GraphHopper

Flexible Routing with GraphHopper

And how it can be ~~mis~~used for data analysis

Peter Karich SOTM **2019**
Co-Founder GraphHopper

GraphHopper

Who am I?

GraphHopper company
What is our mission?

Peter Karich

Stefan Schröder

Dr.-Ing. Michael Zilske

Maximilian Sturm

Robin Boldt

Dr. Michal Maciejewski

Dr. Andreas Barth

graphhopper maps

Demo Link

Heidelberg Hauptbahnhof, 6911!

Universität Heidelberg, 69120, G

OpenStreetMap | Edit | History | Export

Heidelberg Hauptbahnhof, Max-Planck-Ring,

Universität Heidelberg, Im Neuenheimer Feld

Foot (GraphHopper) | Go

Reverse Directions

Directions

Distance: 2.3km. Time: 0:27.
Ascend: 25m. Descend: 22m.

1. Continue
2. Turn right 160m
3. Keep right 40m
4. Turn right 30m

113.39 m service

Distance: **1.6 km**
Elevation: **113.39 m**
Segment length: **0.1 km**
Type: **service**

150 m
140 m
130 m
120 m
110 m
100 m
90 m

0.00 km 0.50 km 1.00 km 1.50 km 2.00 km

Legend

Donate Imprint Terms Privacy

300 m

GraphHopper Routing Engine

- Open Source under Apache License 2.0
- Java library and web service for routing

  No maps, no geocoding
- It is fast and memory efficient
- Works with OpenStreetMap data, GTFS and others
- Algorithms: Dijkstra, A*, Landmarks, CH
- Out of the box: for walking, car, bike, public transit, …

Features

Here is a list of the more detailed features including a link to the documentation:

- Quick installation and start for users - just Java necessary! Simple start for developers due to Maven.
- Works out of the box with OpenStreetMap (osm/xml and pbf) and can be adapted to custom data
- OpenStreetMap integration: Takes care of the road type, speed limit, the surface, barriers, access restrictions, ferries, conditional access restrictions, ...
- GraphHopper is fast. And with the so called "Contraction Hierarchies" it can be even faster (enabled by default).
- Memory efficient data structures, algorithms and the low and high level API is tuned towards ease of use and efficiency
- Provides a simple web API including JavaScript and Java clients
- Multiple weightings (fastest/shortest/...) and pre-built routing profiles: car, bike, racingbike, mountain bike, foot, motorcycle, ...
- Supports public transit routing and GTFS.
- Offers turn instructions in more than 35 languages, contribute or improve here
- Displays and takes into account elevation data (per default disabled)
- Can apply real time changes to edge weights (flexible and hybrid mode only)
- Customization of vehicle profiles per request are possible (flexible and hybrid mode only)
- Possibility to specify a heading parameter of the vehicle for start, end and via points for navigation applications via `pass_through` or `heading` parameters (flexible and hybrid mode only)
- ative routes (flexible and hybrid mode only)
- costs and restrictions
- Country specific routing via SpatialRules
- The core uses only a few dependencies (hppc, jts and slf4j)
- Scales from small indoor-sized to world-wide-sized graphs
- Finds nearest point on street e.g. to get elevation or 'snap to road' or being used as spatial index (see #1485)
- Does map matching with GraphHopper
- Calculates isochrones with GraphHopper
- Shows details along a route ("path details") #1142
- Shows the whole road network in the browser for debugging purposes ("vector tile support") #1572

graphhopper

Selected New Features

Vector Tiles Endpoint

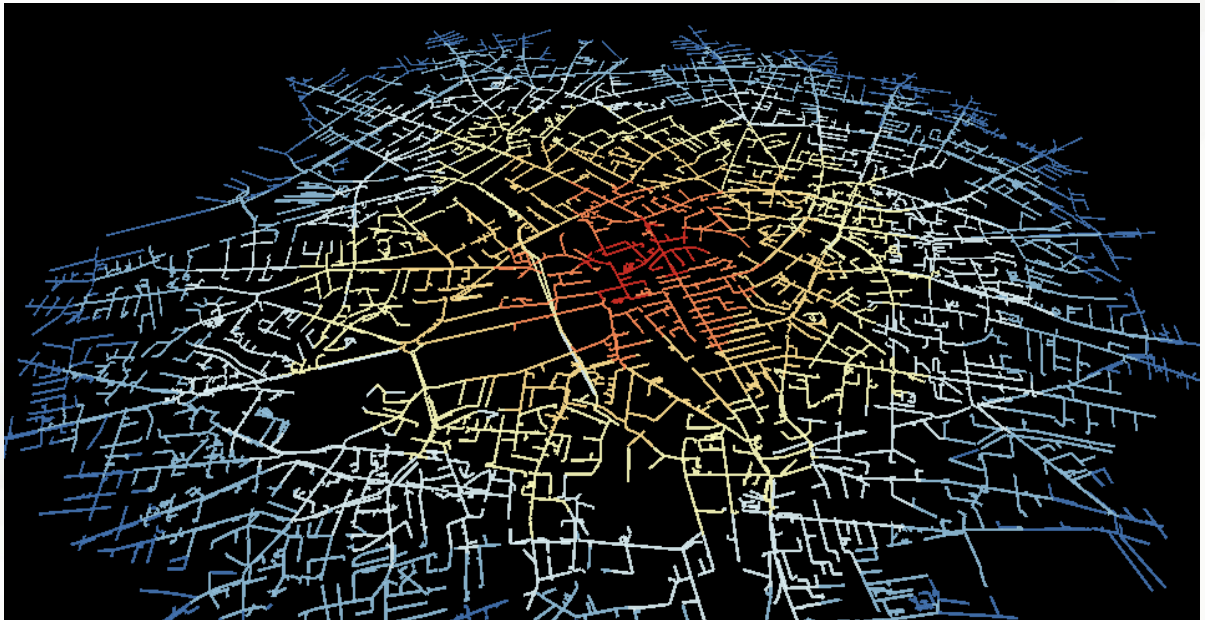
#1572

Shortest Path Tree Endpoint

#1577

- CSV with lat,lon and previous coordinate
- Feedback from community

- Example in R lang:

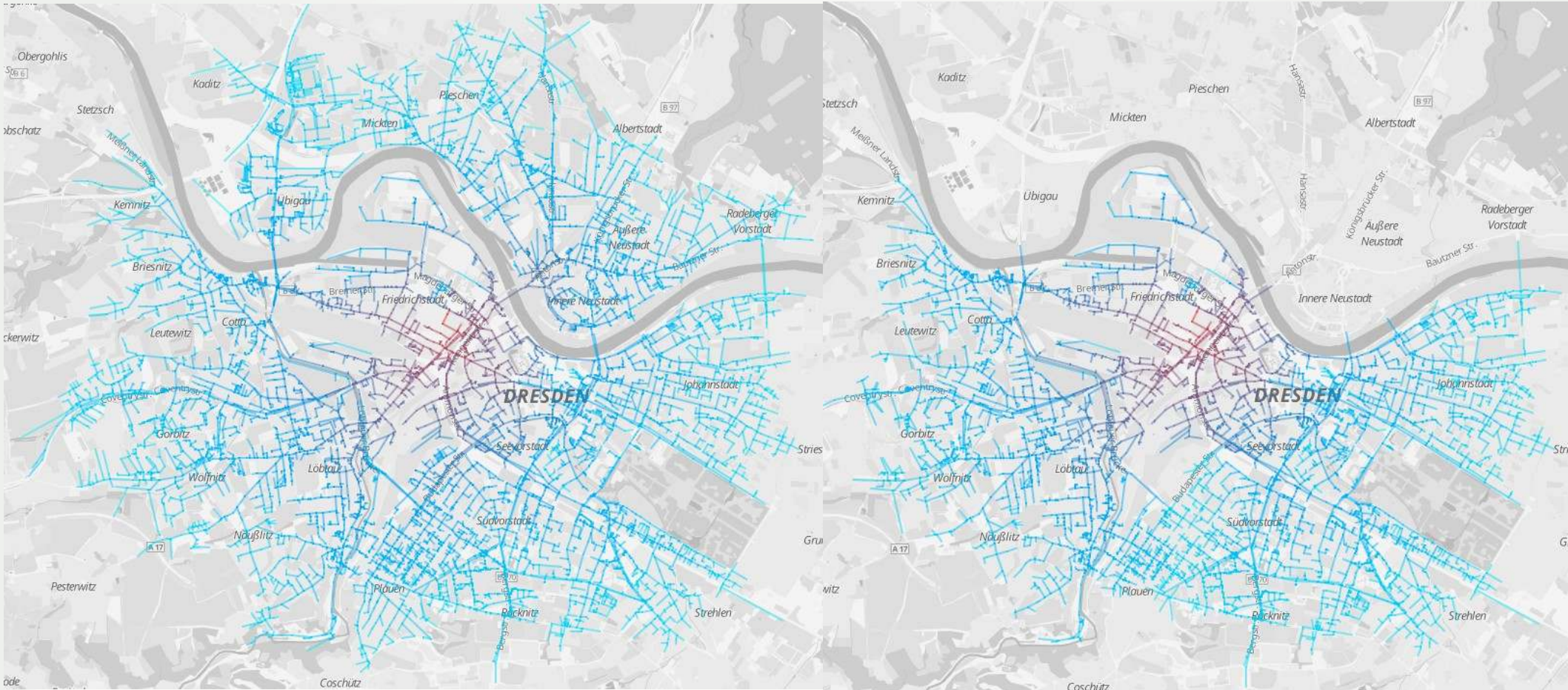


Use GraphHopper For Data Analysis

1. Impact of bridge construction on road network. High precise “Isochrones”: draw shortest path tree directly in browser. Simulate “what if” scenarios
2. Level of Traffic Stress & Highlight curly roads
3. Speed limit debate regarding safety
4. Plan location of new fire station
5. Find closest restaurants by driving time

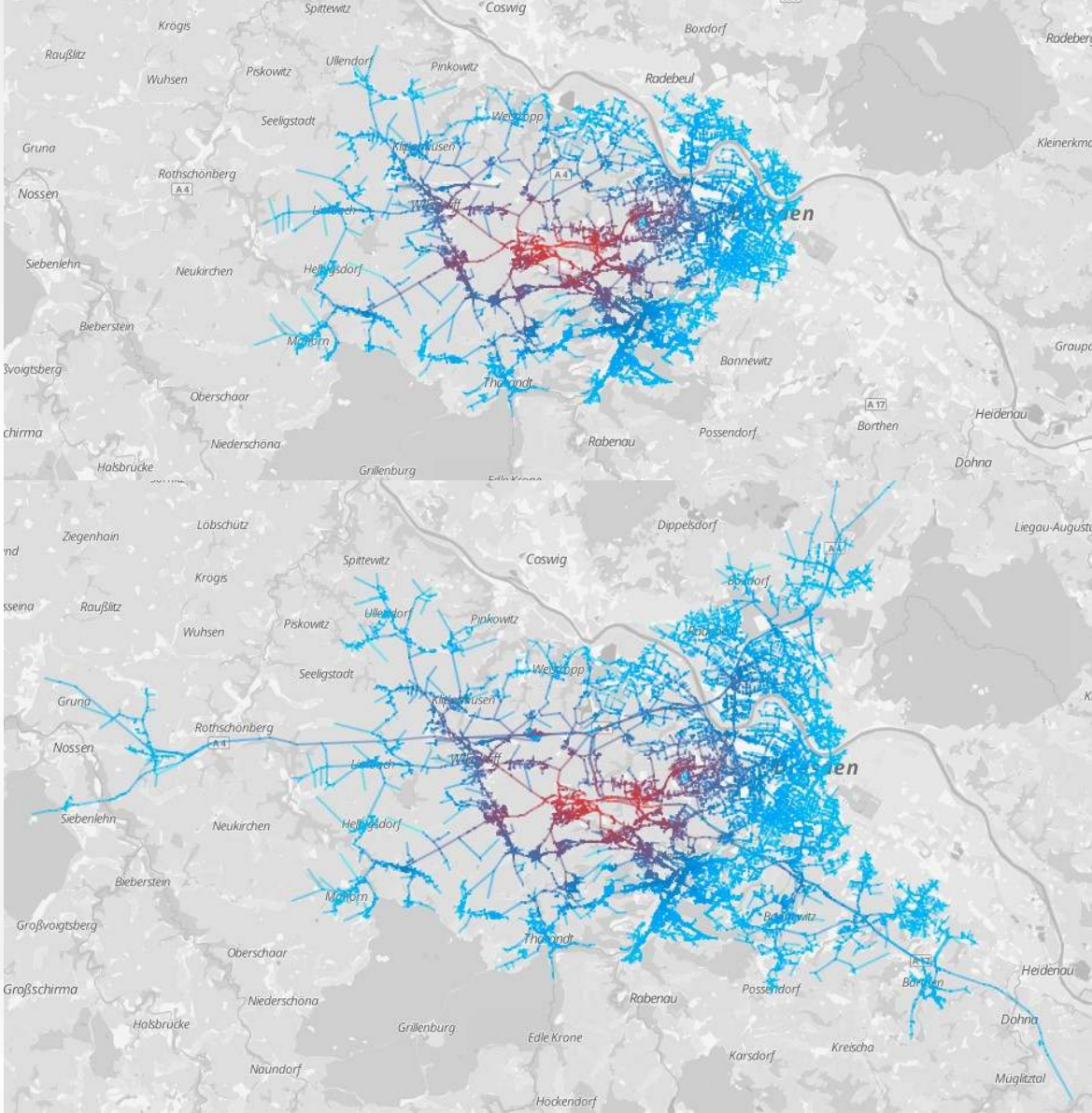
   & Find closest restaurants from a route

1. Impact of bridge construction

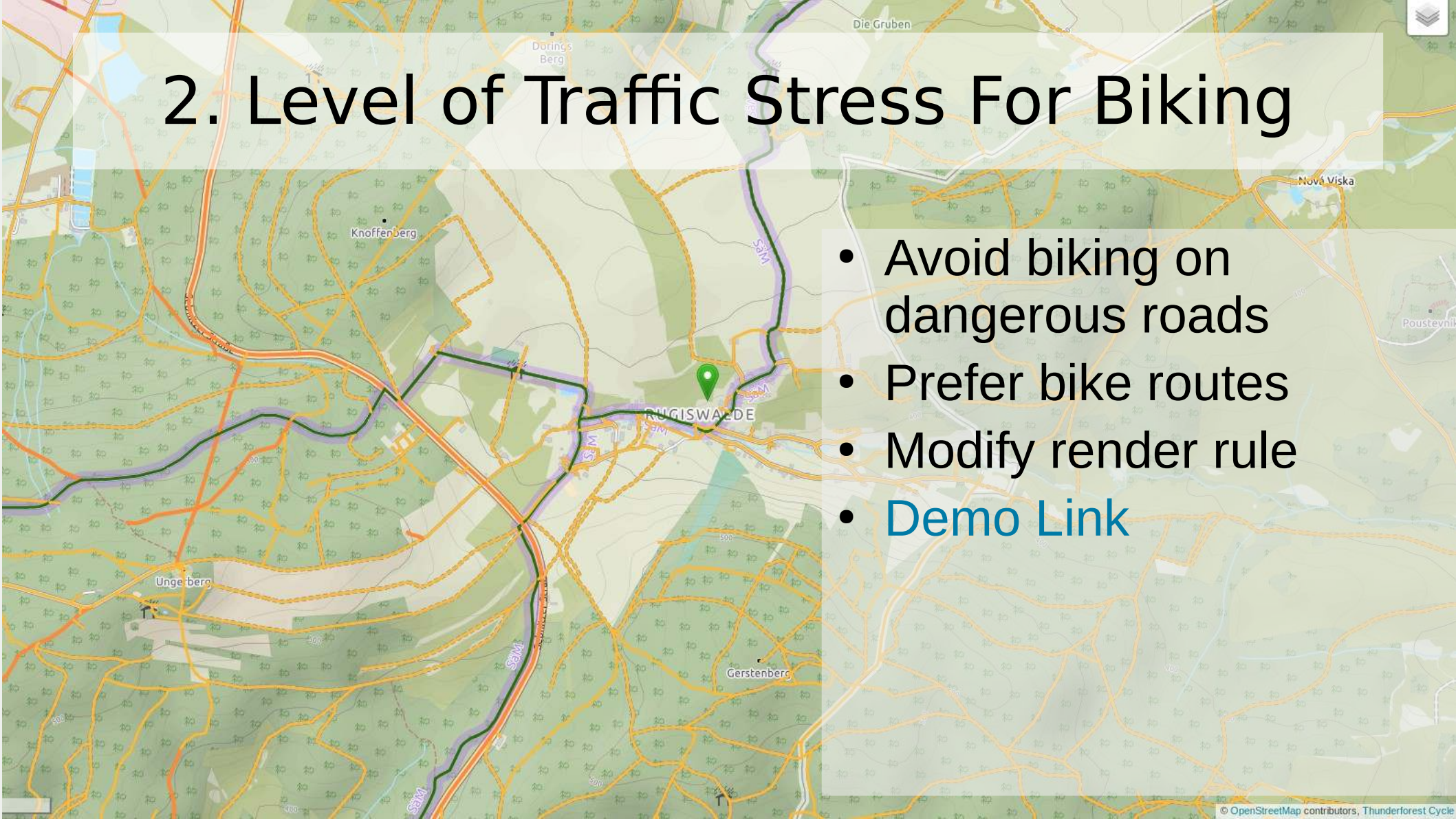


Avoid highway=motorway

Uses /spt endpoint



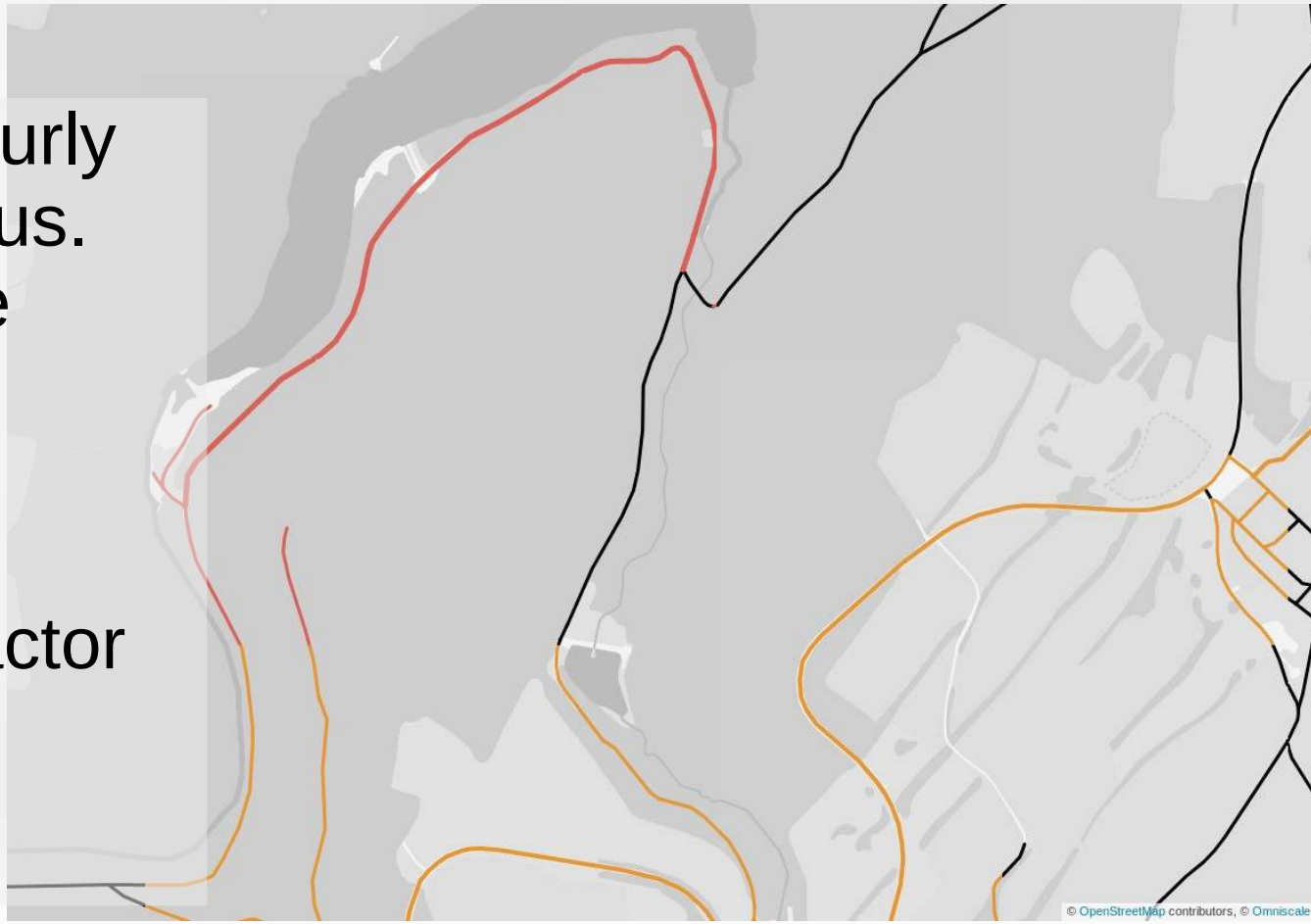
2. Level of Traffic Stress For Biking



- Avoid biking on dangerous roads
- Prefer bike routes
- Modify render rule
- Demo Link

2. Highlight Curvy Roads

- For some people curly roads are dangerous. For others they are attractive
- Fetch vector tiles from /mvt endpoint and return curvy factor e.g. <0.6 → red
- Modify render rule



3. Speed limit debate regarding safety

- German crash data 2016 and 2017 from “destatis”
- OpenStreetMap speed limit data
- Use new storage feature for highway tag, maxspeed and crash counter

3. Speed limit debate regarding safety

Results:

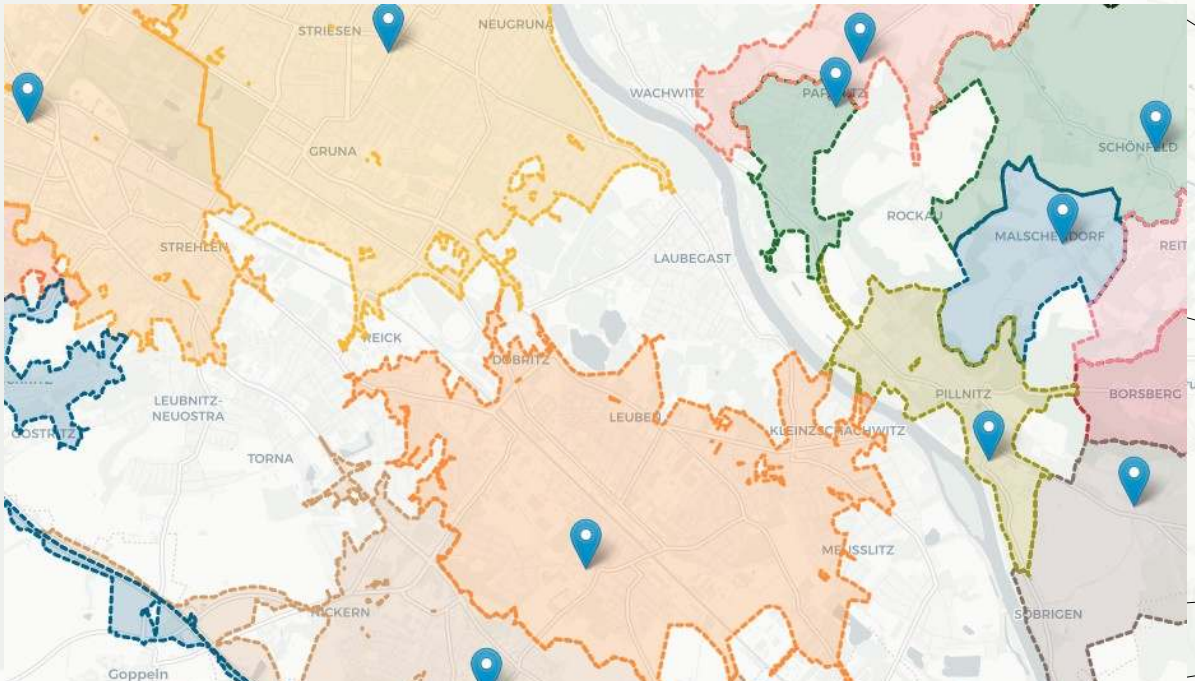
- 13500 km highways in Germany
- ~65% highways without speed limit (official source is similar)
- 69.5% of deaths on segments w/o speed limit
- Traffic density required →
- Signs could safe >100 lifes/a

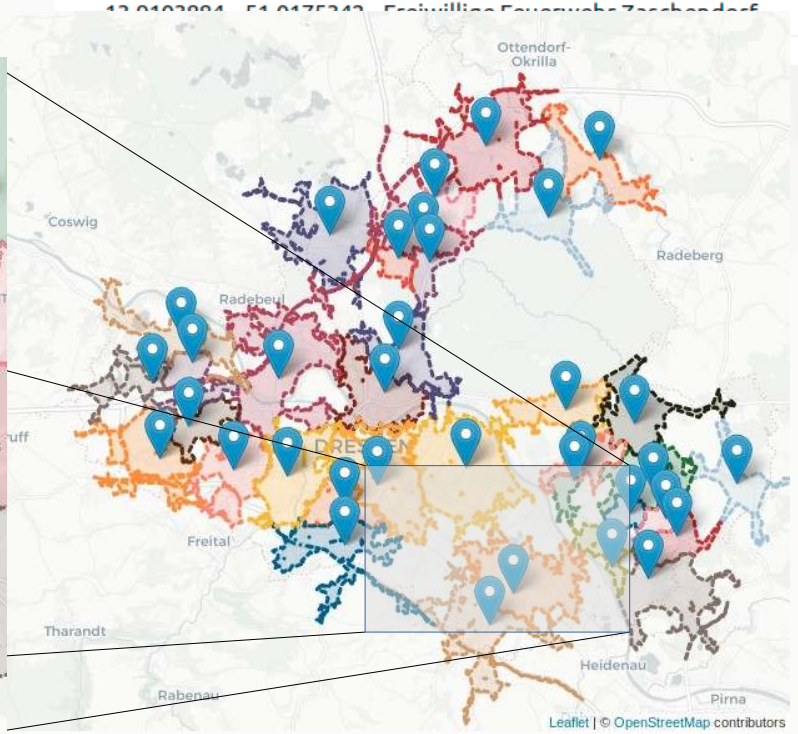
4. Plan Location of new Fire Station

How to find gaps in reachability?

→ Multi-source isochrone

@LON	@LAT	NAME
13.7741223	51.1119795	Freiwillige Feuerwehr Klotzsche
13.8492082	51.0612081	Stadtteilfeuerwehr Bühlau
13.7271078	51.0281545	Betriebliche Feuerwehr der Technischen
13.9428954	51.0355865	Feuerwehr Eschdorf



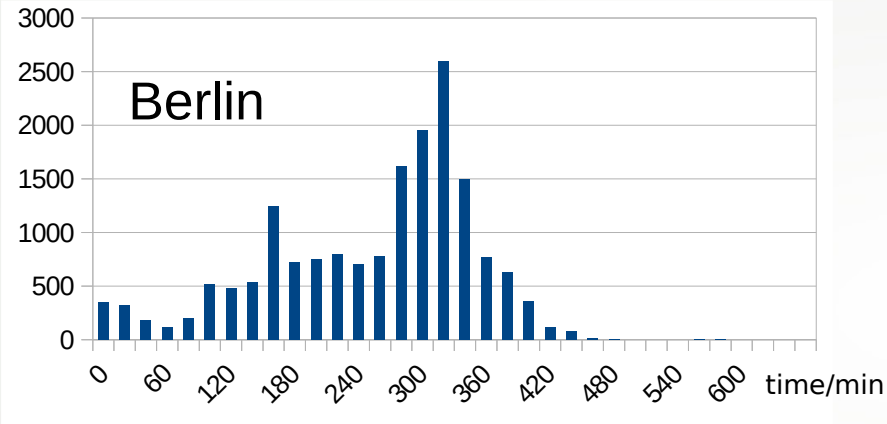


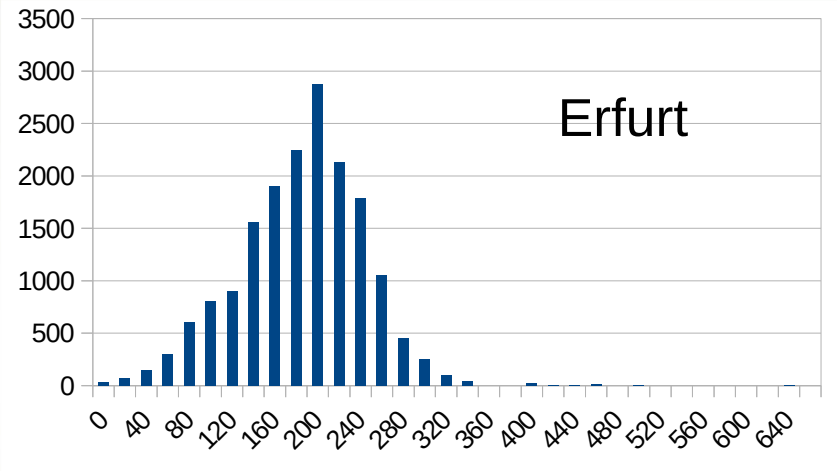
5. Find Closest Restaurants

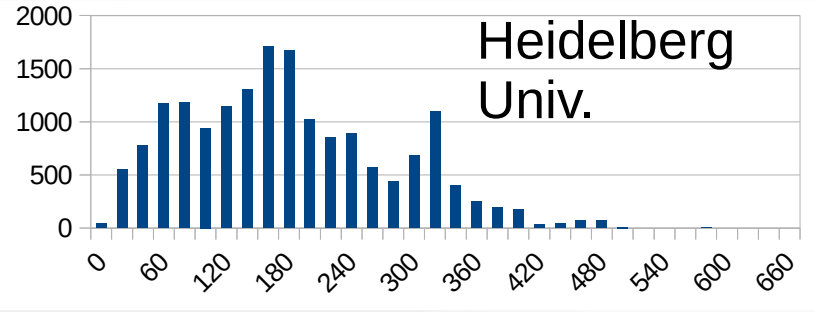
- Get ~18K restaurants:
  bzgrep -B 1 restaurant germany.osm.bz2 | grep node
- Store restaurant count per edge
  → 5s on my old laptop
- Start in “Erfurt” city and explore Germany
  9.3M nodes & 11.8M edges
- Return the list of “driving-time-sorted” restaurants
  → <30s

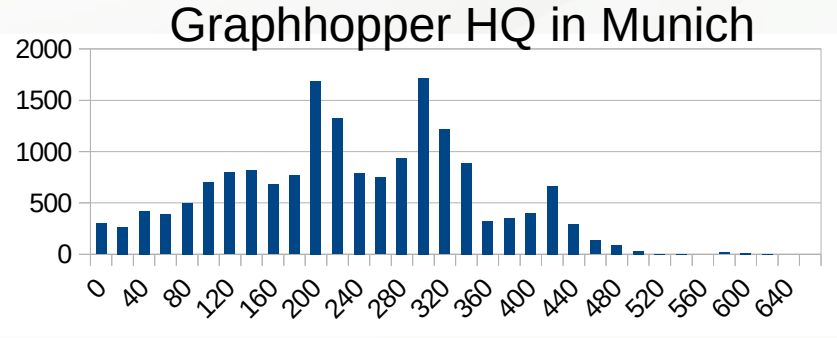
```
1 restaurants, 452min, 534.0km
1 restaurants, 452min, 535.0km
2 restaurants, 453min, 535.0km
1 restaurants, 453min, 535.0km
1 restaurants, 453min, 535.0km
1 restaurants, 453min, 535.0km
1 restaurants, 454min, 535.0km
1 restaurants, 455min, 535.0km
1 restaurants, 458min, 539.0km
1 restaurants, 493min, 614.0km
1 restaurants, 493min, 614.0km
1 restaurants, 493min, 615.0km
1 restaurants, 494min, 615.0km
1 restaurants, 634min, 627.0km
1 restaurants, 643min, 634.0km
```

5. Histograms with Restaurants for Fun



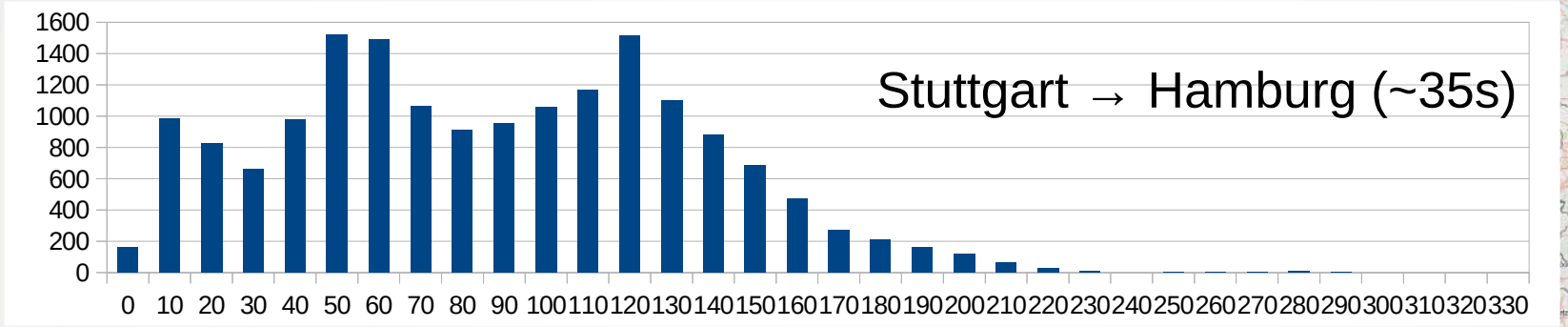


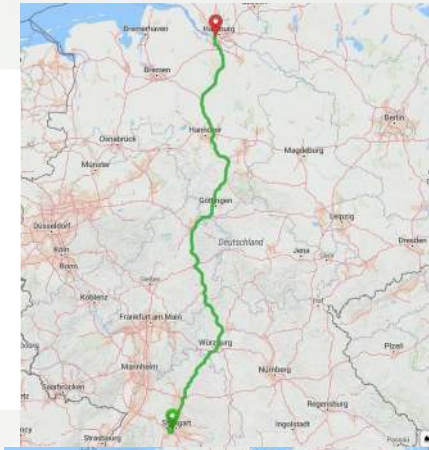


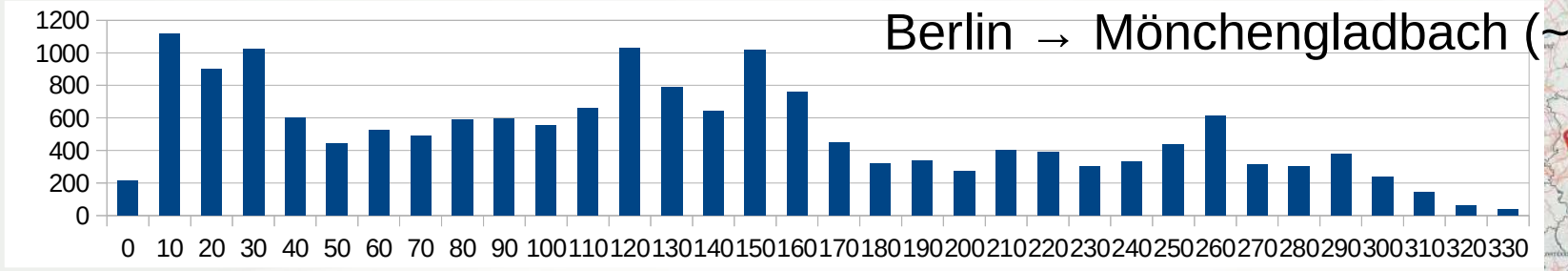


5. Find Closest Restaurants along a route

Or same algorithm, different problem: Find shortest path from location to river







GraphHopper for Data Analysis

advantages

- Fast
- Handles massive data well (even on weak computers)
  avoid loading everything into memory via graph.dataaccess=MMAP_STORE
- Perfect for everything that requires road connectivity

disadvantages

- Need to select properties of the source data that go into the graph
  max_speed, distance, avg_speed, max_height, max_width, road_class, surface, road_environment, toll, ...
- Certain use cases still require Java knowledge

GraphHopper Resources

- Different tweaks like curvy roads & find restaurants along a route: https://github.com/graphhopper/graphhopper/tree/sotm_trials
- Crash stats: https://github.com/karussell/crashstats/
- Destatis: https://unfallatlas.statistikportal.de/

Graphhopper

We are looking for contributors!

Contribute Code & Translations
https://github.com/graphhopper/graphhopper/contribute

Forum
https://discuss.graphhopper.com/

Flexible Routing with GraphHopper

peter.karich@graphhopper.com