MASTER THESIS

Optimal Route Planning on Mobile Systems

Adrian Batzill

Structure

- Route Planning
  - Unidirectional, Bidirectional, Hierarchical
- External Memory
- Results

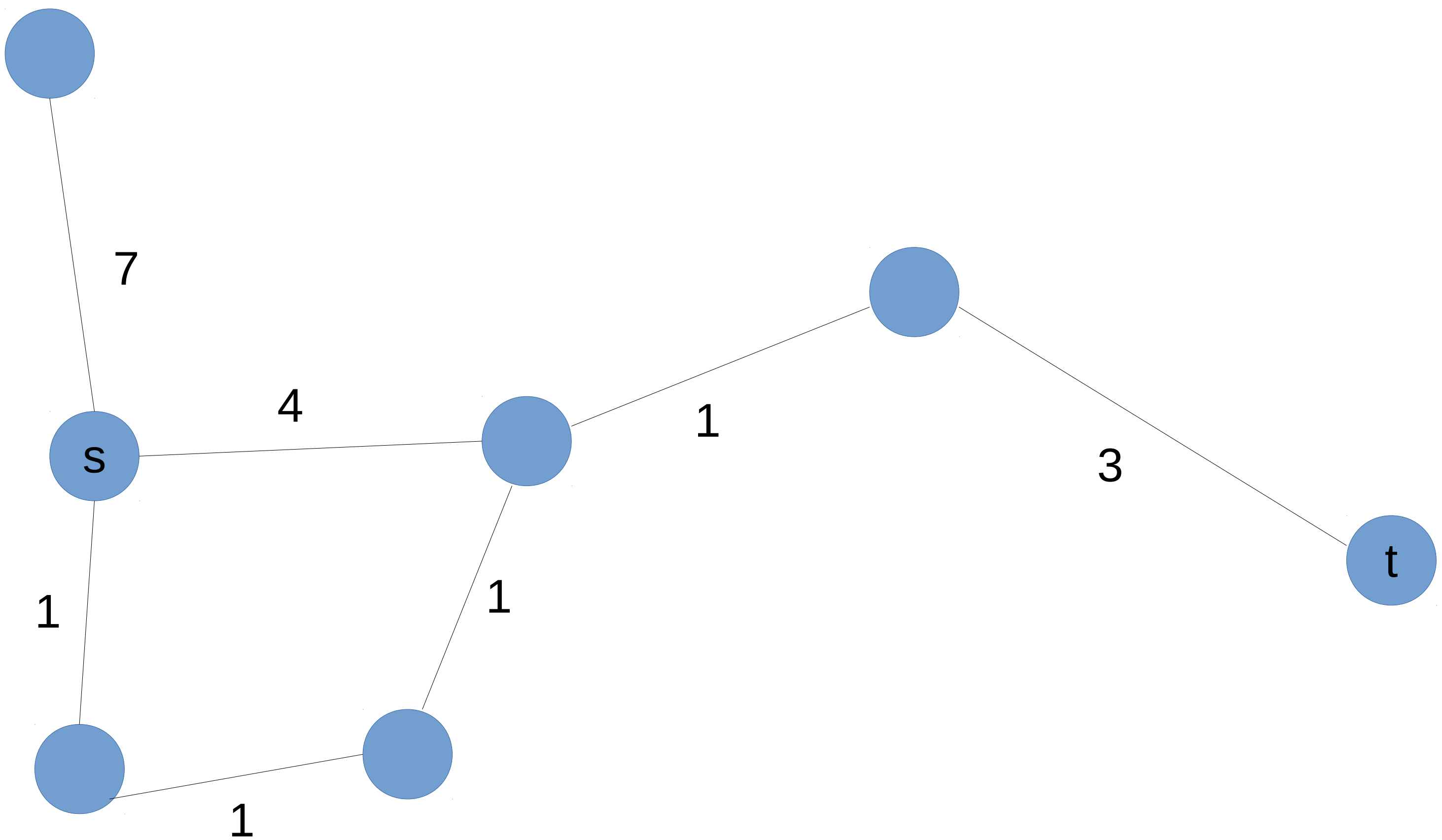
Route Planning

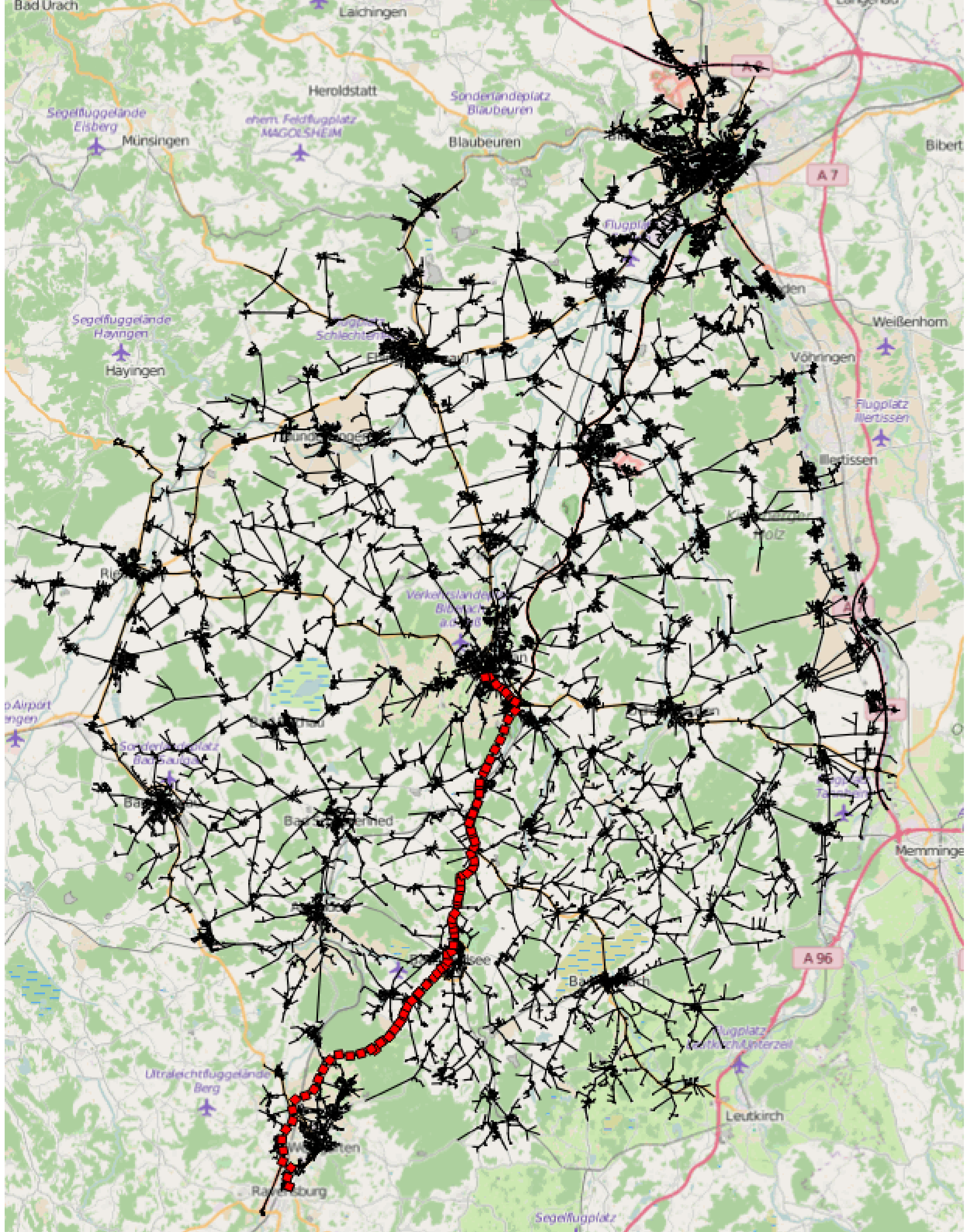
- Road network as weighted, directed graph
- Shortest path algorithms to find *best* route

- *Best?*
  - *Shortest (distance)*
  - *Fastest (traveltime)*
  - *Most beautiful*
  - *Curvy*
  - *Fuel saving*
  - *….*
  - *A mix of multiple criteria*

Dijkstra‘s Algorithm

- Similar to breadth-first search
- Incrementally construct all *shortest paths* from source
- Stop when target is visited
- Priority queue with *tentative distance*


7
4
s
1
3
t
1
1
1


Bad Urach
Laichingen
Heroldstatt
Sonderlandeplatz
Blaubeuren
Segelfluggelände
Eisberg
Münsingen
ehem. Feldflugplatz
MAGOLSHEIM
Blaubeuren
A 8
A 7
Weißenhorn
Vöhringen
Segelfluggelände
Hayingen
Hayingen
Flugplatz
Illertissen
Illertissen
A 96
Flugplatz
Leutkirch/Unterzeil
Leutkirch
Ultraleichtfluggelände
Berg
Segelflugplatz

A* Algorithm

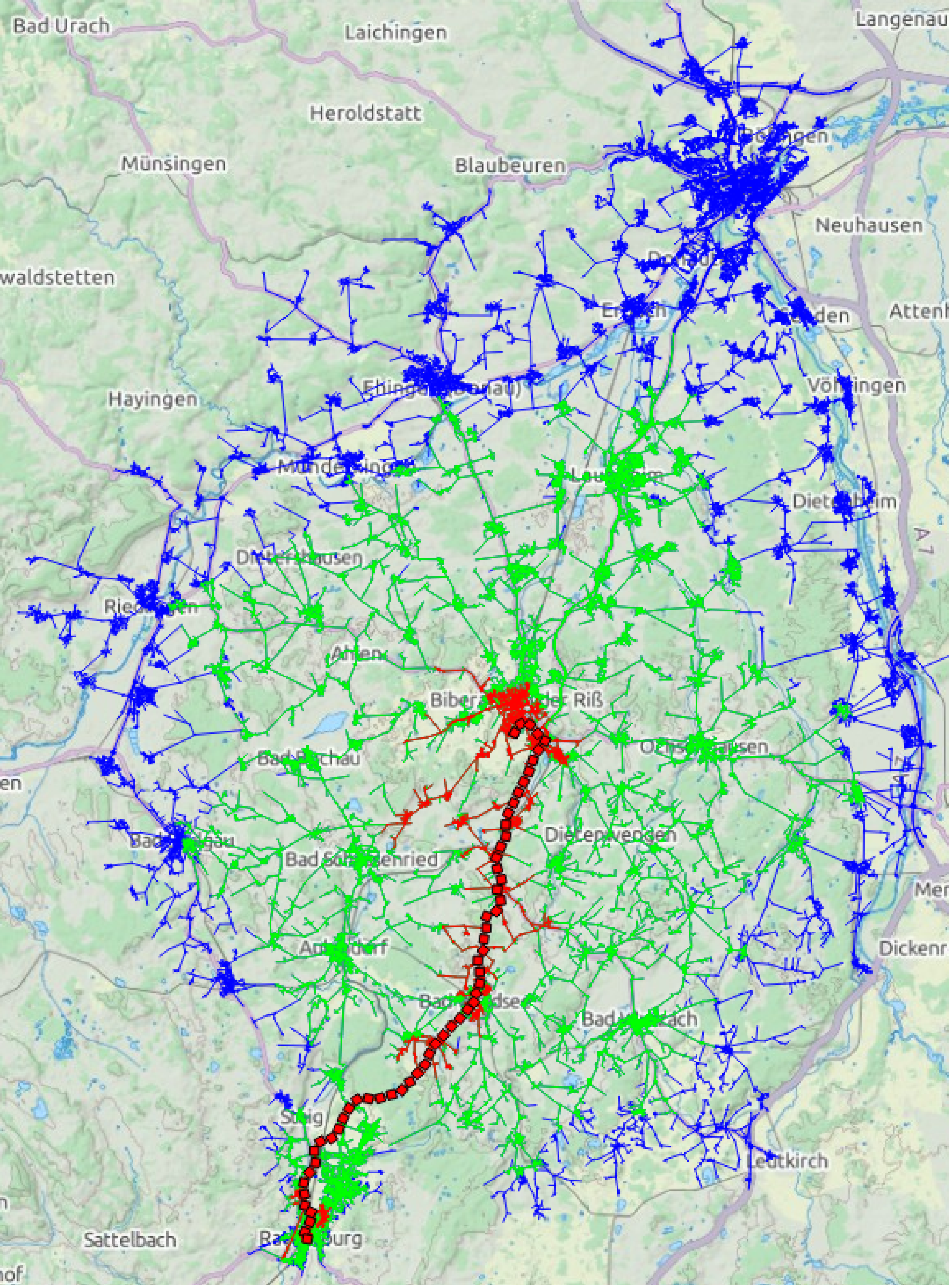
- Very similar to Dijkstra
- Goal-directed with heuristic
- Optimal for monotonic (underestimating) heuristics
- Simple: Use straight line
  - At least for a *distance*-weighting
  - For *traveltime*-weighting: $\frac{dist}{max-speed}$

ALT (A*, Landmarks, Triangle inequality)

- Heuristic for A*
- Two-phase algorithm
- Preprocessing on dev machine
  - Pick set of landmarks
  - Precompute distances to/from all nodes to all landmarks
- Use as heuristic during runtime

$$c(s,l) \leq c(s,t) + c(t,l) \qquad c(l,t) \leq c(s,t) + c(l,s)$$

$$\rightarrow c(s,l) - c(t,l) \leq c(s,t) \qquad c(l,t) - c(l,s) \leq c(s,t)$$

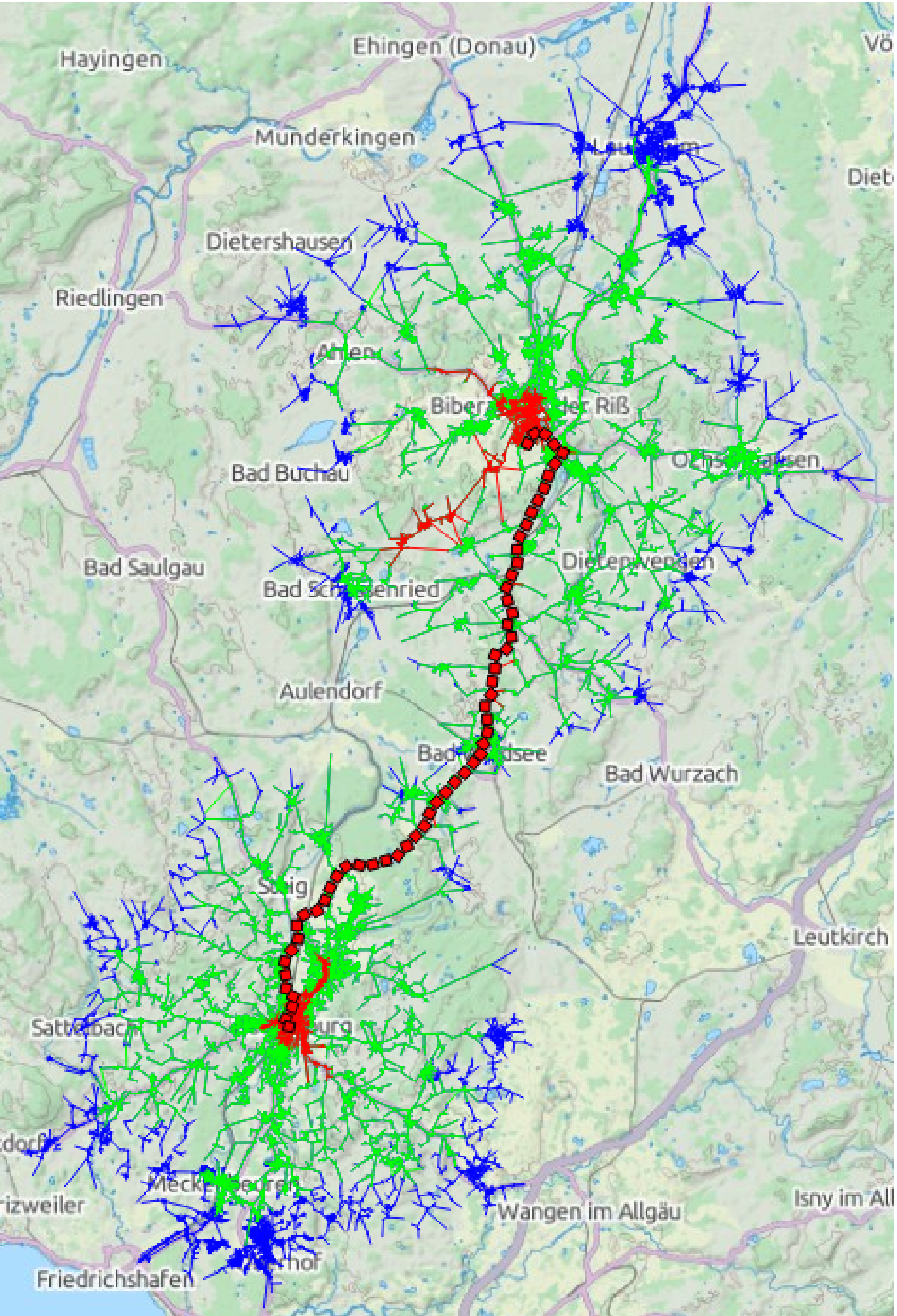


Query Times BW:

- Dijkstra: 171 ms
- A*-straight: 110 ms
- A*-lm20: 17 ms

Bidirectional search

- Invert graph, search from target to source
  → search from both directions
- Stop when searches meet
- For Dijkstra: two circles with radius $\frac{c(s,t)}{2}$
- A*: not as simple



Query Times BW:

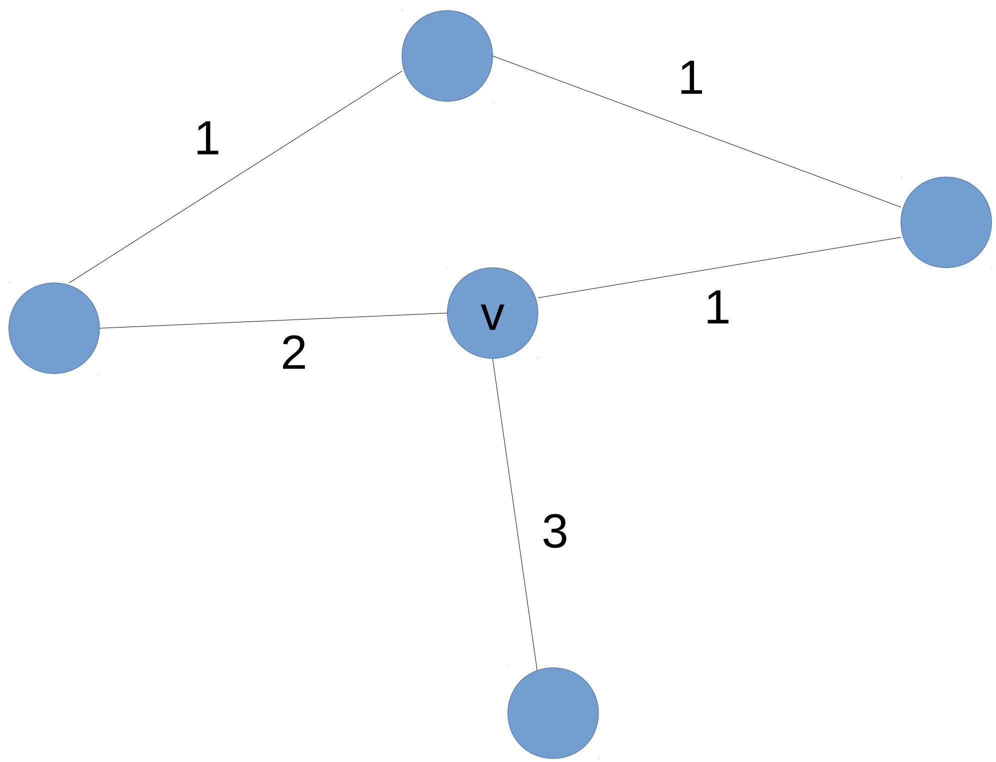
- Dijkstra: 171 ms
- A*-straight: 110 ms
- A*-lm20: 17 ms

Bidirectional:

- Dijkstra: 121 ms
- A*-straight: 89 ms
- A*-lm20: 10 ms

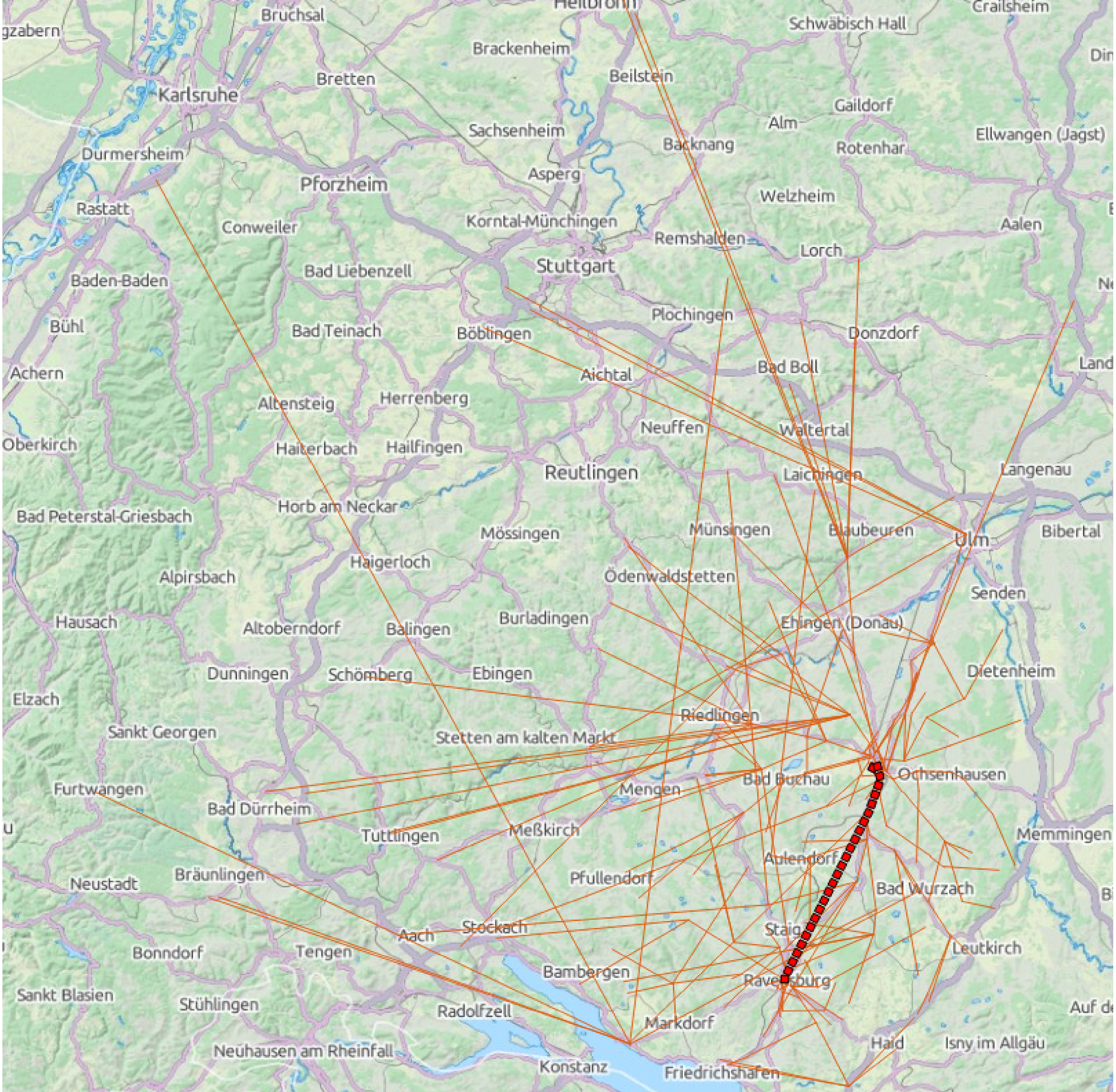
Contraction Hierarchies

- Node Contraction:



Contraction Hierarchies

- Precomputation:
  - Assume nodes ordered by *importance*
  - Contract nodes in this order:
    - → Mark node contracted
    - → Add required shortcuts
- Bidirectional Dijkstra
  - Only search upwards by importance
  - Unpack shortcuts to retrieve path



Query Times BW:

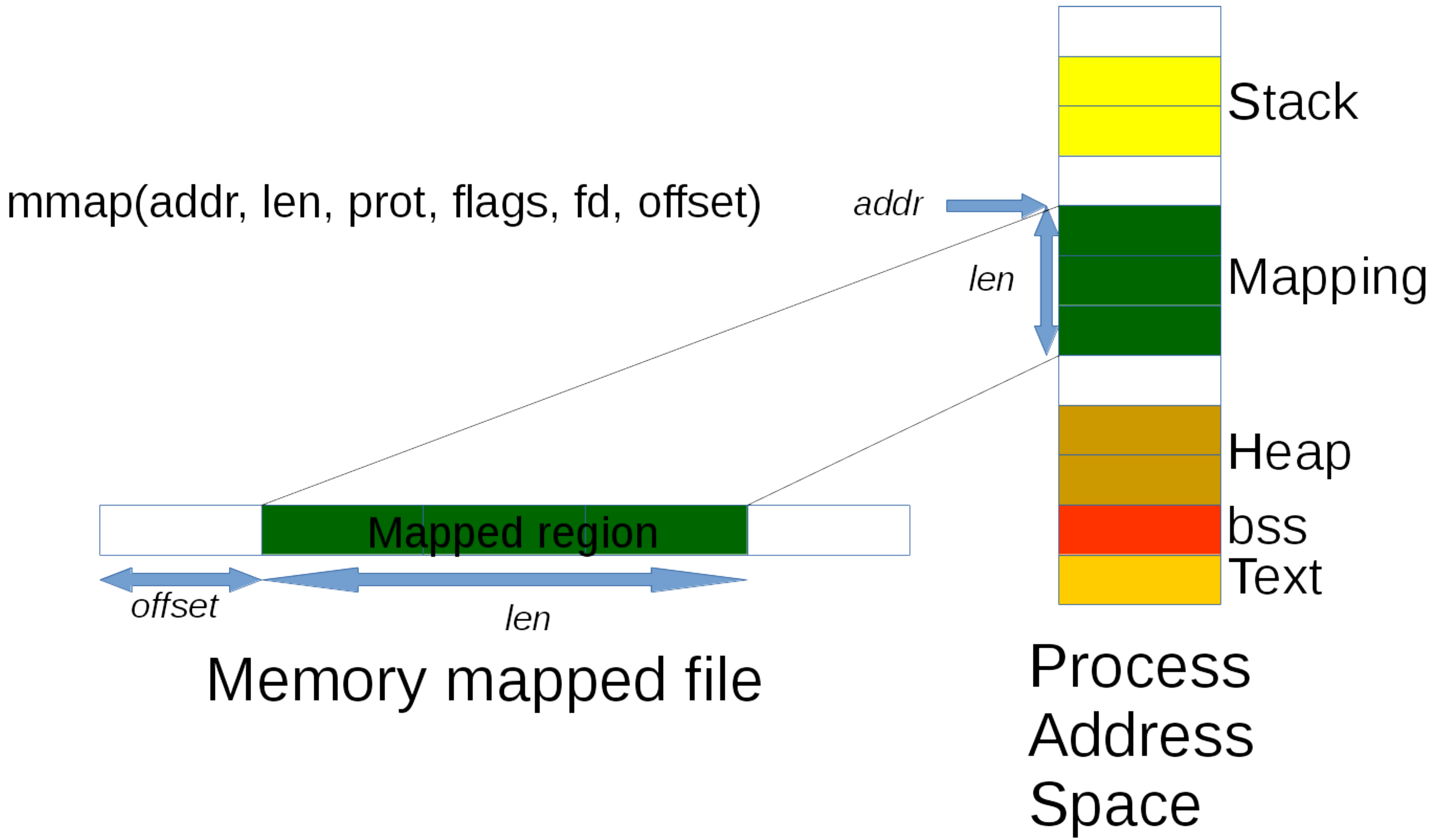
- CH: 0.5 ms

External Memory

- Secondary storage for working data
  - Persistent
  - Cheap
  - Slow

Access pattern	Block size	Main memory	SSD	HDD
Sequential Read	1 MiB	10174 MiB/s	351 MiB/s	128 MiB/s
Sequential Read	64 KiB	9688 MiB/s	343 MiB/s	128 MiB/s
Random Read	1 MiB	9388 MiB/s	317 MiB/s	62 MiB/s
Random Read	64 KiB	7193 MiB/s	119 MiB/s	7,4 MiB/s

Memory mapped files



Chunked/Managed mmap

- Problem: 32 bit systems
  - Not enough address space (Windows: max 2Gb)

    → Chunked mmap
- Problem: management
  - Managed memory mapped files
  - Custom C++ Allocator with placement-new
  - Custom pointer-type
  - Custom containers

    → auto graph = InMemoryAllocator<Graph>().allocate()

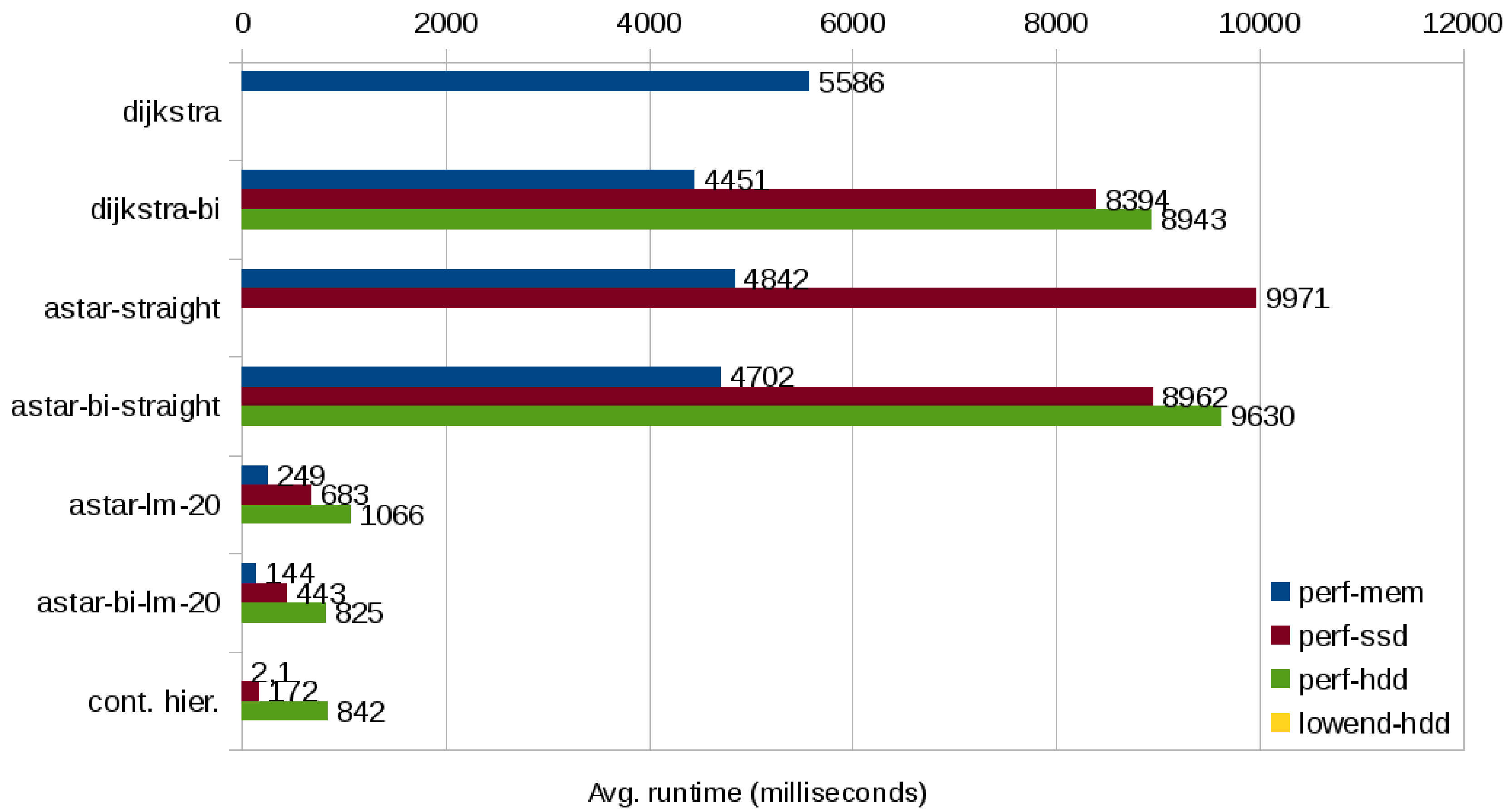
    → auto graph = MmapAllocator<Graph>().allocate()

Graph Optimization

- Sort nodes by
  - Breadth-first search
  - Locality (Z-order/Geohash, Hilbert curve)
  - CH-level (importance)
- Sort edges by
  - Breadth-first search
  - Source node

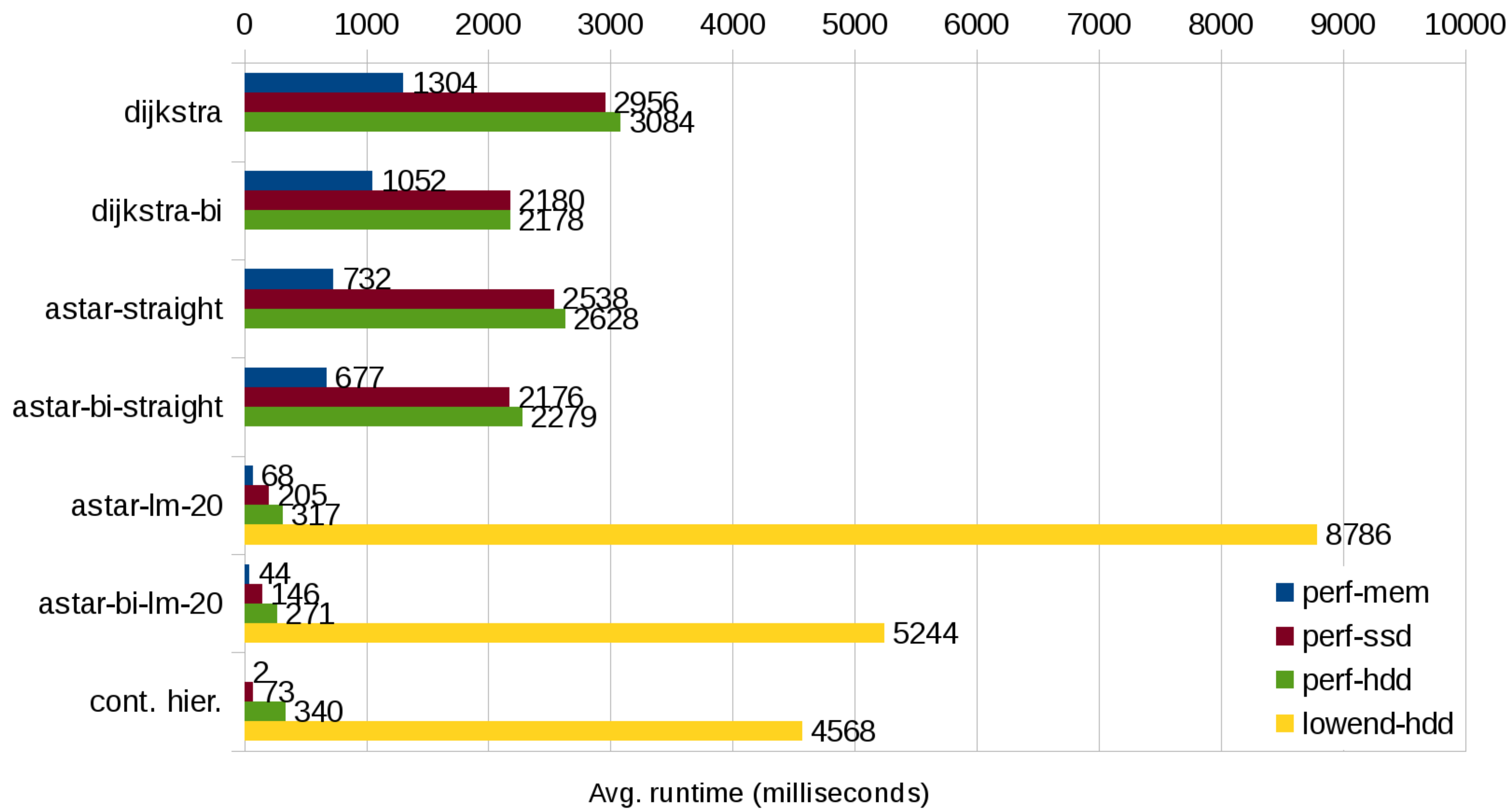
Performance

Baseline performance



Performance

Best: Geohash/Sourcenode



Performance

- CH slower than landmarks
  - Node order suboptimal
  - Sorting by CH-level:

    → 589 ms on lowend-hdd (vs. 4568 ms)

Demo