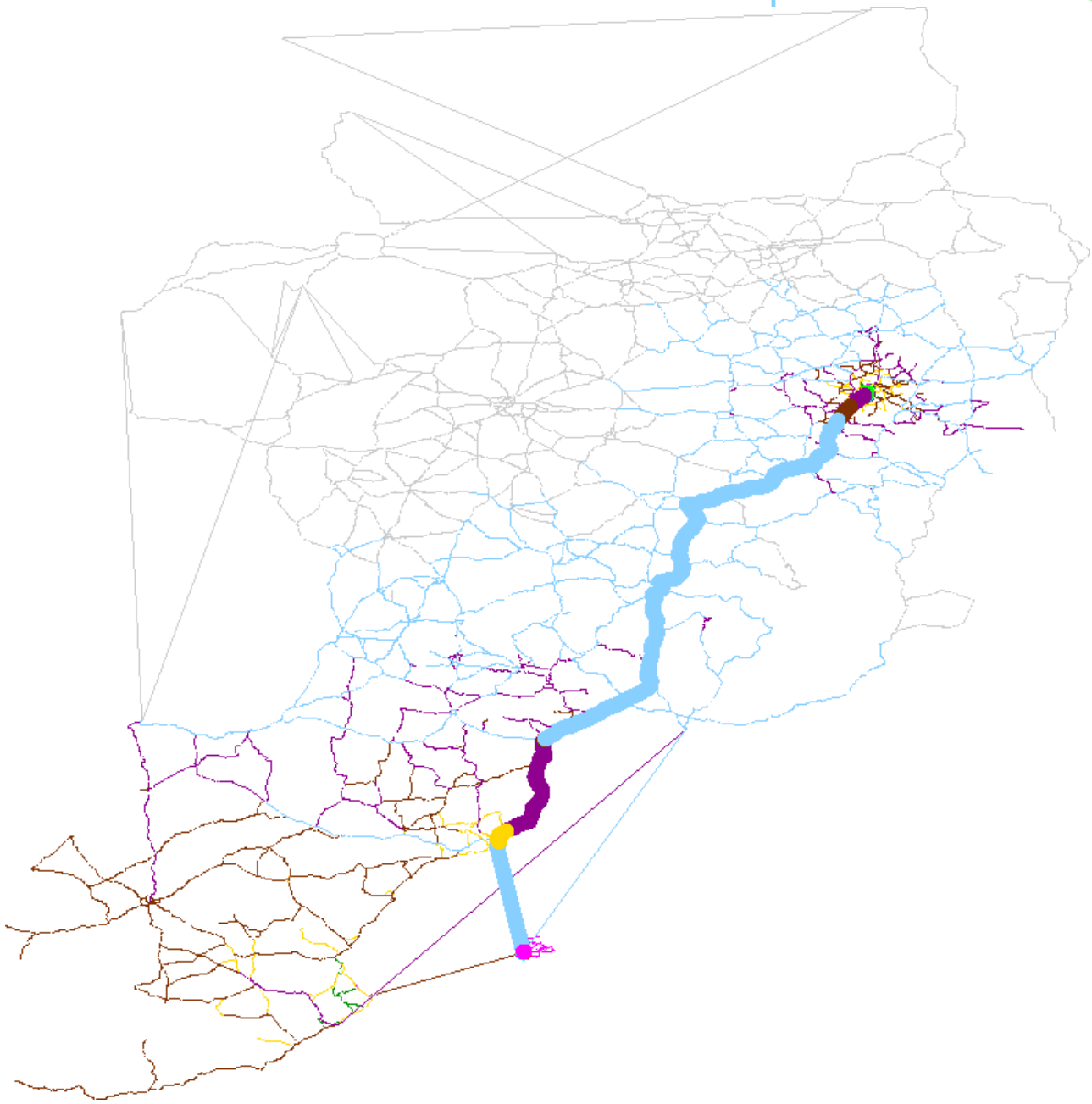
Bounding Box: 80 km Level 4 Search Space



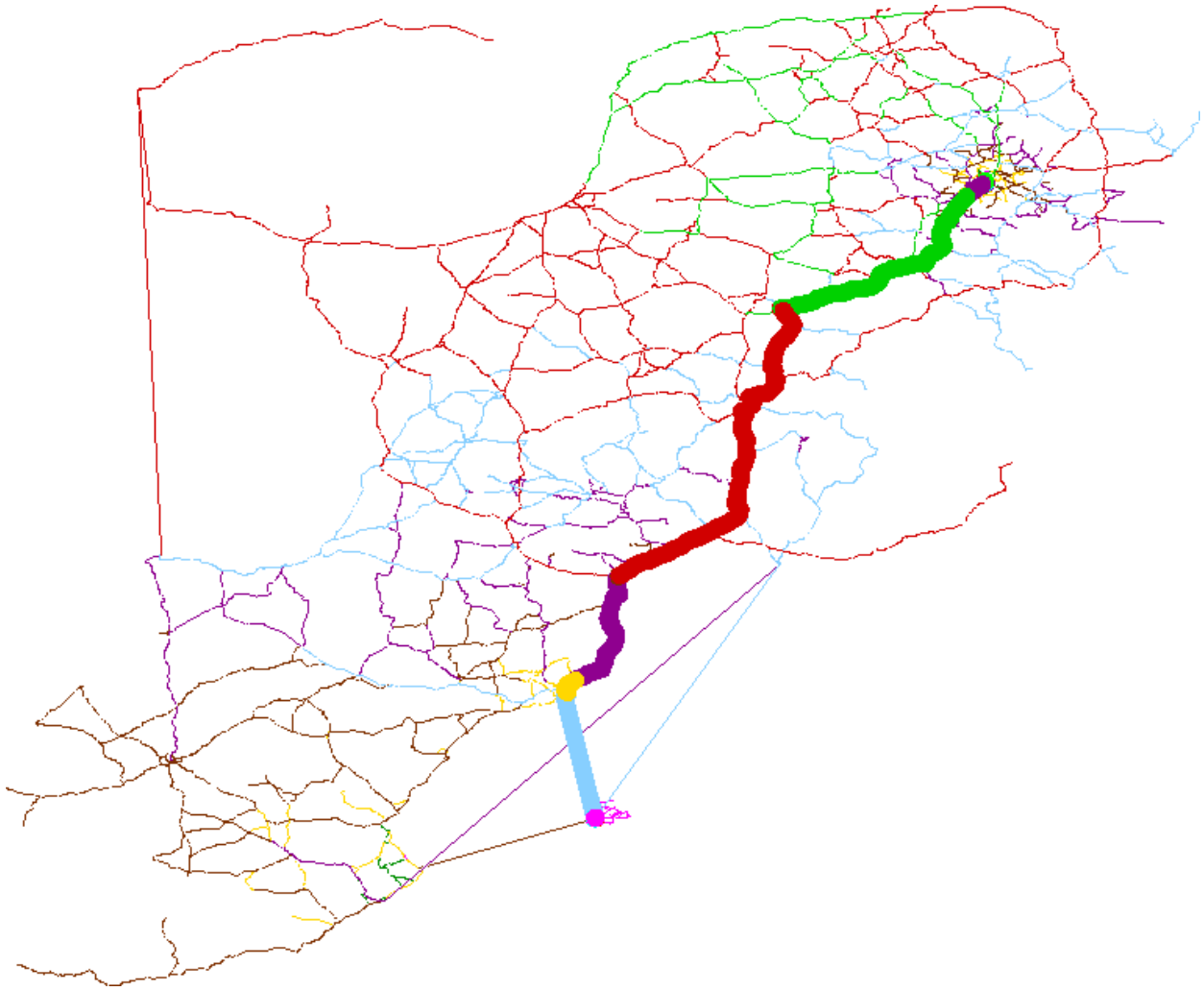
Bounding Box: 400 km Level 6 Search Space



Level 8 Search Space



Level 10 Search Space



Optimisation: Distance Table

Construction:

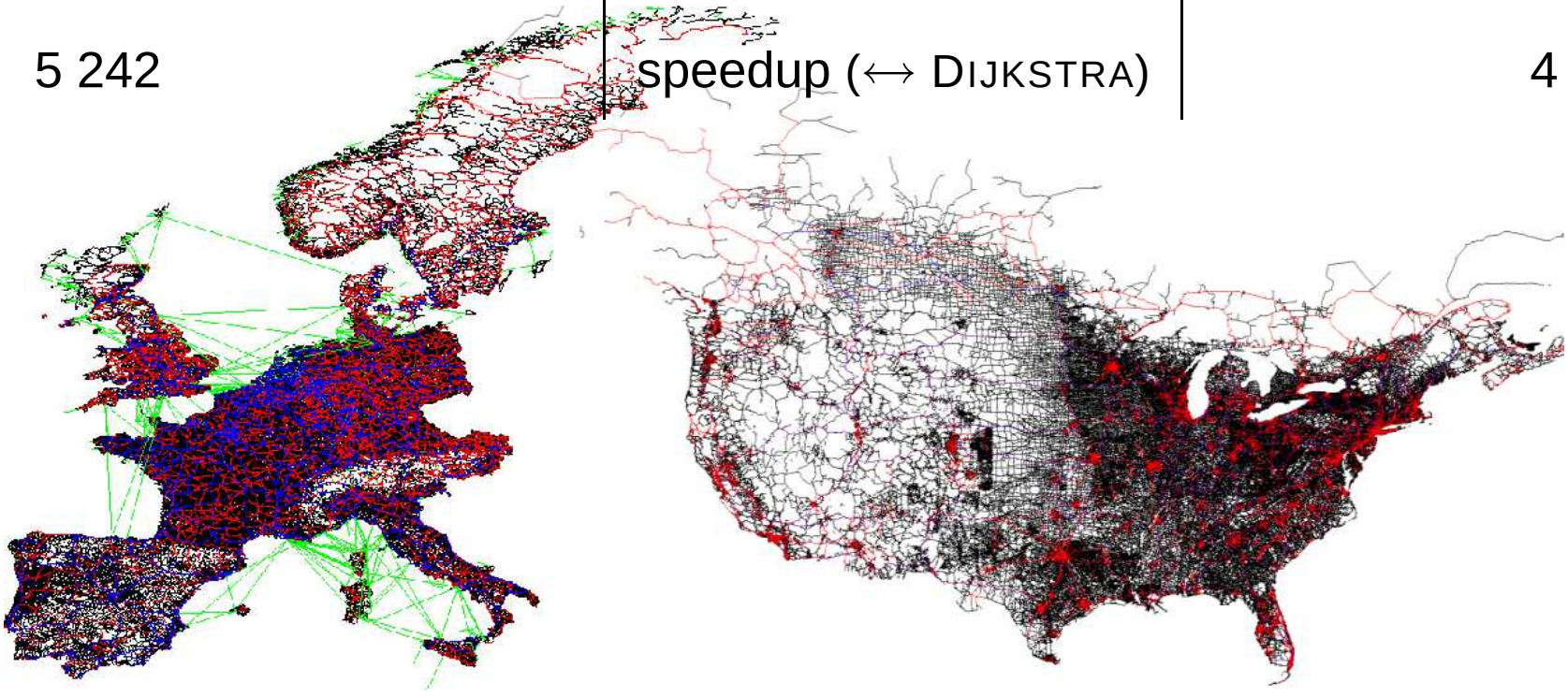
- Construct fewer levels. — e.g. 4 instead of 9
- Compute an all-pairs distance table for the topmost level L. — 8 776 $\times$ 8 776 entries

Query:

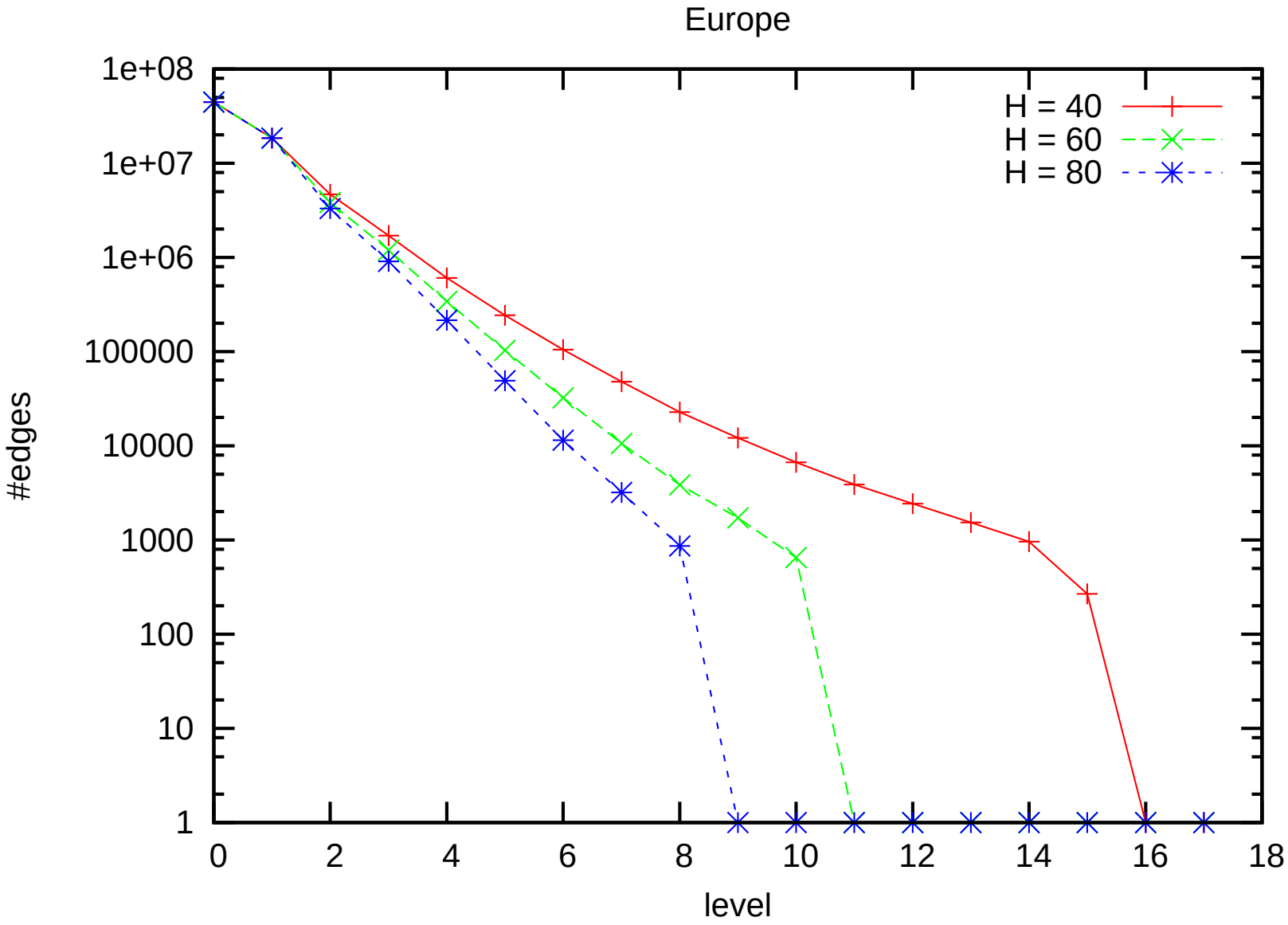
- Abort the search when all entrance points in the core of level L have been encountered. — $\approx$ 70 for each direction
- Use the distance table to bridge the gap. — $\approx$ 70 $\times$ 70 entries

Experiments

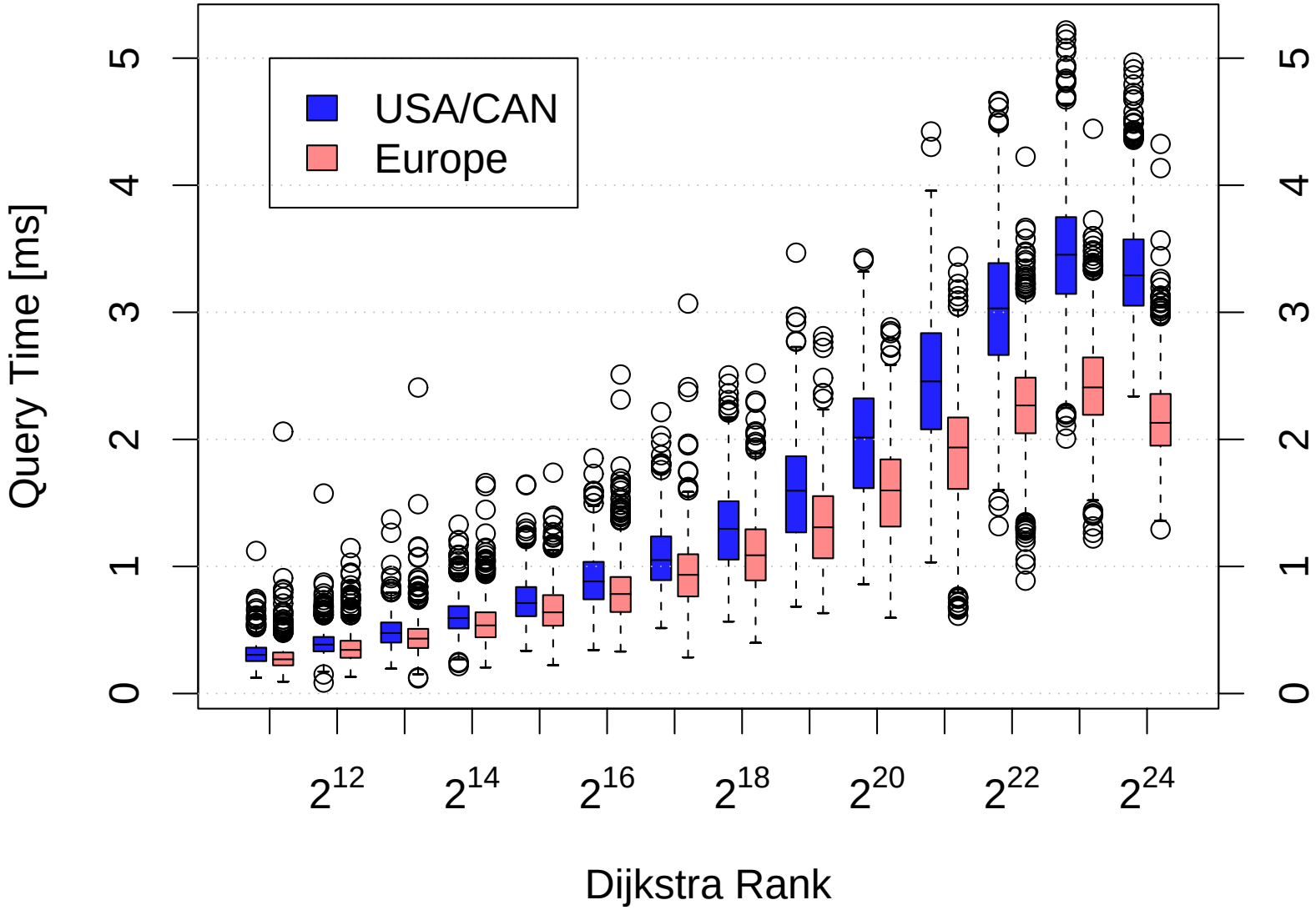
W. Europe (PTV)		USA/CAN (PTV)
18 029 721	#nodes	18 741 705
42 199 587	#directed edges	47 244 849
21	construction [min]	30
1.94	search time [ms]	2.49
5 242	speedup ($\leftrightarrow$ DIJKSTRA)	4 021



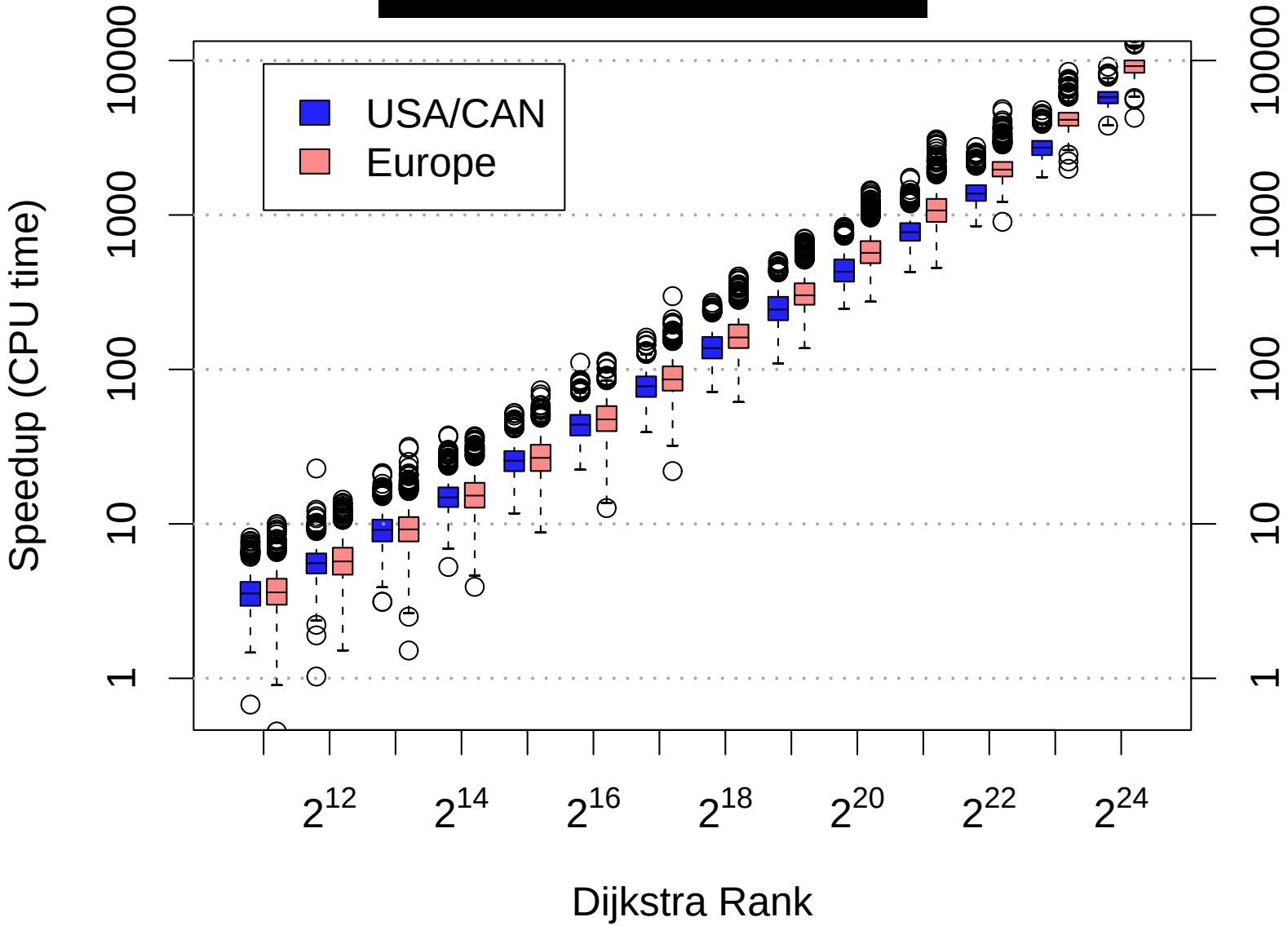
Shrinking of the Highway Networks



Queries – Time



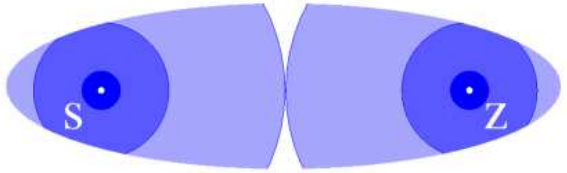
Queries – Speedup



Summary

- exact routes in large street networks — e.g. $\approx$ 18 million nodes
  - $\rightsquigarrow$ quality advantage, advertisement argument
- fast search — $\approx$ 2.5 ms
  - $\rightsquigarrow$ cheap, energy efficient processors in mobile devices
  - $\rightsquigarrow$ low server load
  - $\rightsquigarrow$ lots of room for additional functionality
- fast preprocessing — $\approx$ 30 min
- low space consumption — $\ll$ data base
- no manual postprocessing of data
  - $\rightsquigarrow$ less dependence on data sources
- organic enhancement of existing commercial solutions

Future Work



- combination with goal directed approaches
- fast, local updates on the highway network

  (e.g. for traffic jams)

- Implementation for mobile devices

  (flash access . . .)

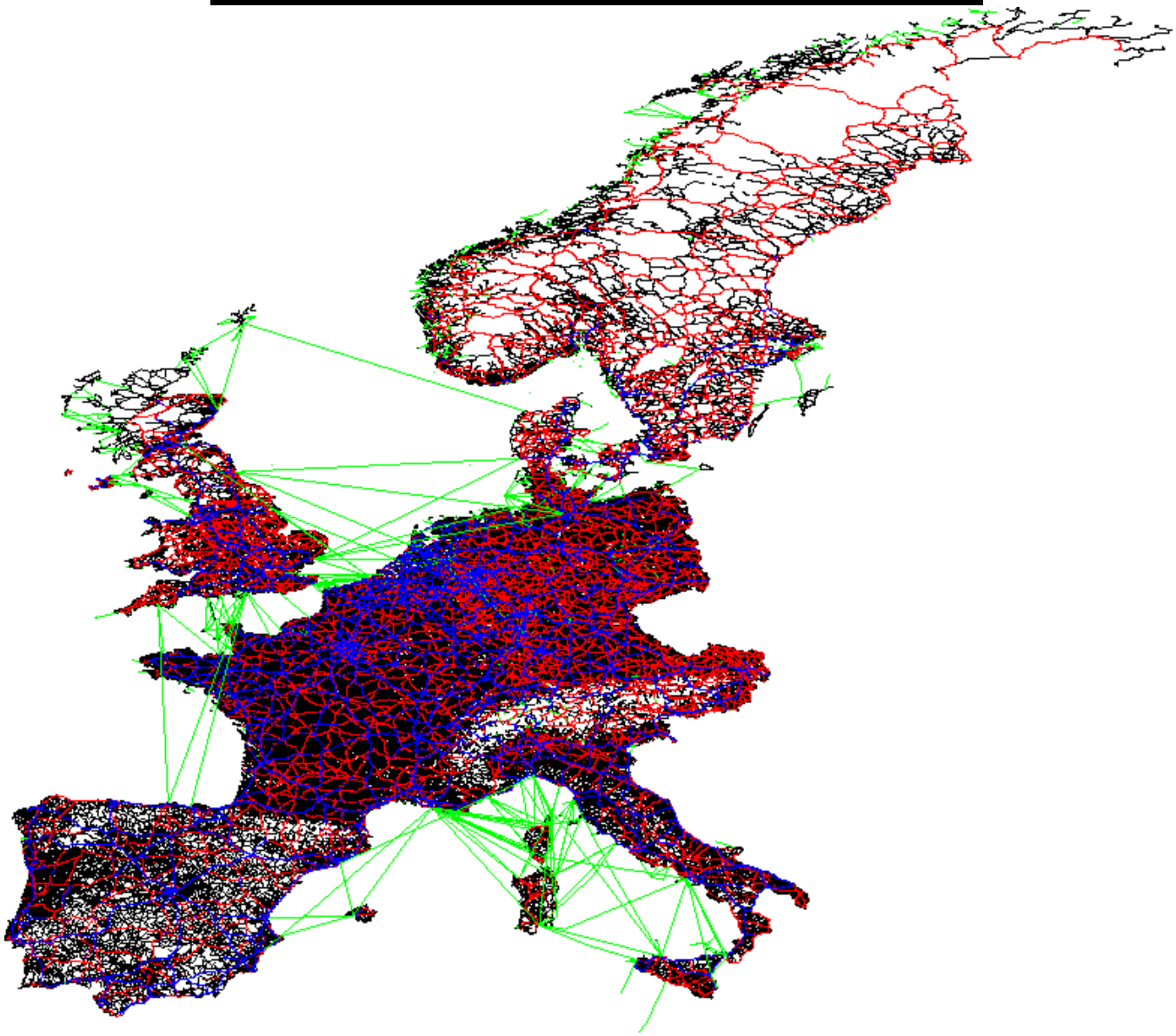
- Flexible objective functions
- . . .

Industrial Cooperations

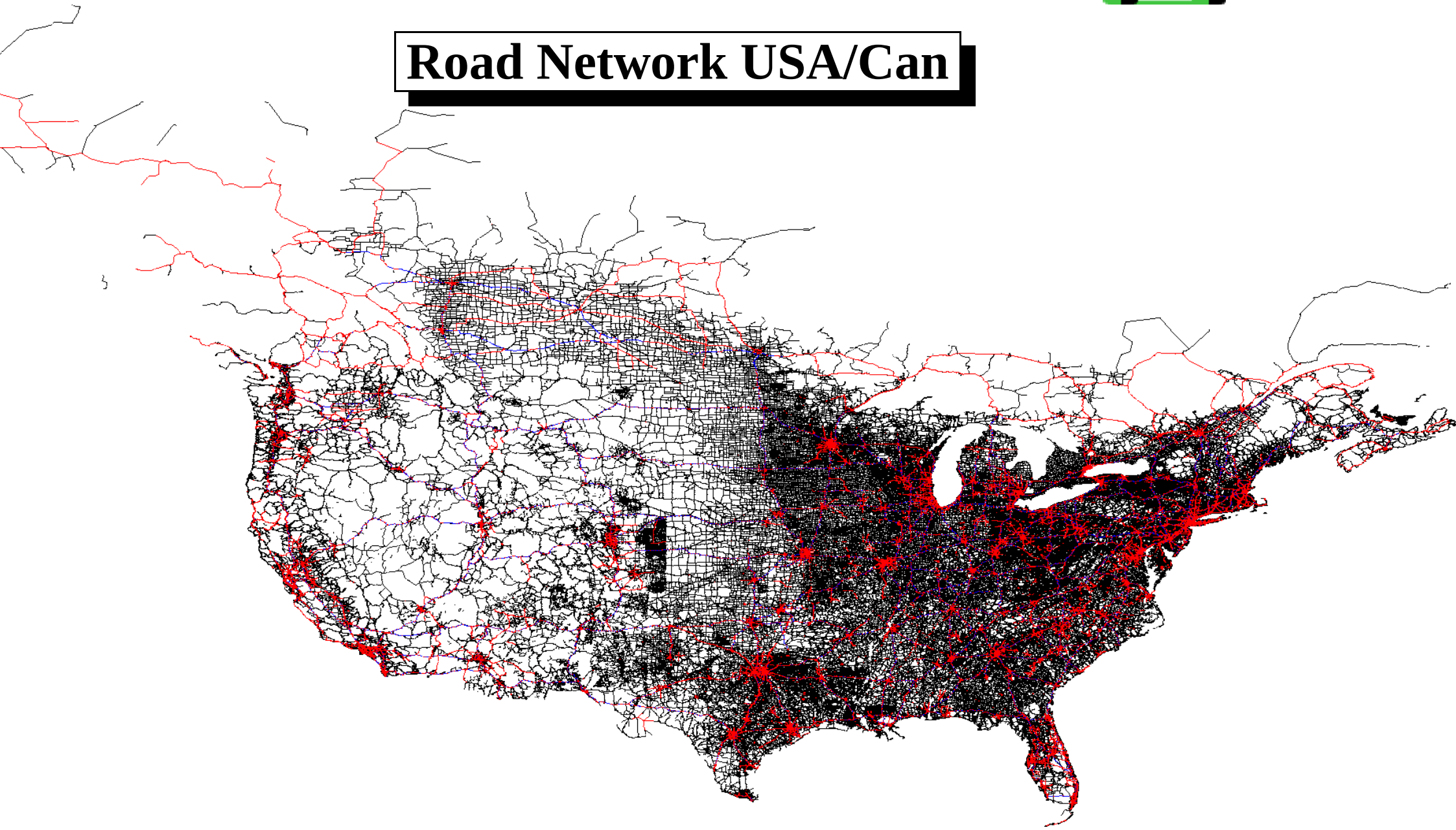
- We help transforming technology into products: consulting . . .
- Joint projects for further features

. . .

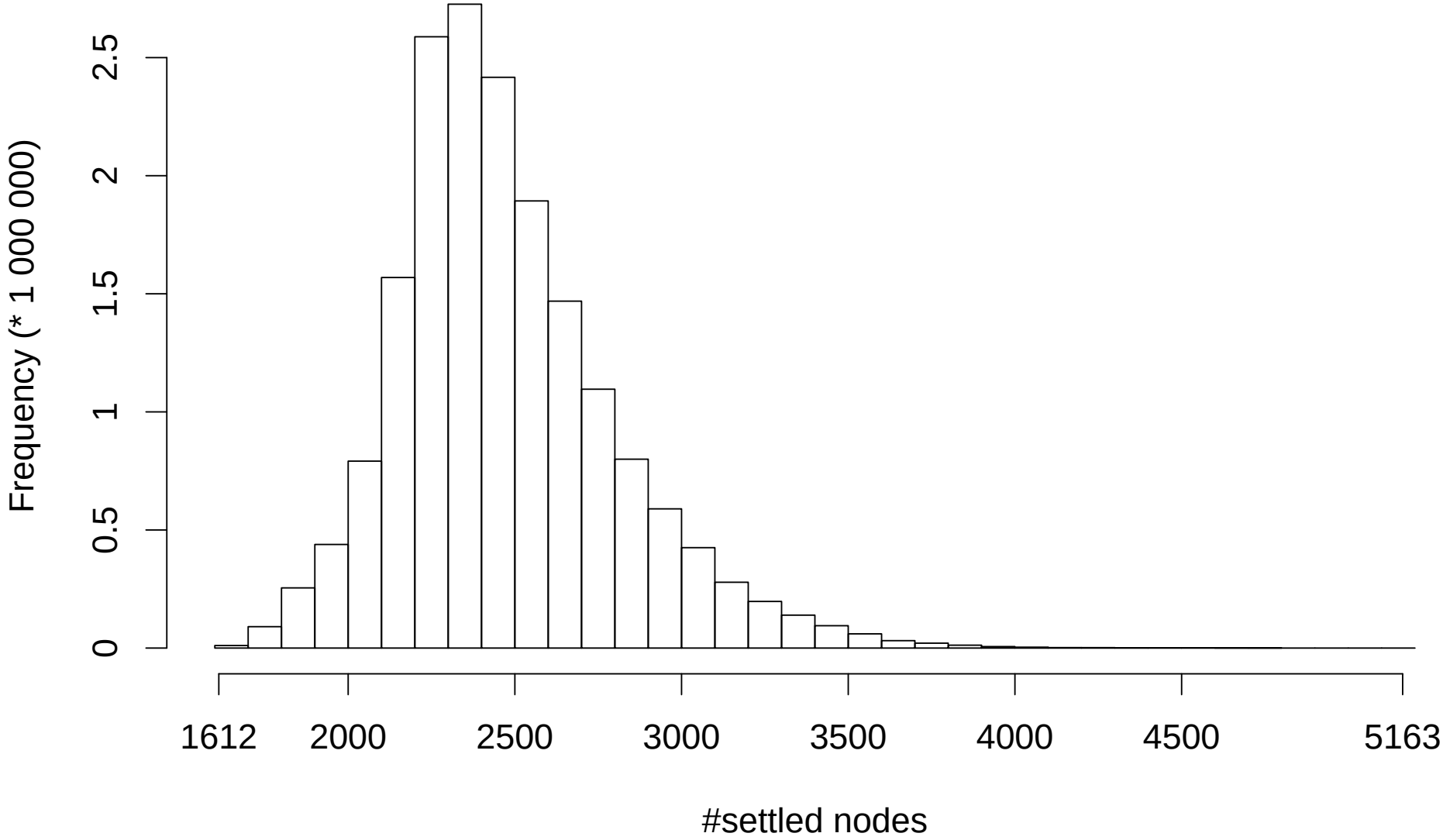
Road Network of Europe



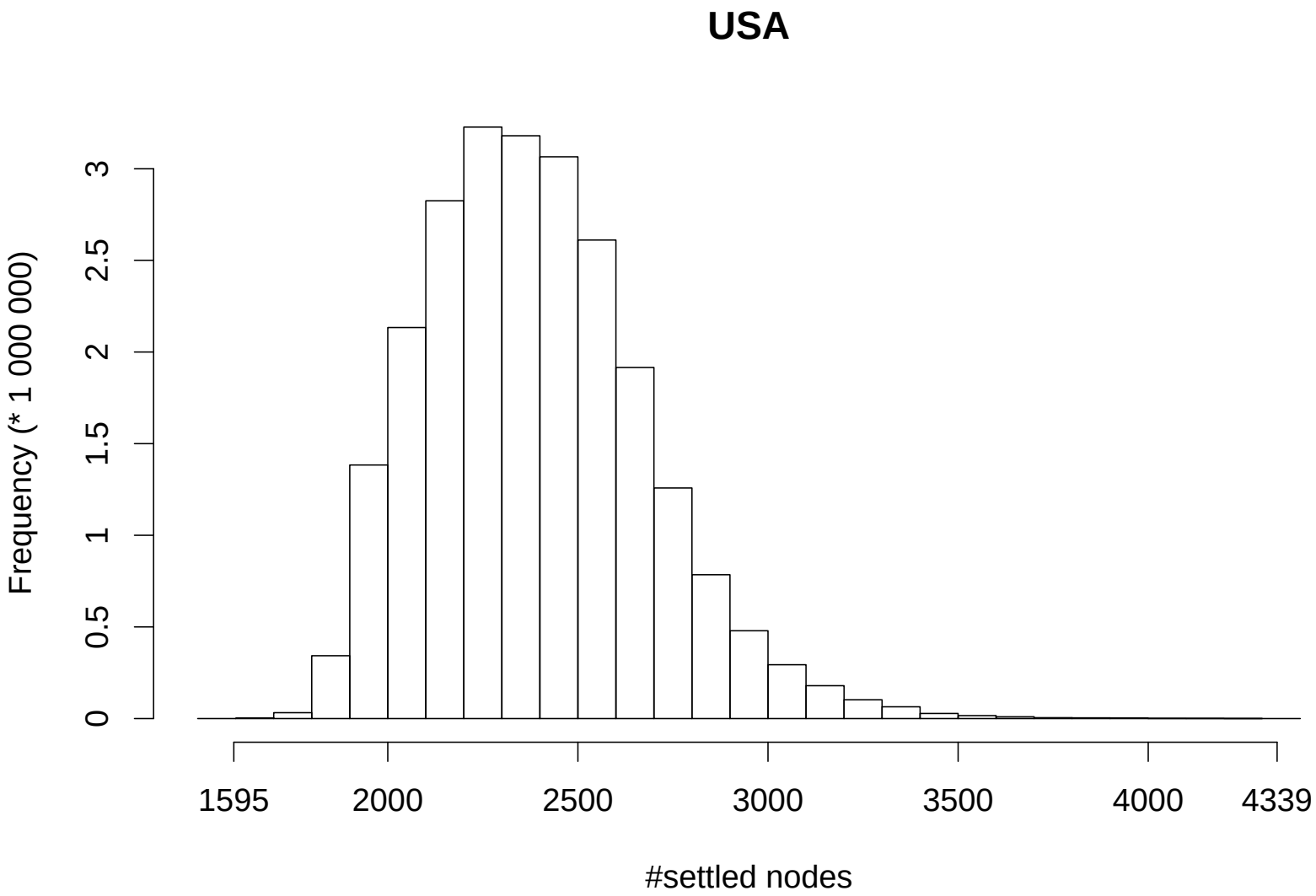
Road Network USA/Can



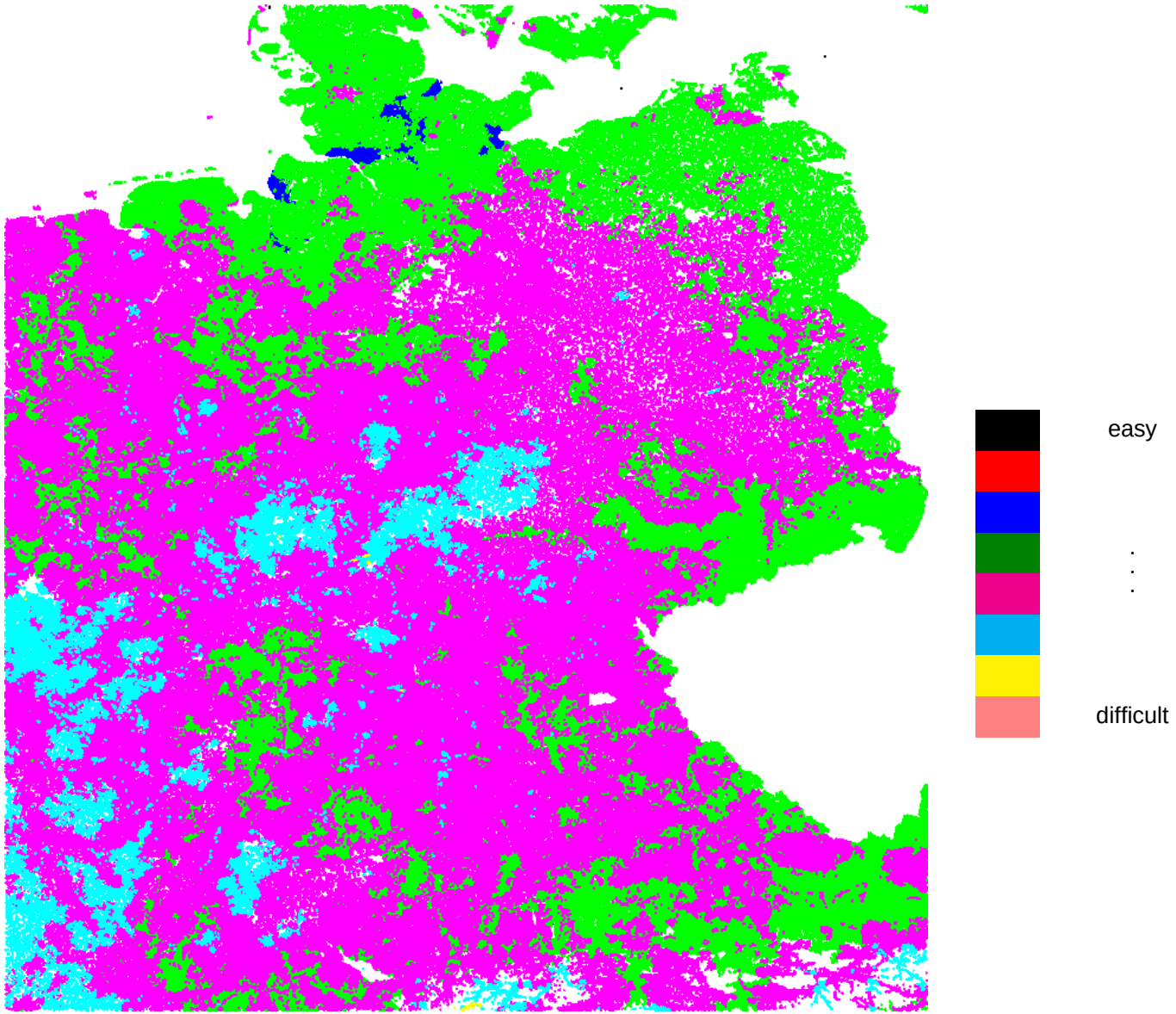
Worst Case Costs



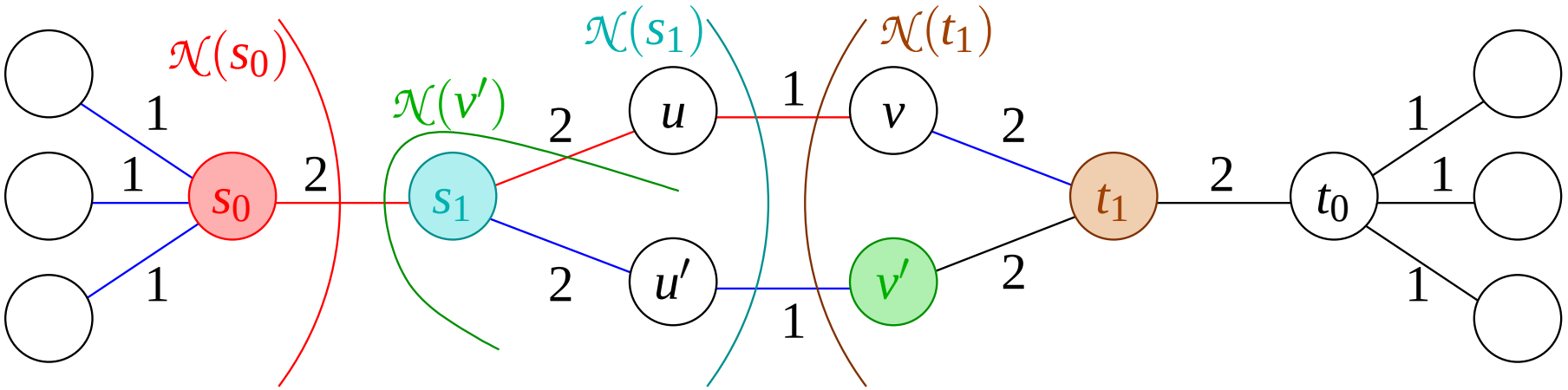
Unidirectional Search – Histogram



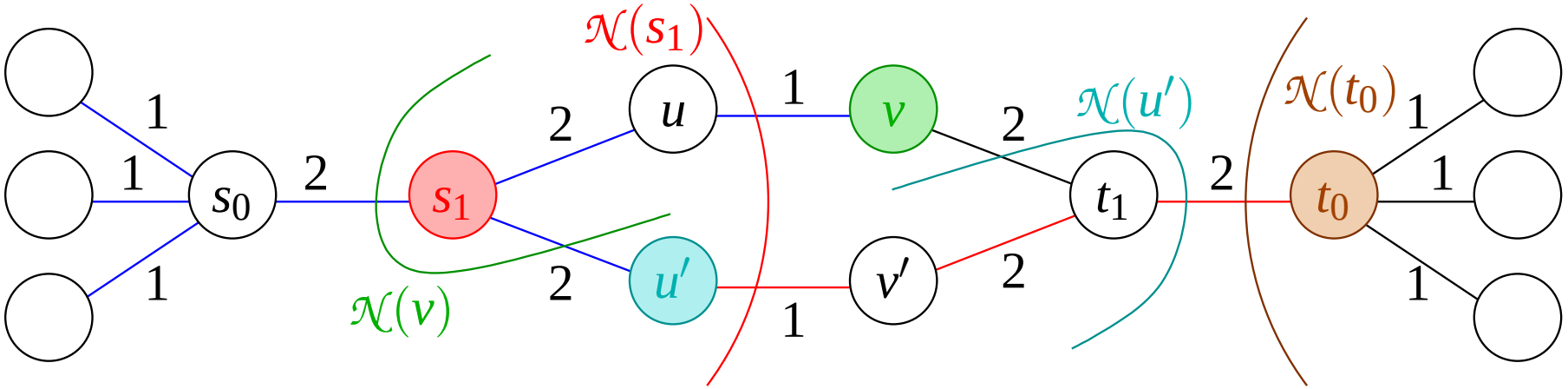
Unidirectional Search – Costs By Region



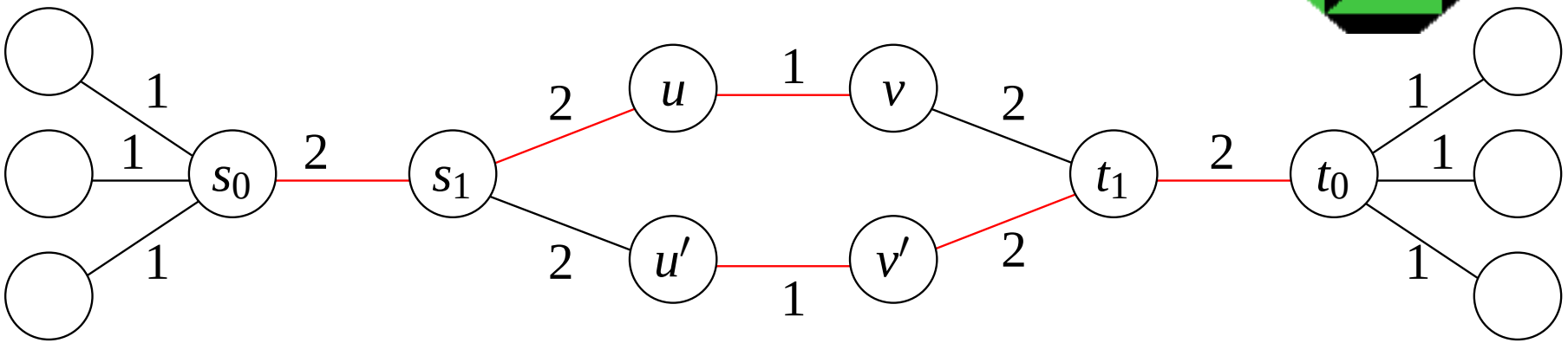
Canonical Shortest Paths



(a) Construction, started from s_0.



(b) Construction, started from s_1.



(c) Result of the construction.