Highway Hierarchies Hasten Exact Shortest Path Queries

Peter Sanders and **Dominik Schultes**

Universität Karlsruhe

October 2005

How do I get there from here ?

Applications

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

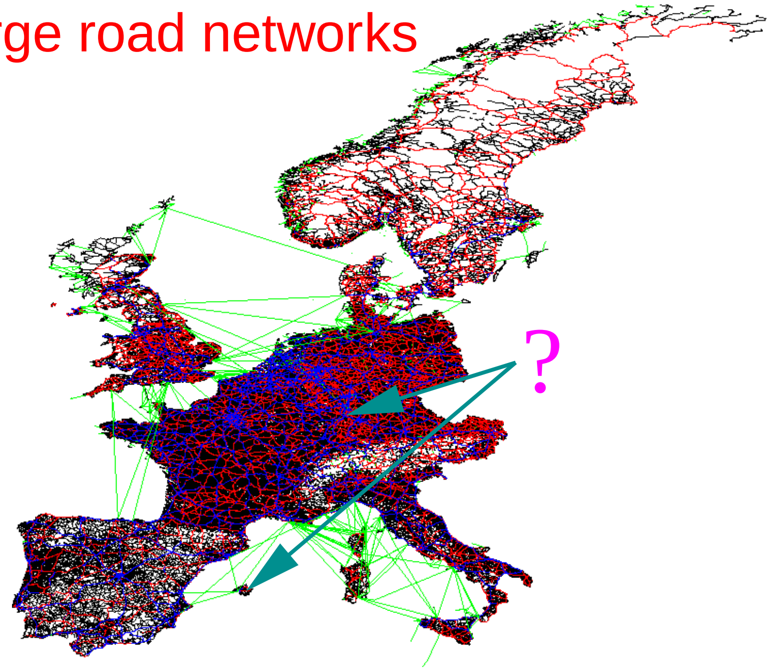
- car navigation systems

- ...

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

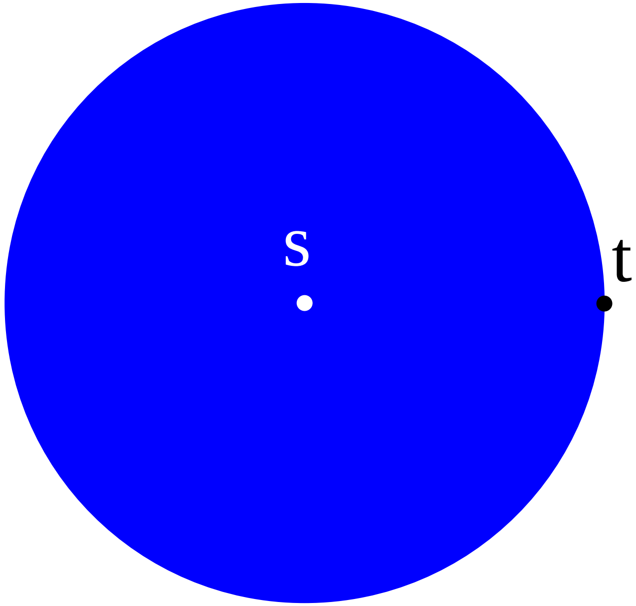


Related Work

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

DIJKSTRA's Algorithm

Dijkstra



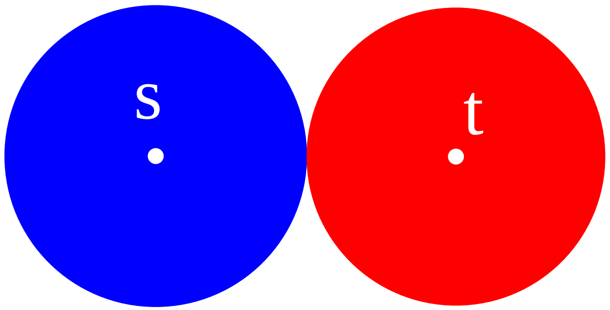
not practicable

for large road networks

(e.g. Western Europe:

≈ 18 000 000 nodes)

bidirectional Dijkstra



improves the running time,

but still too slow

Commercial Systems

1. Search from the source and target node ('bidirectional')
   within a certain radius (e.g. 20 km),
   consider all roads

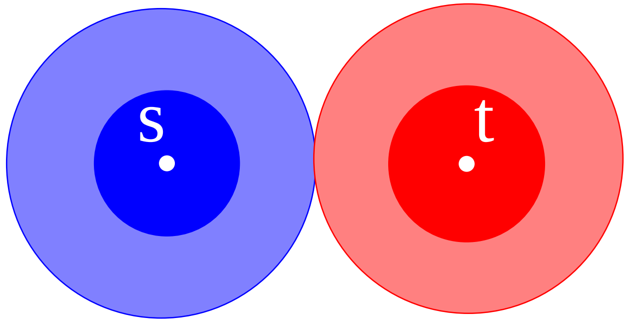
2. Continue the search within a larger radius (e.g. 100 km),
   consider only national roads and motorways

3. Continue the search,
   consider only motorways

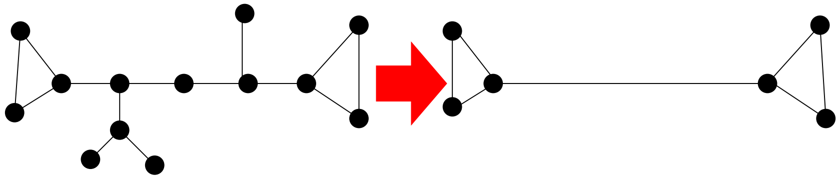
fast, but not exact

Exact Highway Hierarchies

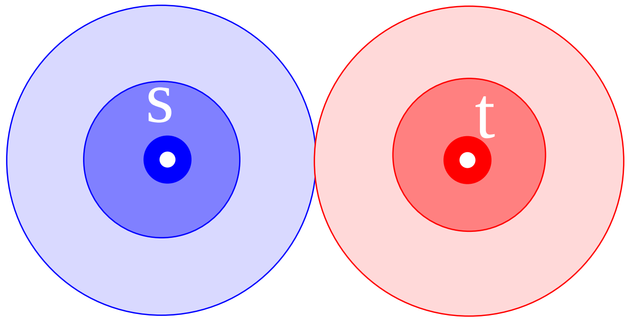
- complete search within a local area
- search in a (thinner) highway network



= minimal graph that preserves all shortest paths



- contract trees and lines
- iterate $\rightsquigarrow$ highway hierarchy



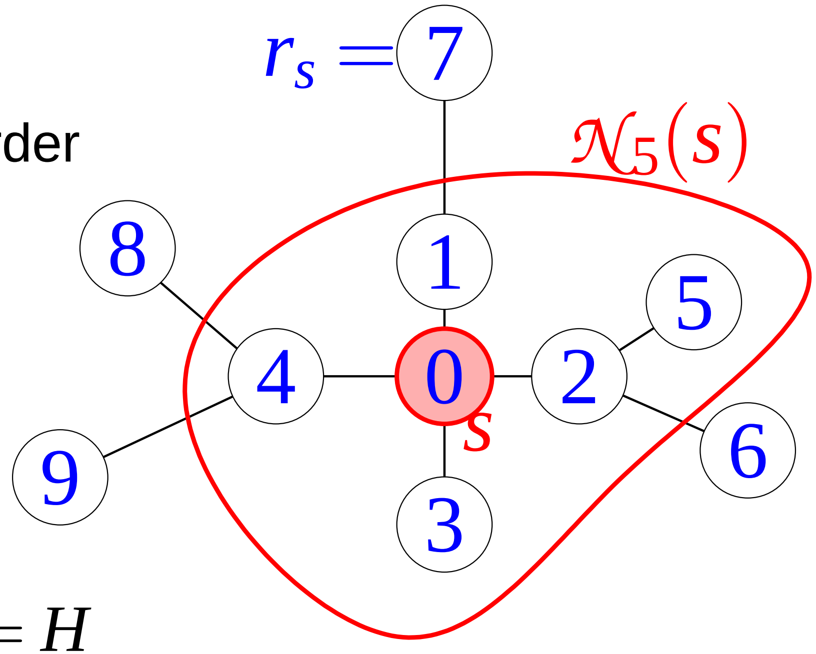
A Meaning of “Local”

- Dijkstra’s Algorithm from node s

  $\rightsquigarrow$ nodes are settled in a fixed order

  Dijkstra rank $r_s(v)$ of node v

  = rank w.r.t. this order



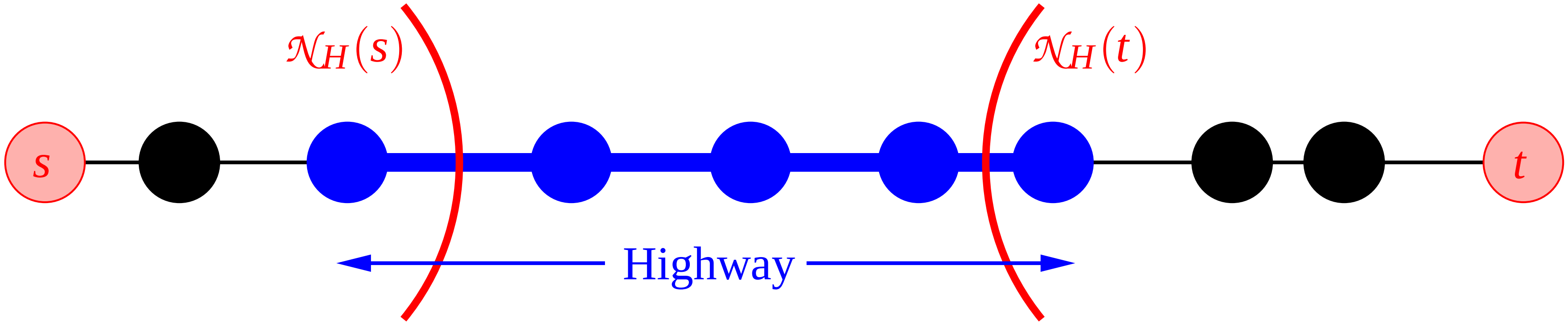
- For a parameter H,

  $d_H(s) := d(s,u)$, where $r_s(u) = H$

- H-neighbourhood of s:

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

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

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

and

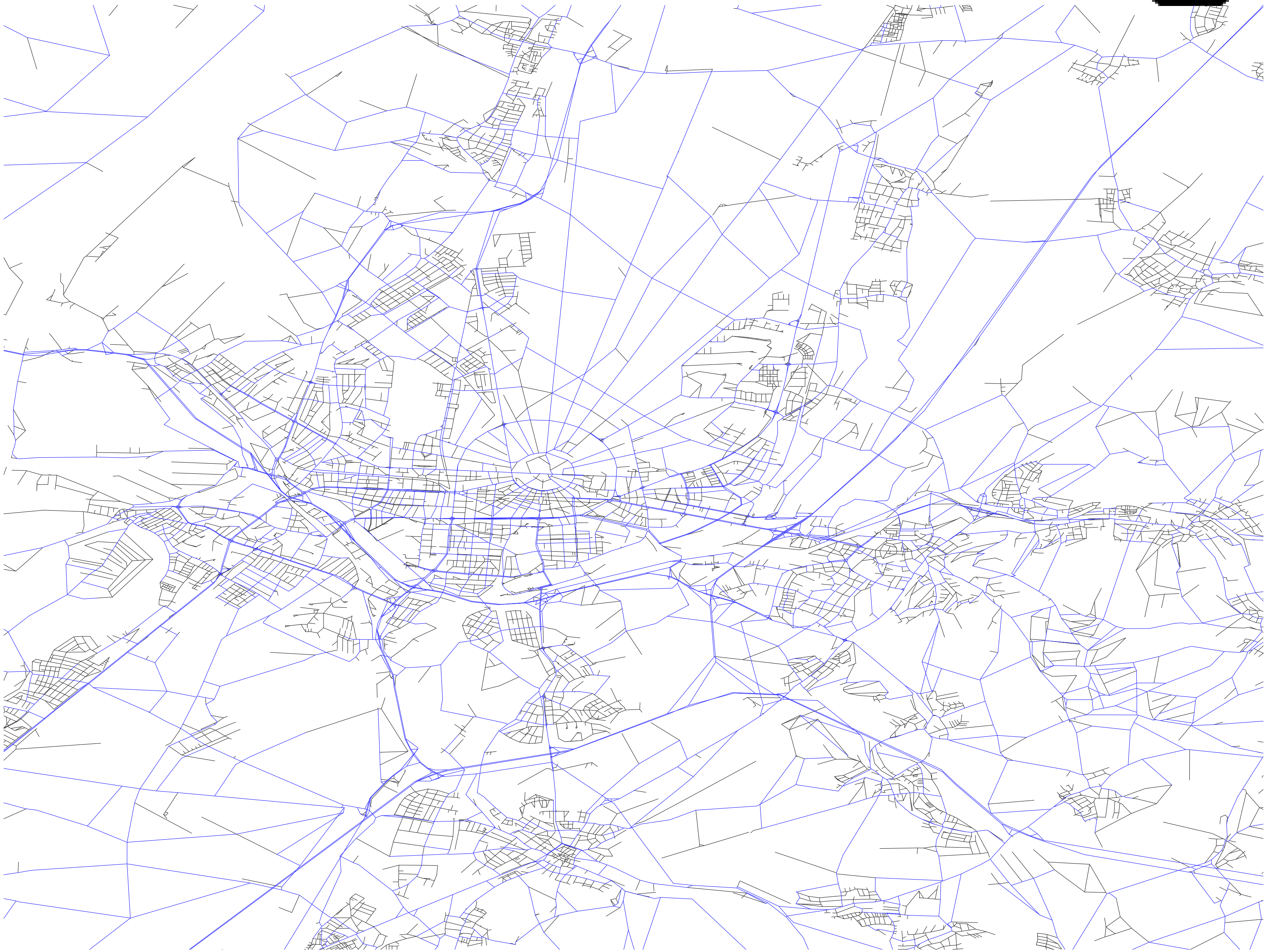
- $u \notin \mathcal{N}_H(t)$

Construction

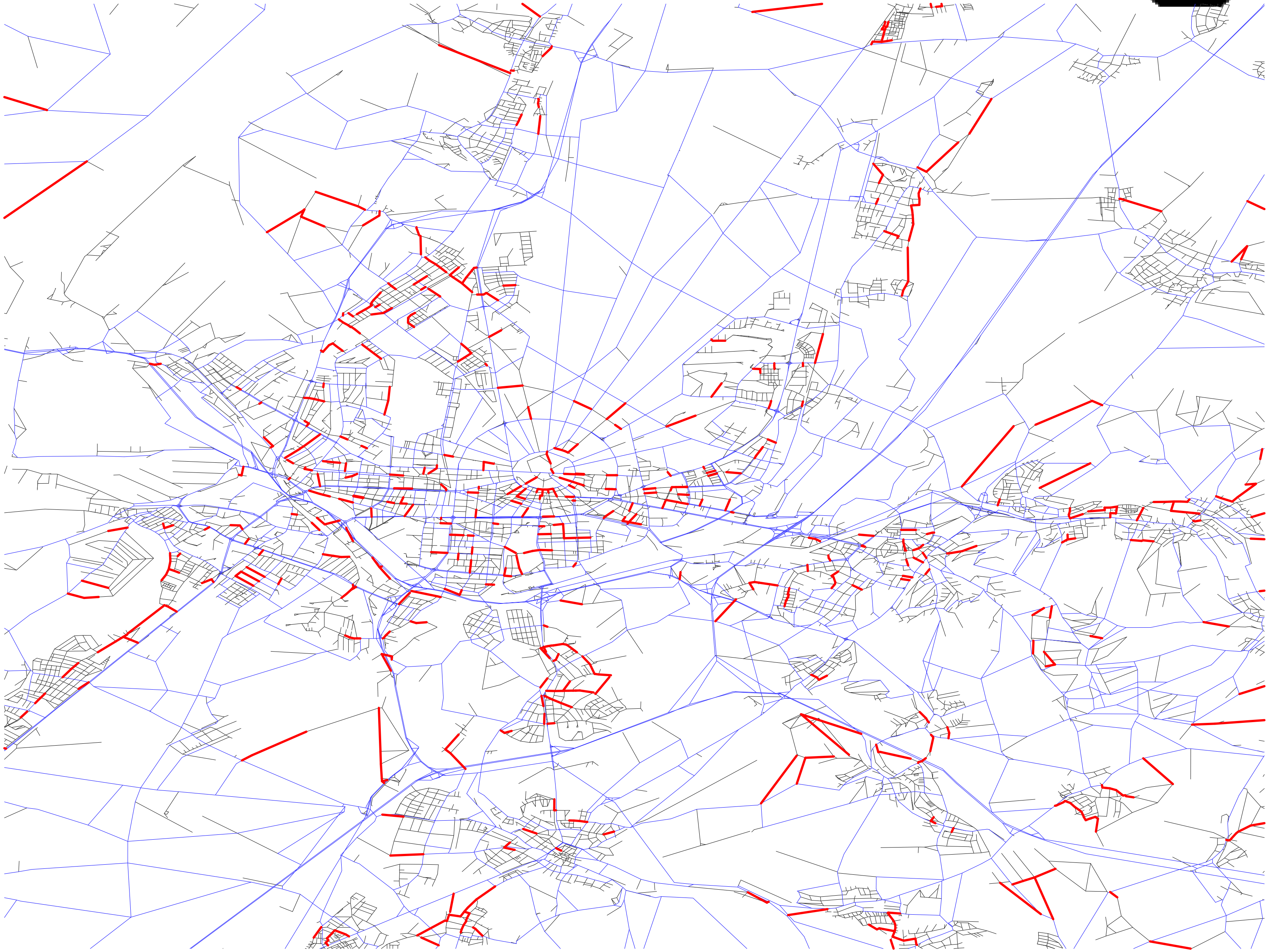
Example: Western Europe, bounding box around Karlsruhe

Complete Road Network

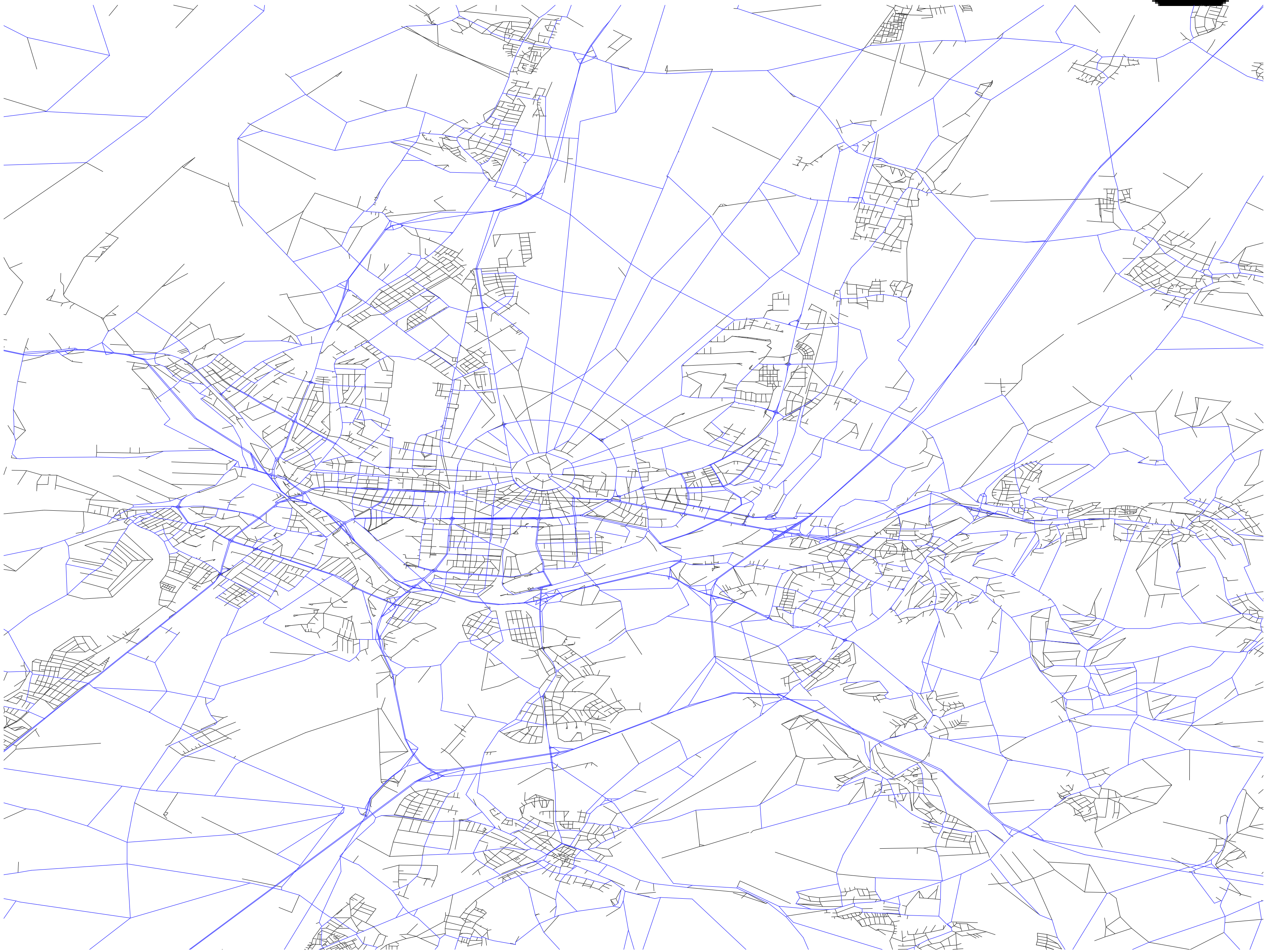
Highway Network



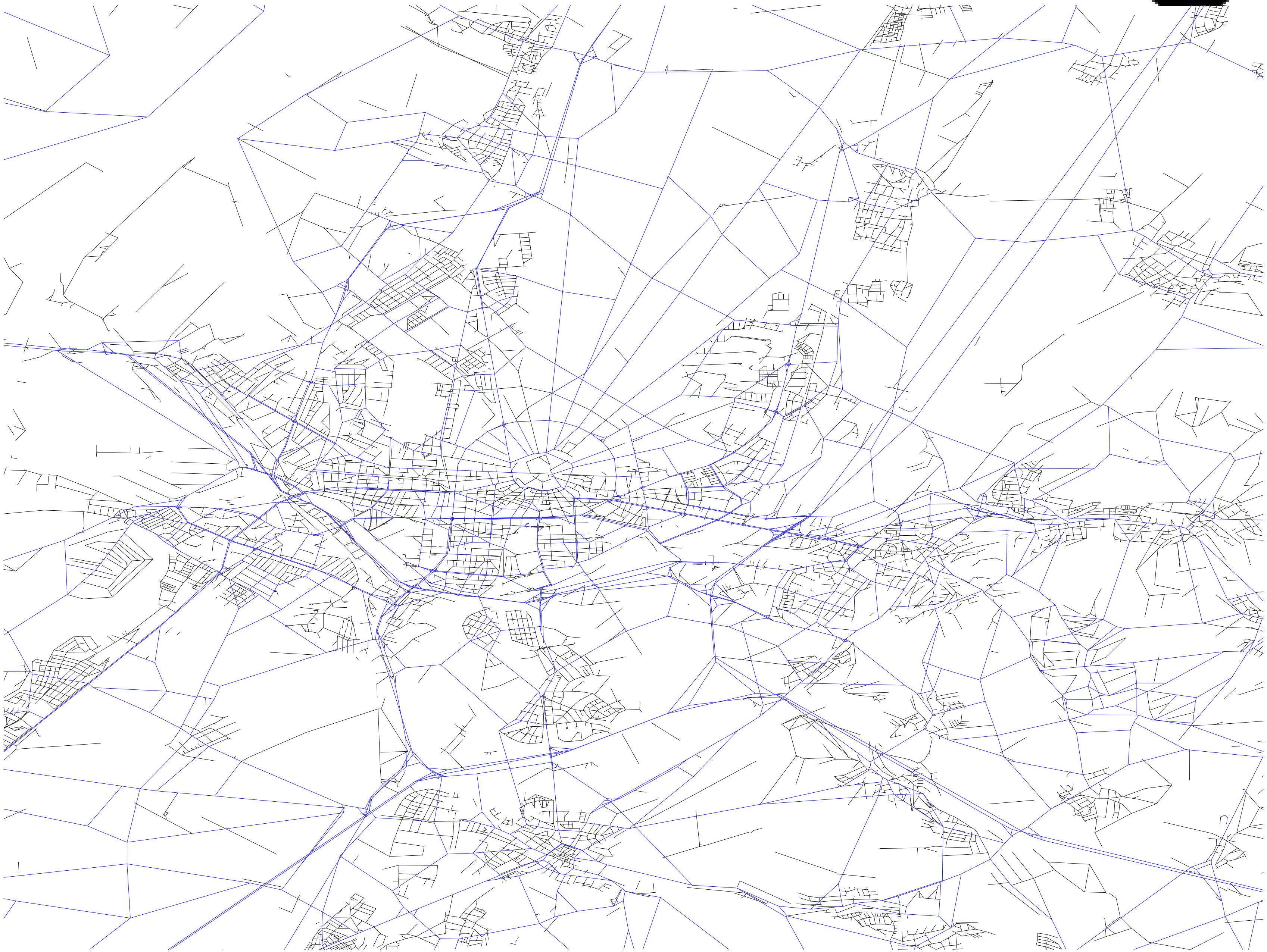
Attached Trees



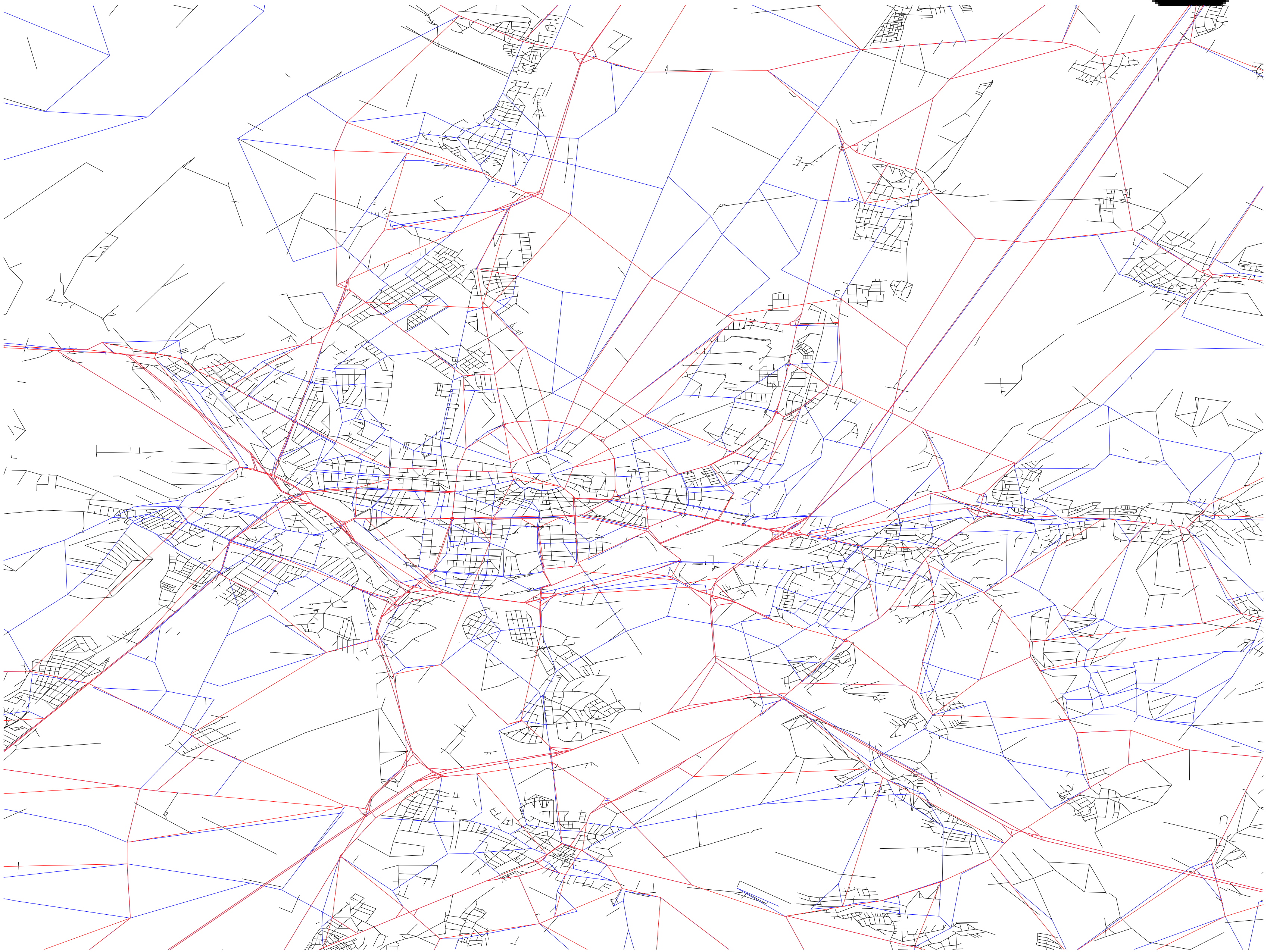
2-Core



Contract Lines



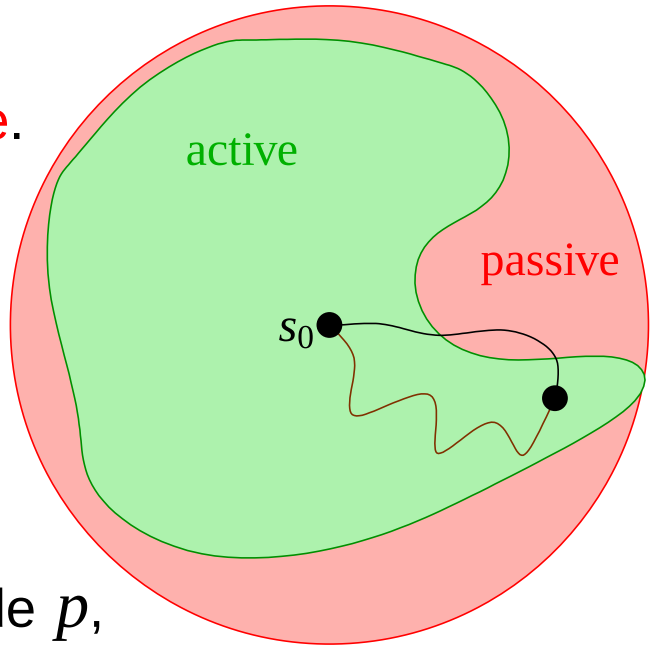
Highway Hierarchy: Level 1 and Level 2



Fast Construction

Phase 1: *Construction of Partial Shortest Path Trees*

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

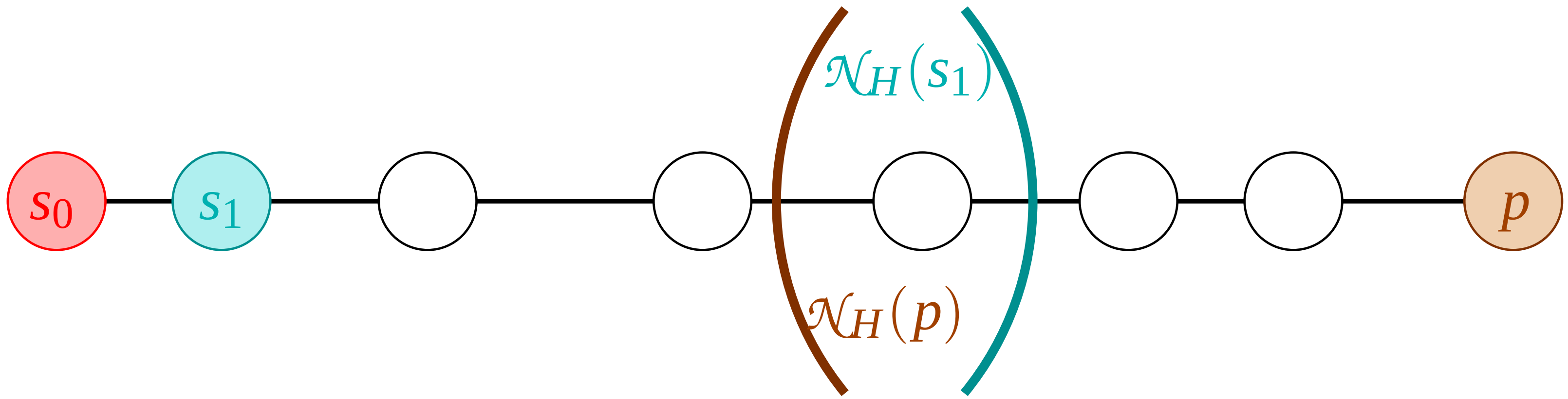


- A node's state is either active or passive.
- s_0 is active.
- A node inherits the state of its parent in the shortest path tree.
- If the abort condition is fulfilled for a node p, p's state is set to passive.

The search is aborted when all queued nodes are passive.

Fast Construction

Abort Condition:



p is set to passive *iff*

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

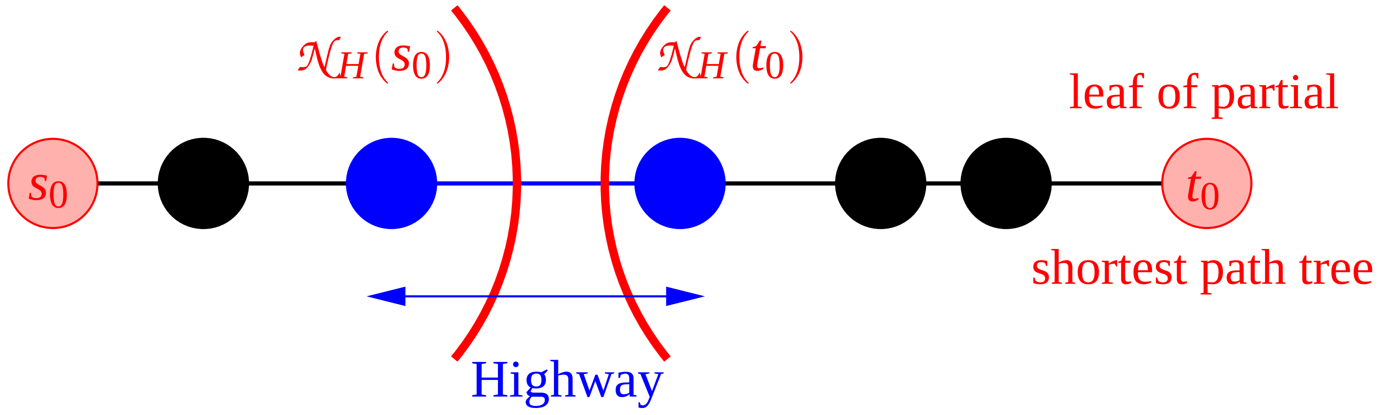
Fast Construction, Phase 2

Theorem:

The tree roots and leaves encountered in Phase 1

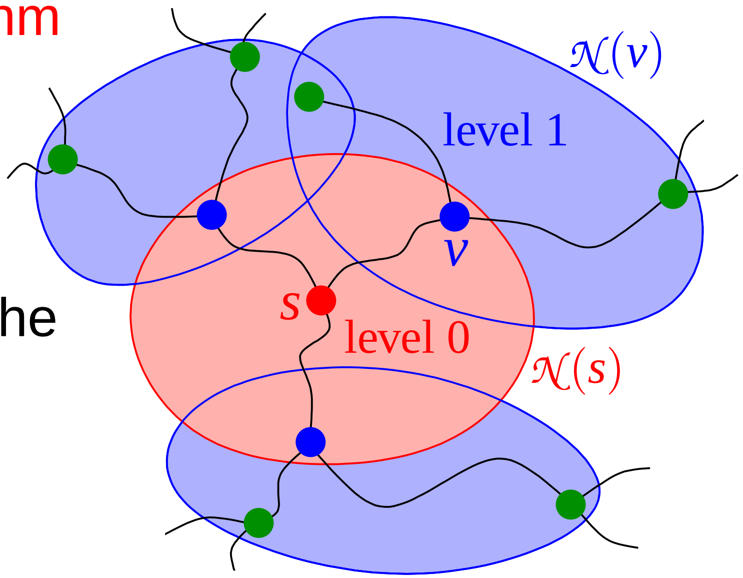
witness all highway edges.

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



Query

Bidirectional version of Dijkstra's Algorithm

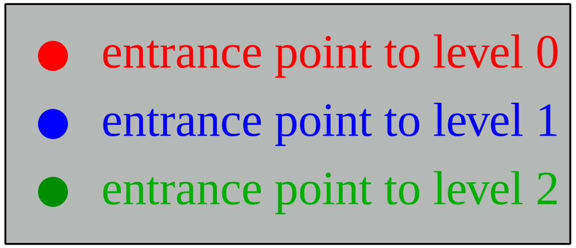


Restrictions:

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

  Instead: switch to the next level.

- Do not enter a tree or a line.



Query

Theorem:

We still find the shortest path.

Query

Example: from Karlsruhe, Am Fasanengarten 5

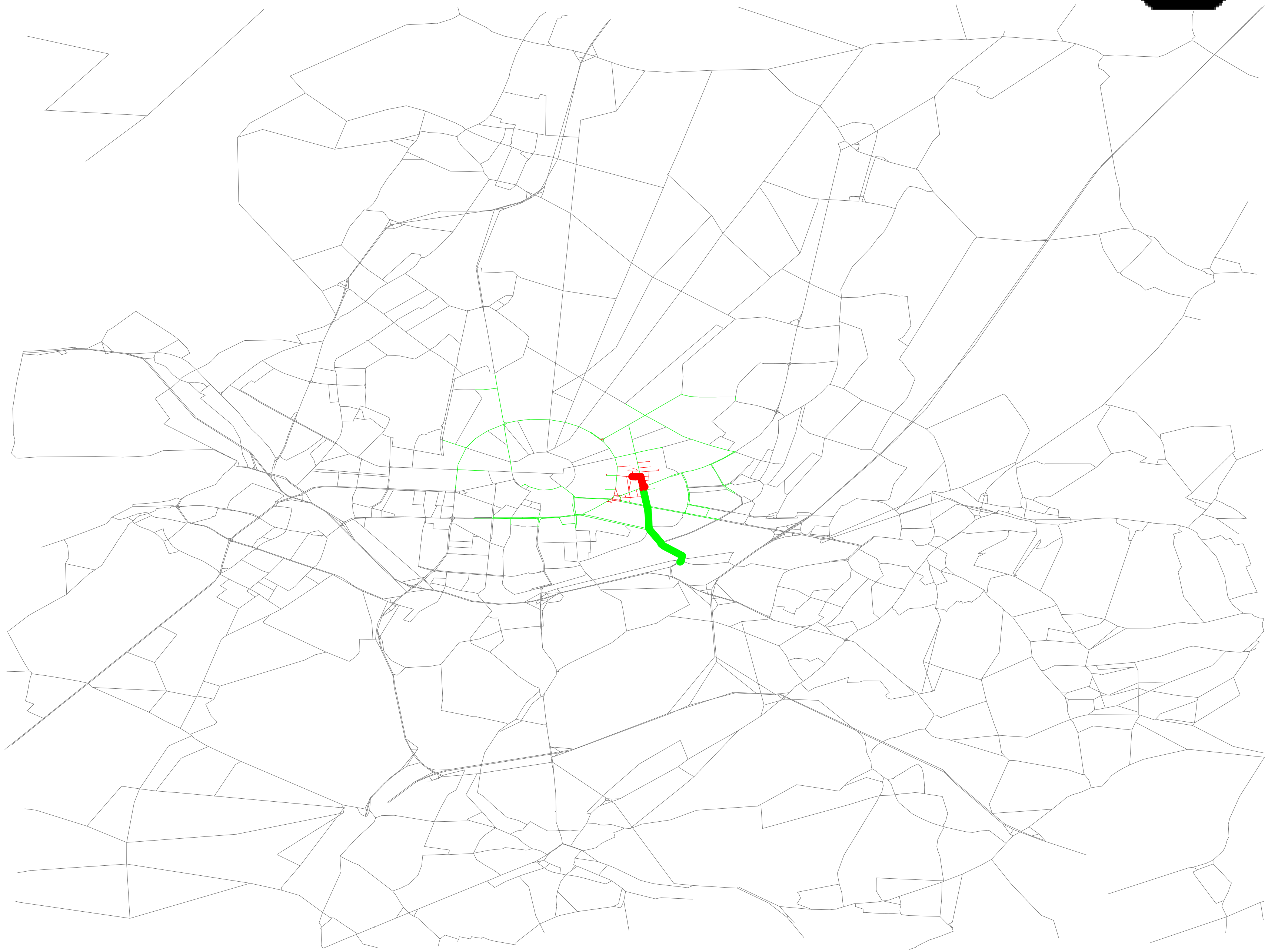
to Palma de Mallorca

Bounding Box: 20 km Level 0

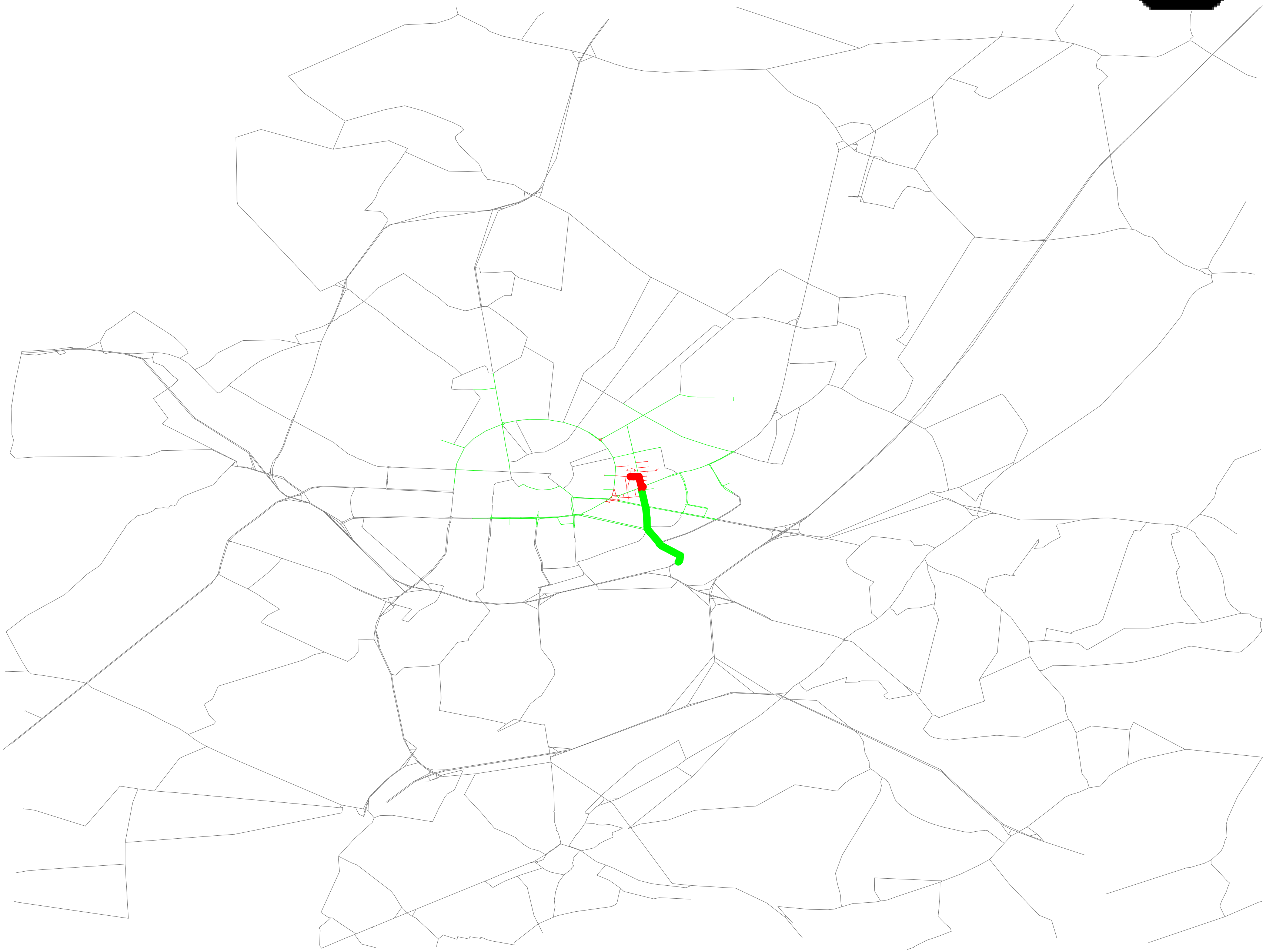
Bounding Box: 20 km Level 0 Search Space

Bounding Box: 20 km Level 1

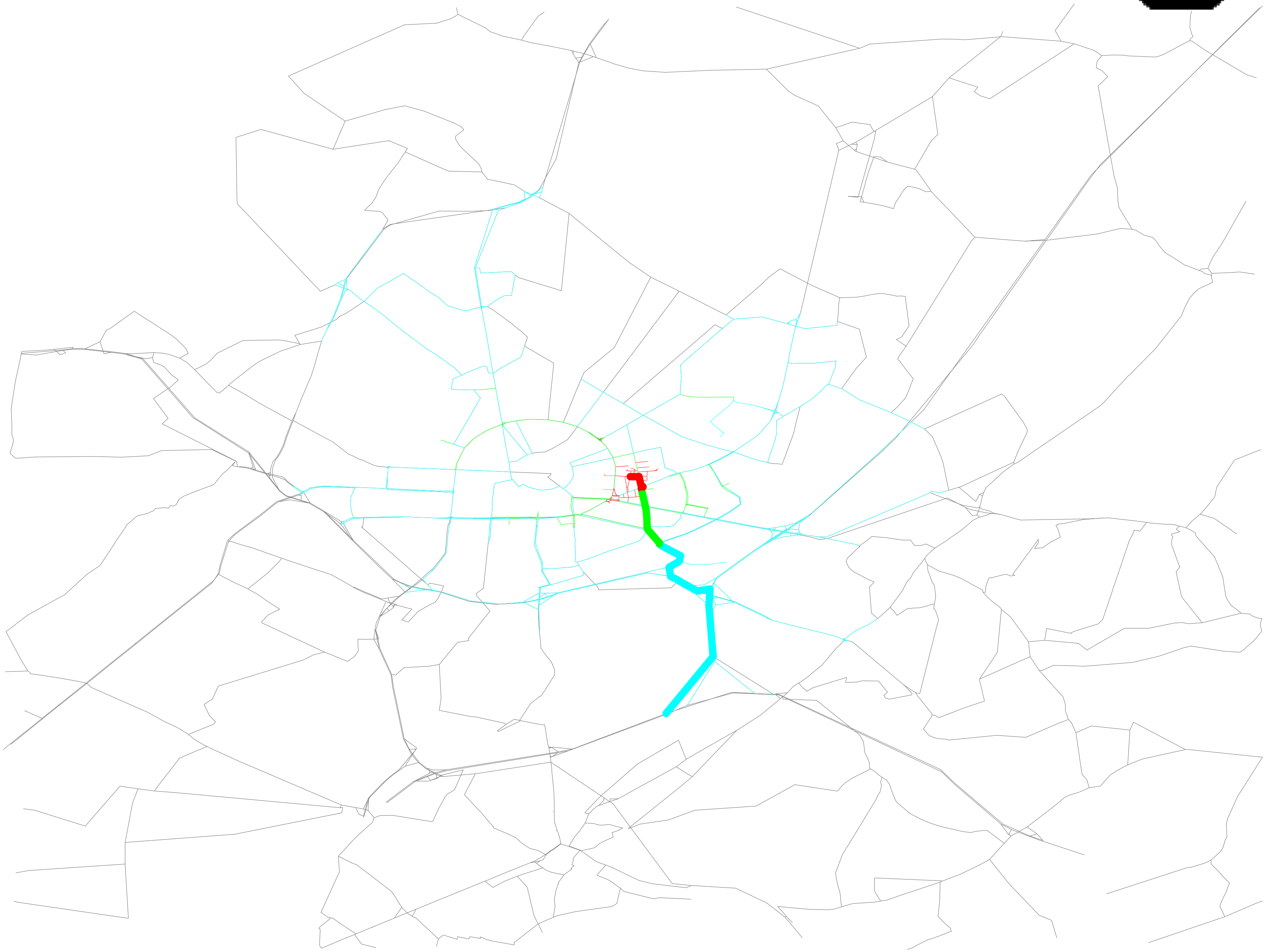
Bounding Box: 20 km Level 1 Search Space



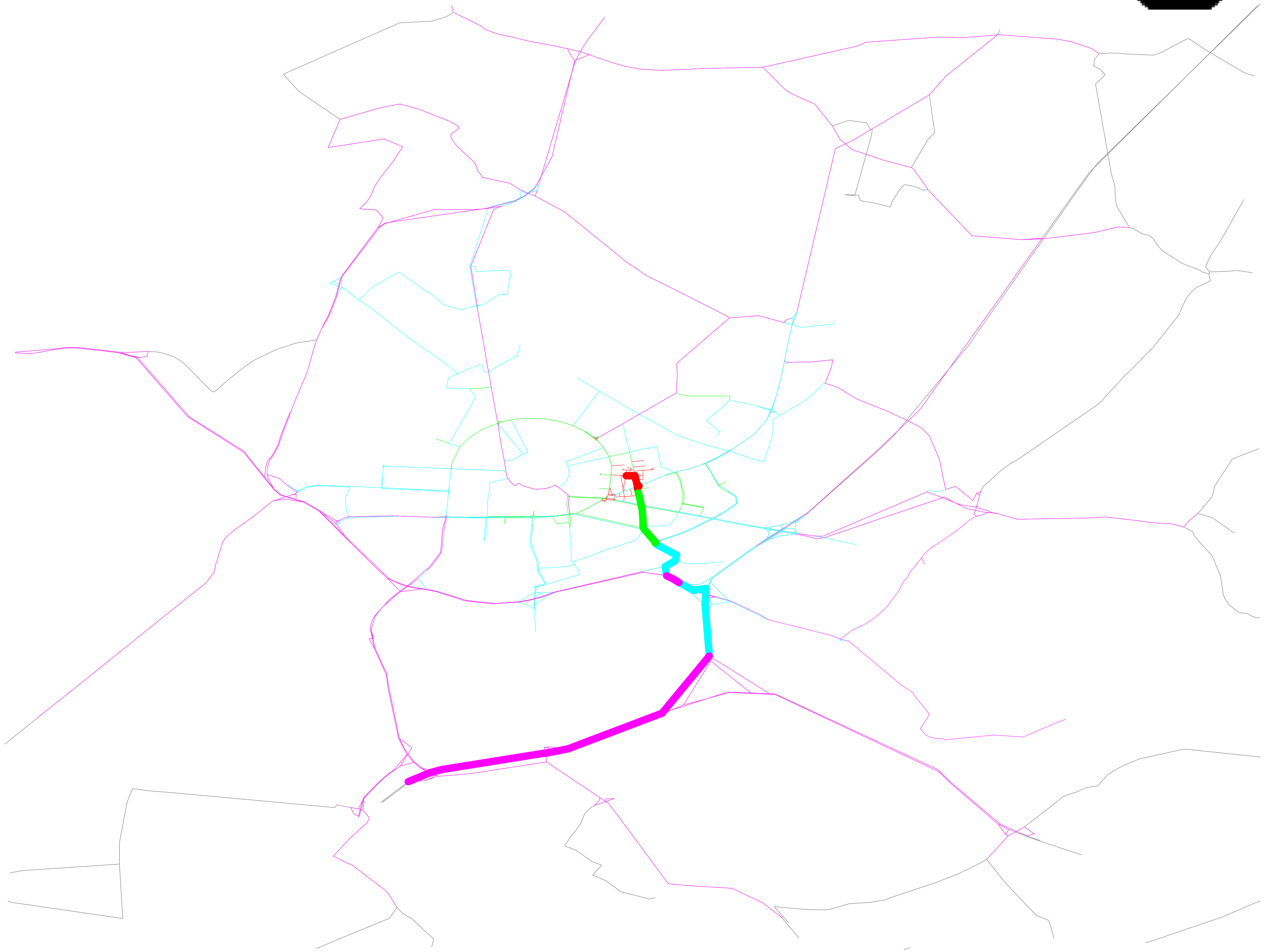
Bounding Box: 20 km Level 2



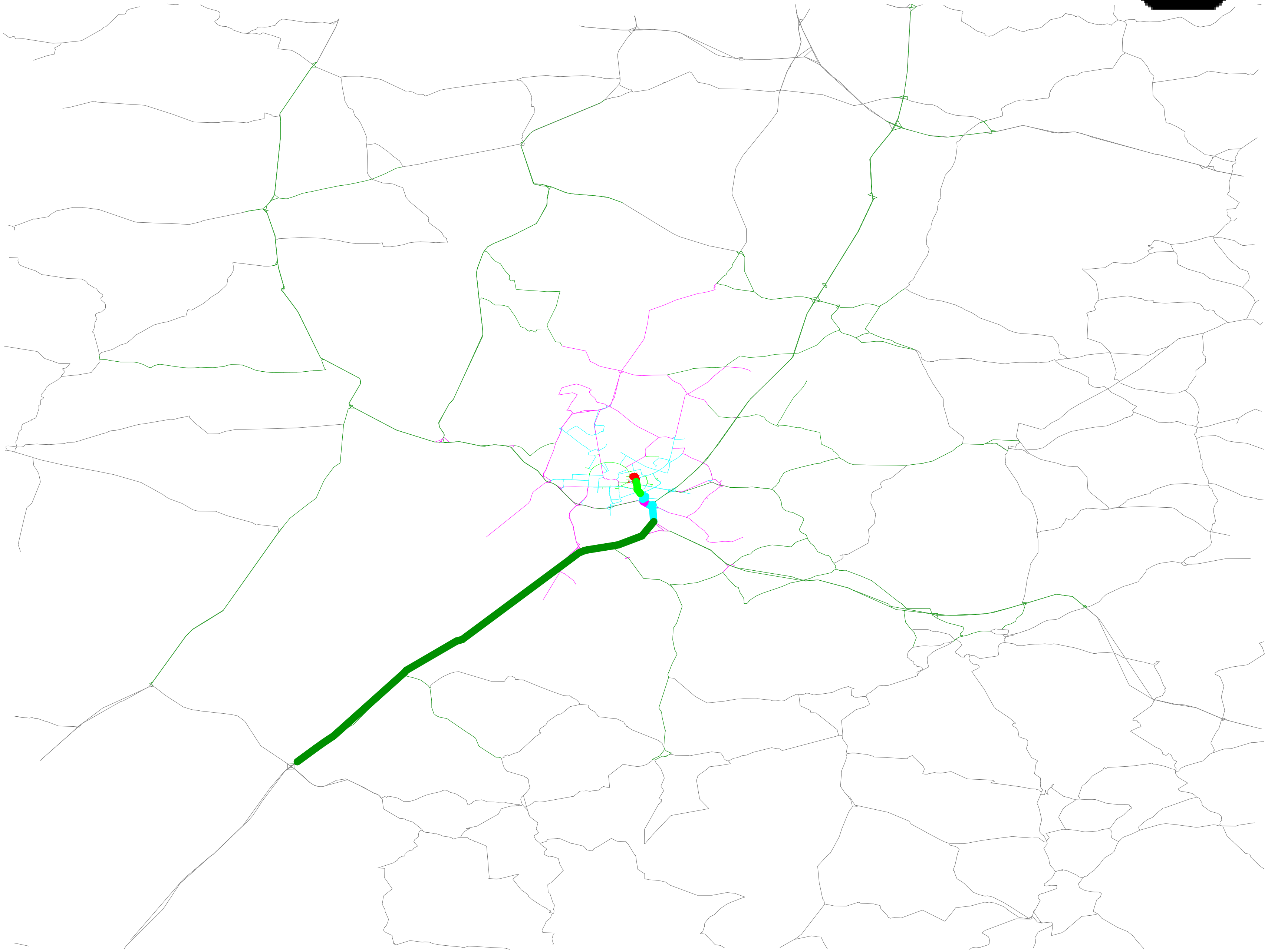
Bounding Box: 20 km Level 2 Search Space



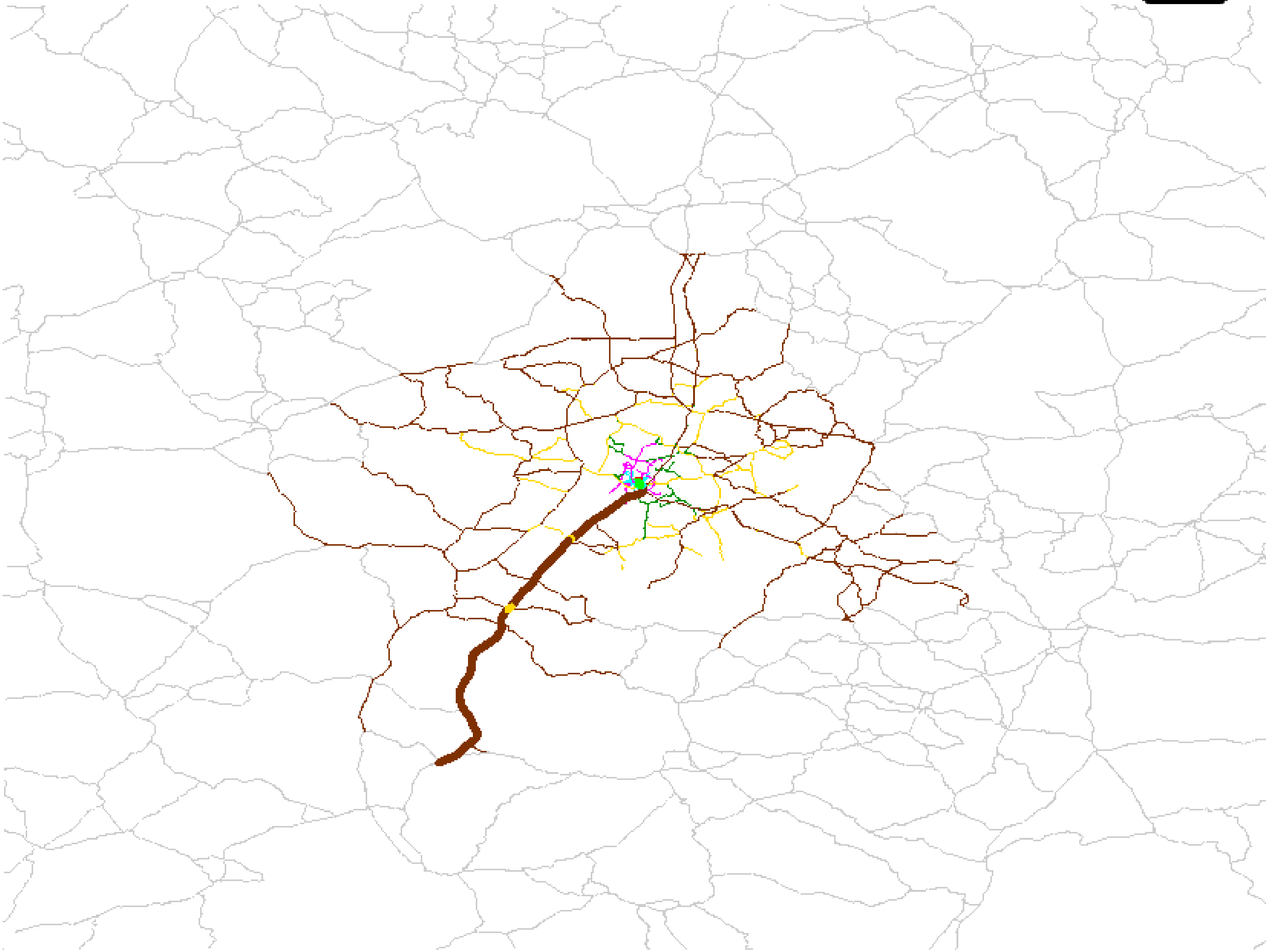
Bounding Box: 20 km Level 3 Search Space



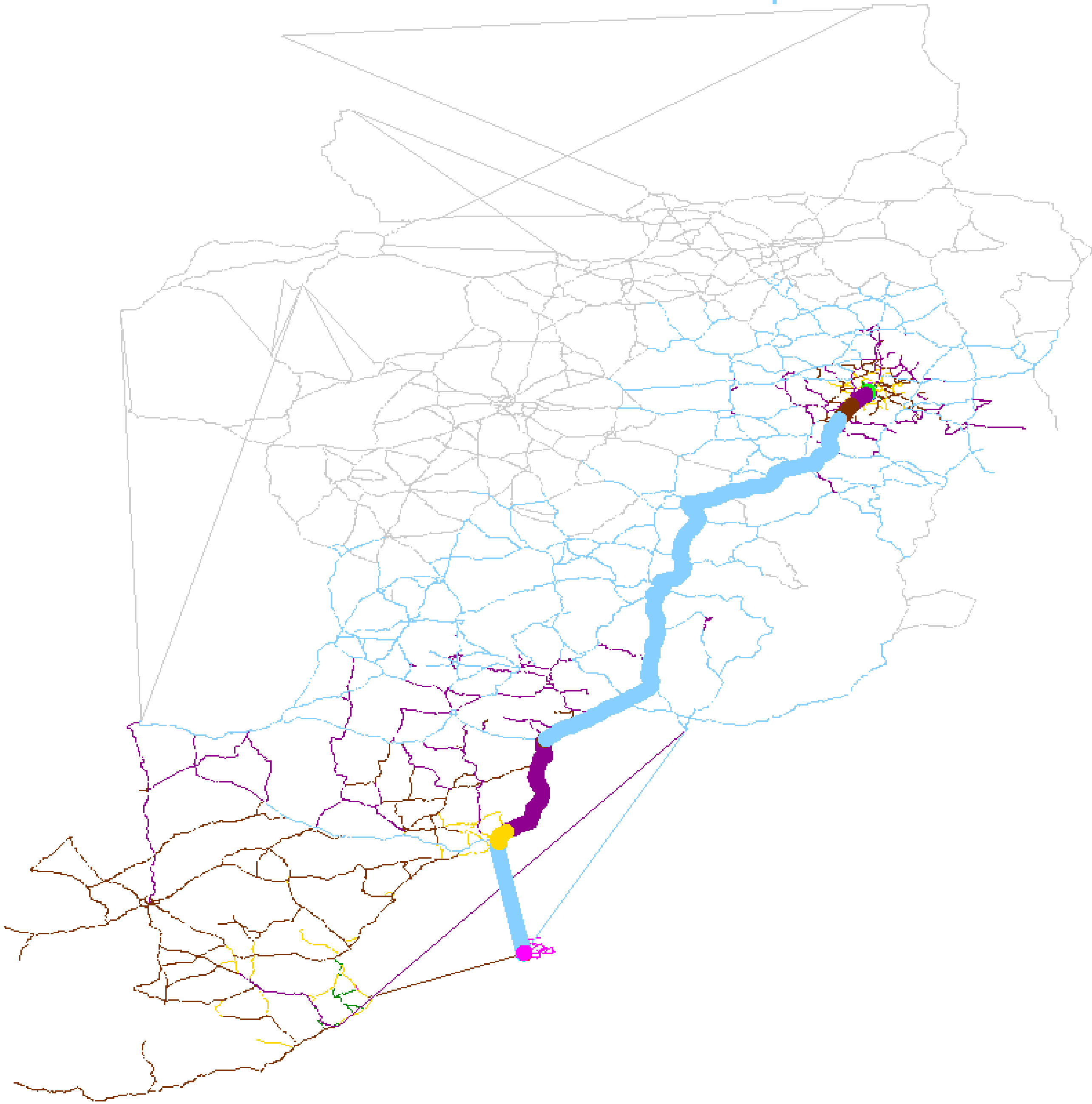
Bounding Box: 80 km Level 4 Search Space



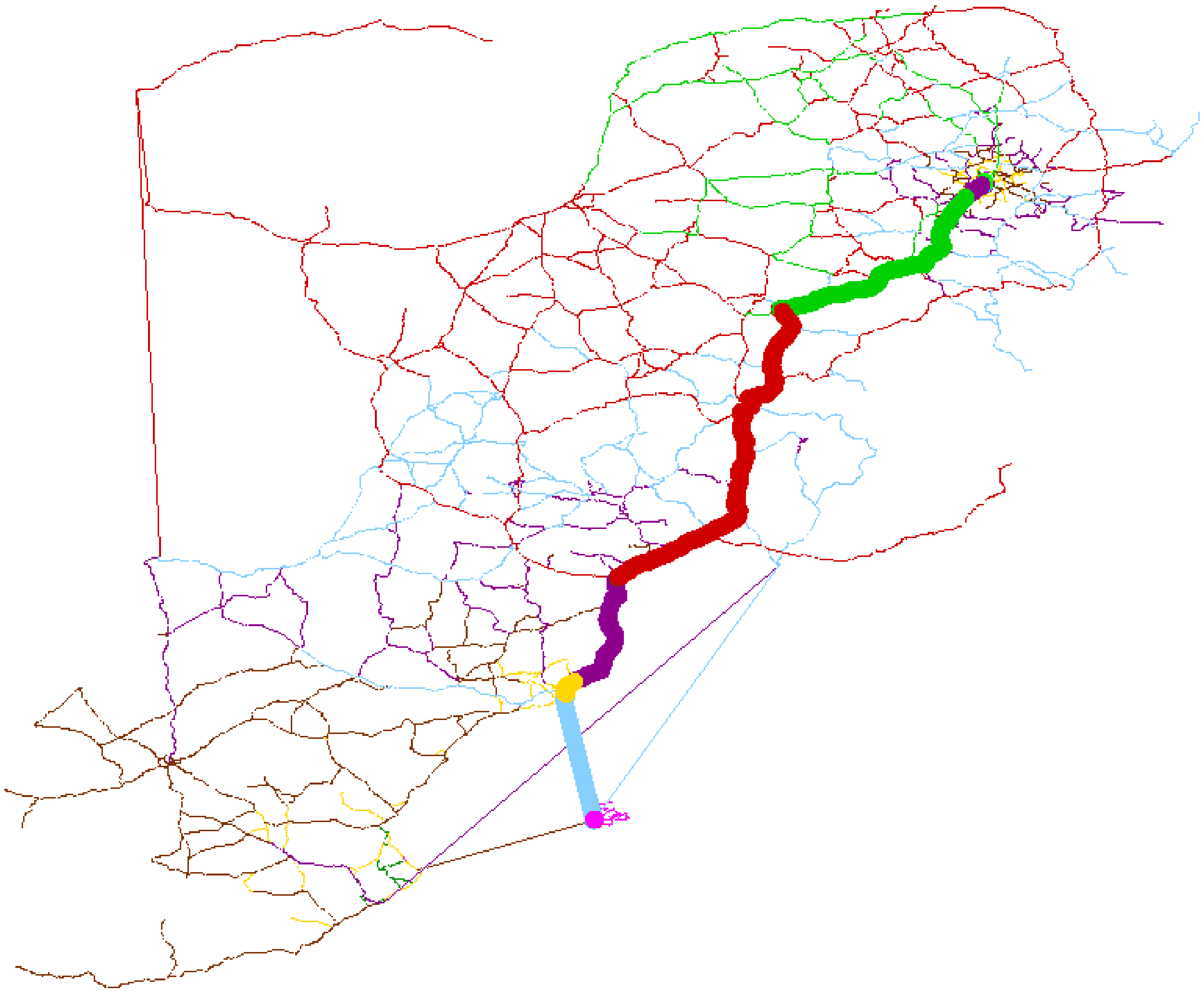
Bounding Box: 400 km Level 6 Search Space



Level 8 Search Space

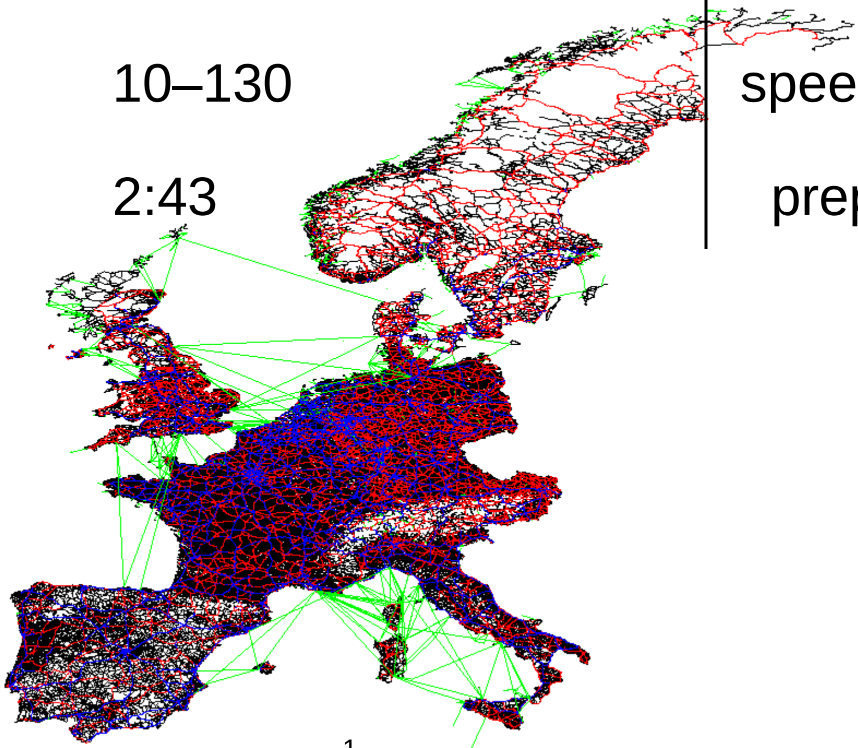


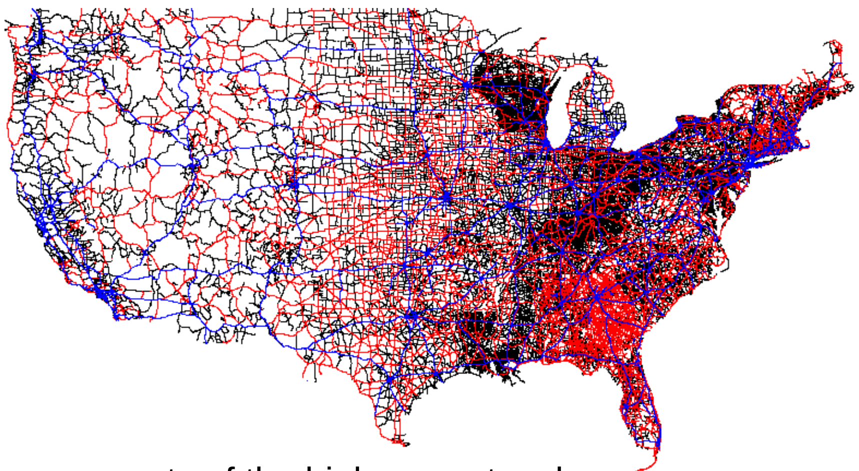
Level 10 Search Space



Experiments

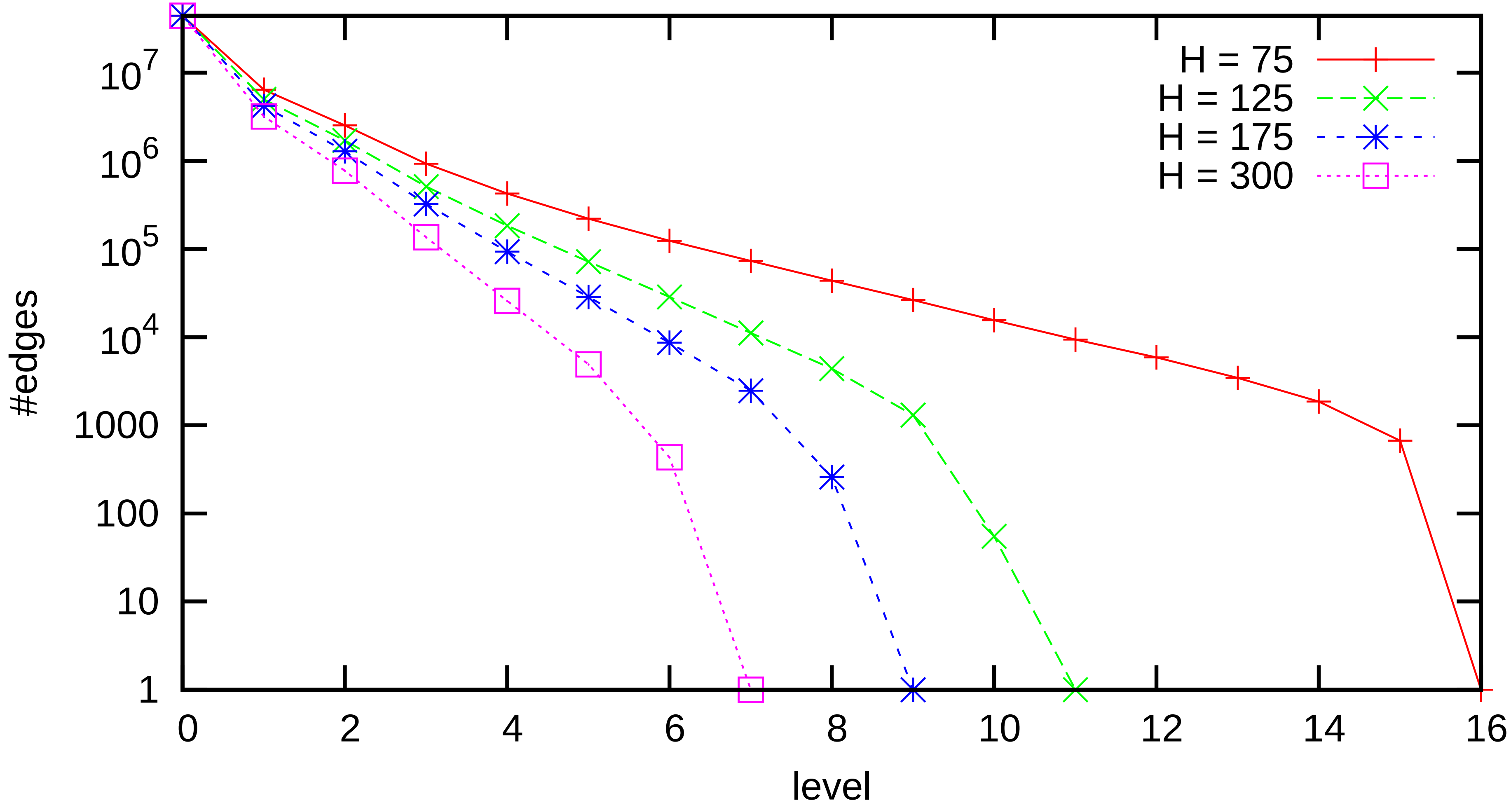
W. Europe (PTV)	**Our Inputs**	USA (Tiger Line)
18 029 721	#nodes	24 278 285
22 217 686	#edges	29 106 596
13	#road categories	4
10–130	speed range [km/h]	40–100
2:43	preproc time[1] [h]	4:20



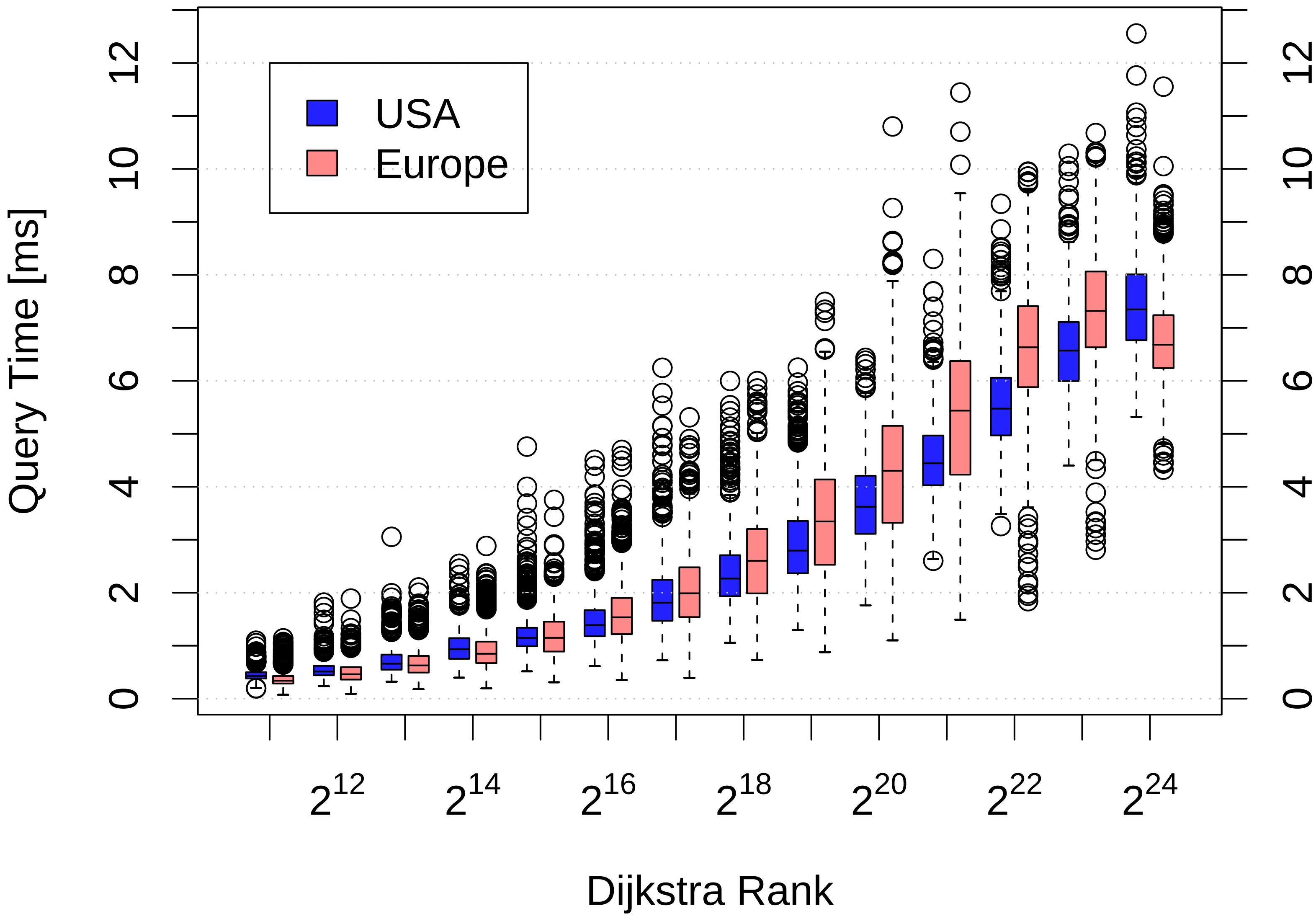


[1] using a faster method that computes supersets of the highway networks

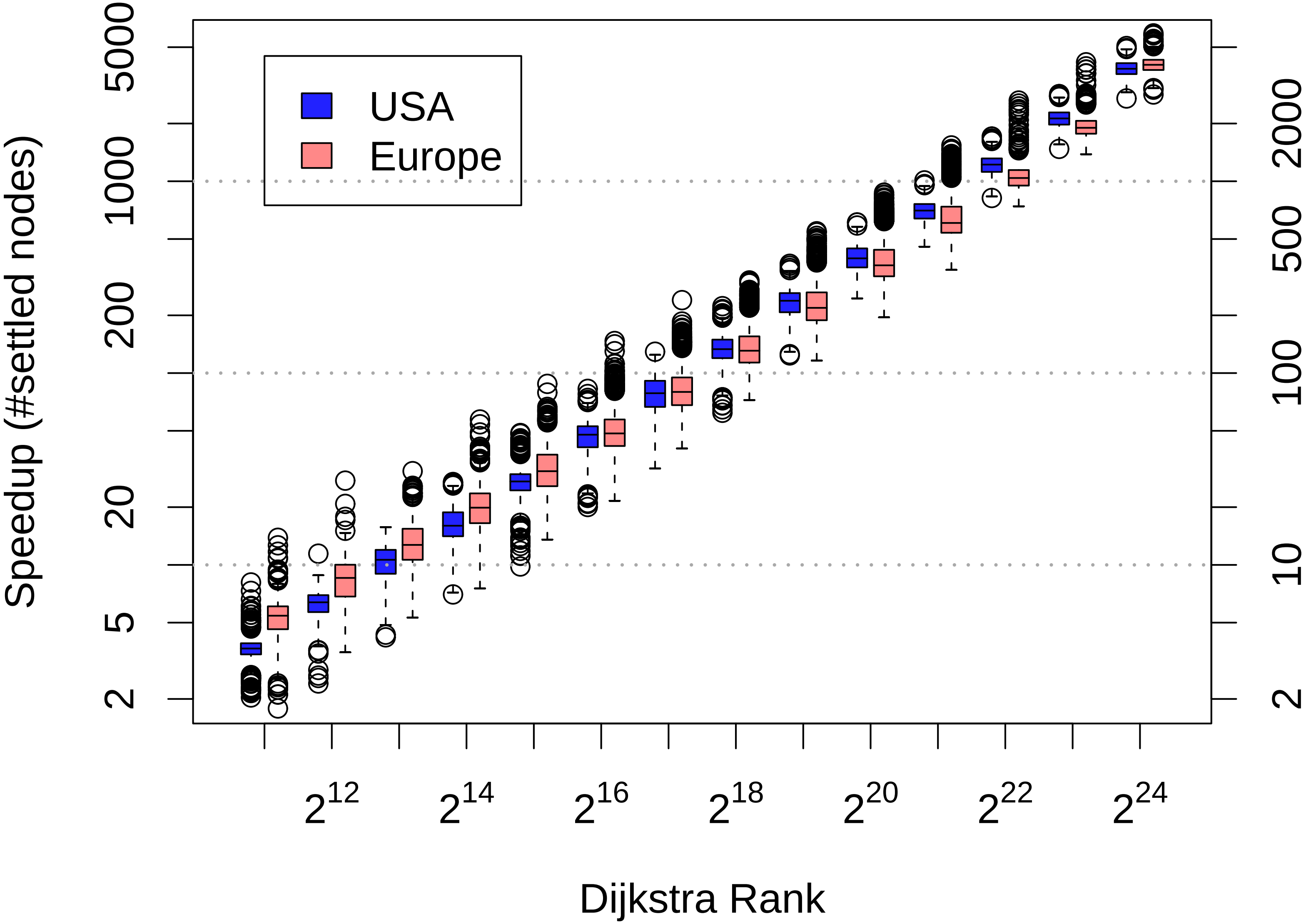
Shrinking of the Highway Networks — Europe



Queries – Time



Queries – Speedup



Conclusion

- exact shortest (i.e. fastest) paths in large road networks

  highway network e.g. Europe ≈ 18 000 000 nodes

  preserves shortest paths

- fast queries

  < 8 ms on average

- fast preprocessing

  3 hours

- reasonable space consumption

  1.8 GB, improvable

- scale-invariant, i.e., optimised not only for long paths

  multilevel approach

Future Work

- speed up preprocessing (e.g. parallelisation)
- speed up queries (e.g. combination with goal directed approaches like landmarks, geometric containers, or bit vectors)
- fast local updates of the highway network (e.g. due to traffic jams)
- external memory
- ...

Summary of the Results

		USA	Europe	Germany
input	#nodes	24 278 285	18 029 721	4 345 567
	#edges	29 106 596	22 217 686	5 446 916
	#degree 2 nodes	7 316 573	2 375 778	604 540
	#road categories	4	13	13
params	average speeds [km/h]	40–100	10–130	10–130
	H	225	125	100
constr.	CPU time [h]	4.3	2.7	0.5
	#levels	7	11	11
query	avg. CPU time [ms]	7.04	7.38	5.30
	#settled nodes	3 912	4 065	3 286
	speedup (CPU time)	2 654	2 645	680
	speedup (#settled nodes)	3 033	2 187	658
	efficiency	112%	34%	13%
	main memory usage [MB]	2 443	1 850	466 (346)