Taxi Trajectory Analysis

EDAA - G04

Diogo Rodrigues
Eduardo Correia
João Sousa

Context

Taxis are an important part of the **transportation** system, especially in **large** cities.
Over the years, the taxi industry has been **evolving** rapidly due to the increasing **efficiency needs**.

This resulted in installing mobile data terminals in each vehicle and provide information on **GPS** localization.

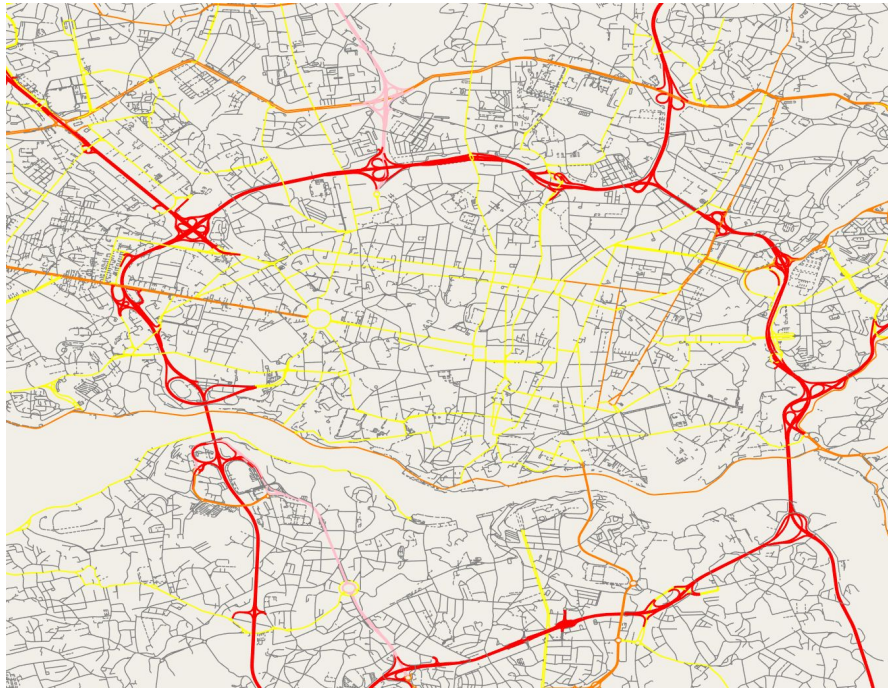
Dataset

Map

A graph representing road network in Porto Metropolitan Area (AMP)

Data extracted from OpenStreetMap

- Original file: 289.5 MB
- Filtered data:
  - `AMP.nodes`: 9.0 MB; 304,345 nodes
  - `AMP.edges`: 8.8 MB; 568,735 directed weighted edges



Dataset

Trips

A list of all 1,710,669 **taxi trips** of all 442 taxis in the city of Porto from 01/07/2013 to 30/06/2014

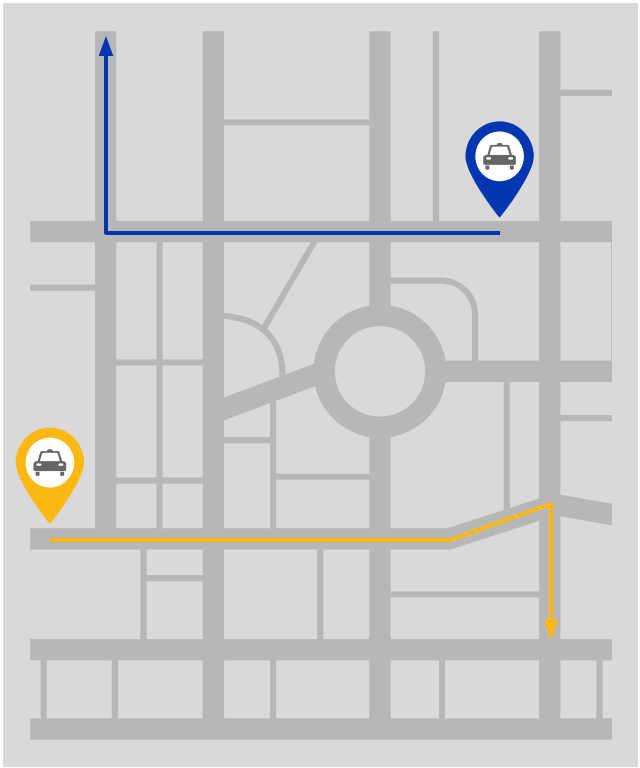
Data from **Kaggle** competition PKDD 15 (I)

- Original file: 1.9 GB
- Filtered out runs with:
  - Missing data: 10
  - Speed errors: 173,909
  - Coordinate errors: 5,609
- Processed file: 1.5 GB
  - 1,531,135 trips
  - 72,227,758 coordinates

Using 200k trips

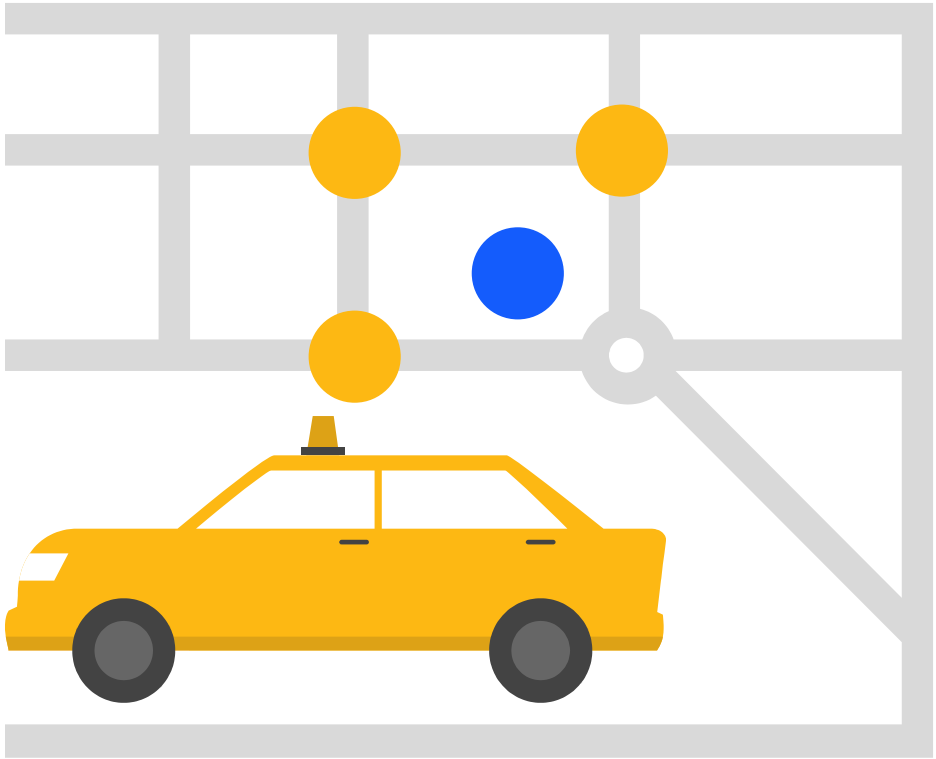
Goals

The goal of this project would be to analyze **taxi trajectories**. Taking into account the **size** of the data we will be dealing with, we have to implement **efficient data structures** and **algorithms**.



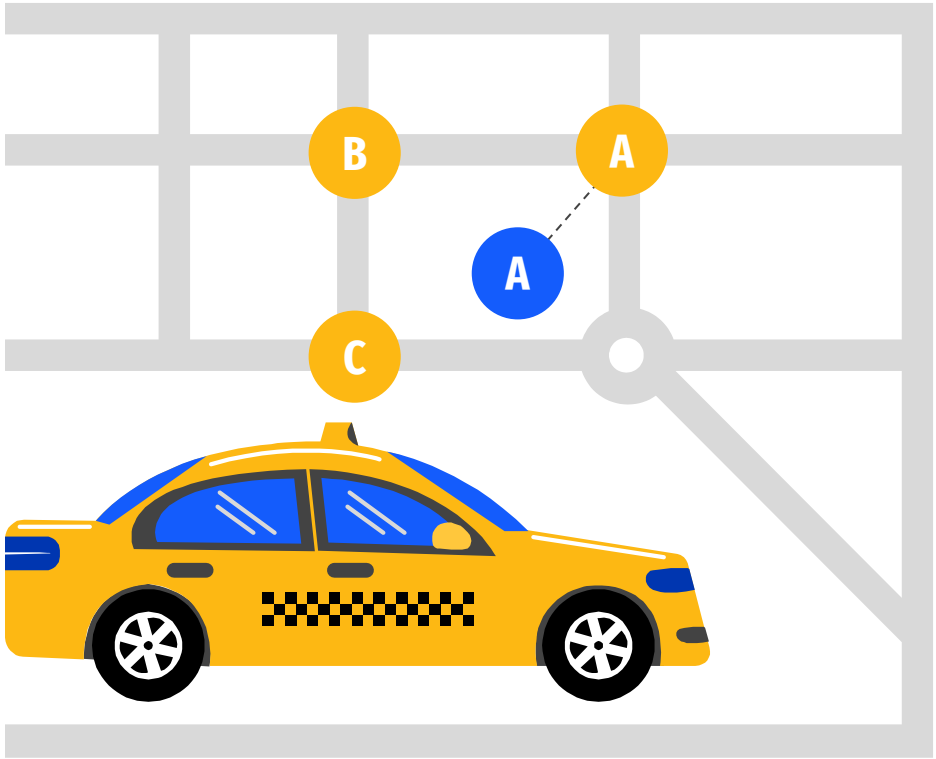
Problem definition

However, we're struck with the problem of **mapping GPS coordinates** with **network nodes** that represent the real-life roads.



Solution

Our problem might be reduced to a **nearest neighbour search (NNS)**, in which for each **taxi** position we locate the **nearest node** in the **graph** that represents the **roads network**.



Data structures and algorithms

k-d trees

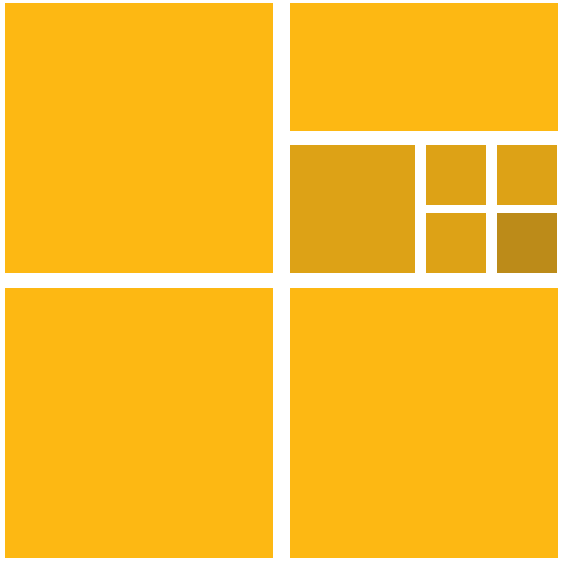
Binary tree which splits the dataset using alternating axes

Build data structure:

- Time: $O(N \log N)$
- Space: $O(N)$

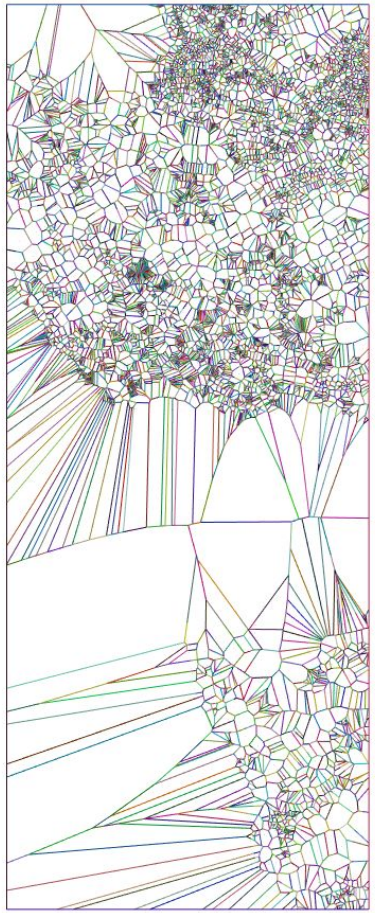
Query:

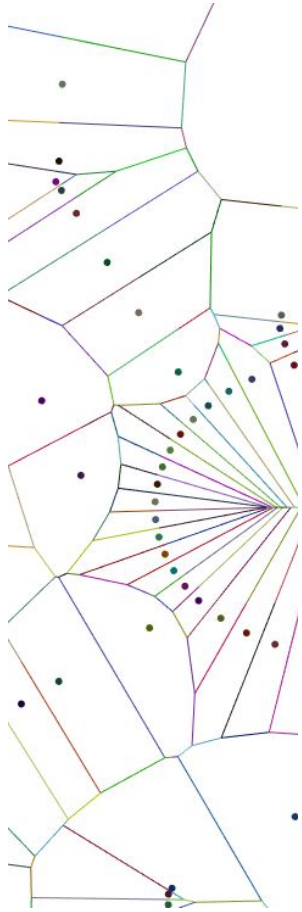
- Time: $\Theta(\log N)$, $O(N)$
- Space: $O(\log N)$



Voronoi Diagram

We decided to use Fortune's Algorithm to construct a Voronoi diagram of our dataset, however, due to the huge quantity of nodes and the precision of the coordinates, we ran into some floating point errors which ultimately prevented us from using it practically.





VStripes

Divide plane into vertical stripes of width Δ, insert points into correct stripes by *x*, and sort points in each stripe by *y*.

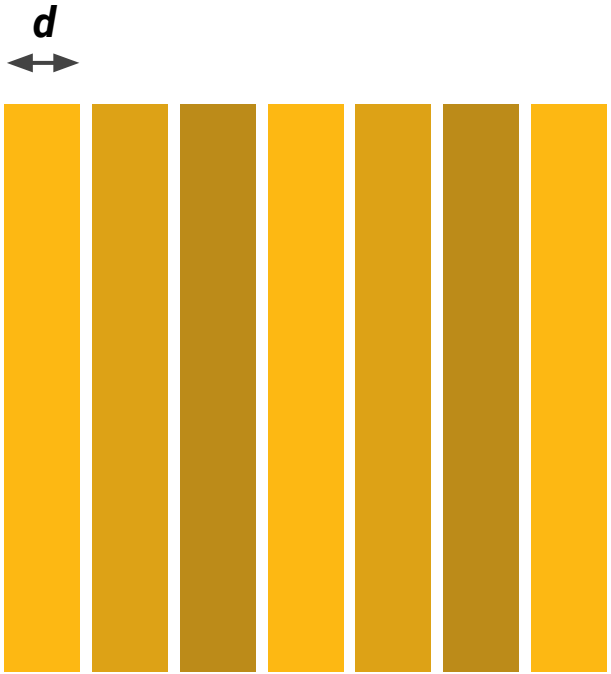
Build data structure:

- Time: ***O(N log N)***
- Space: ***O(N)***

Query:

- Time: ***Θ(log N)***, ***O(N)***
- Space: ***O(1)***

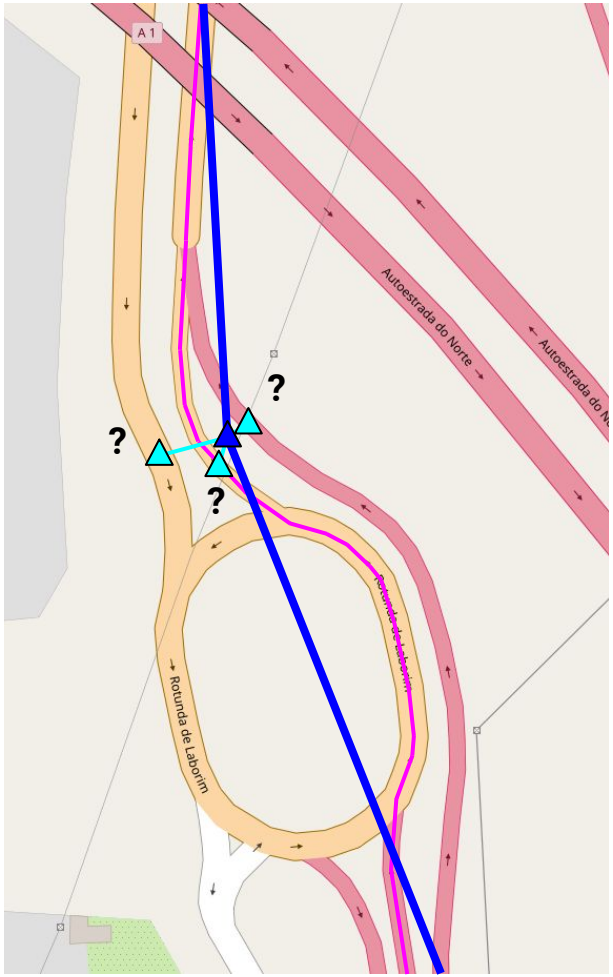
Apply DeepVStripes to solve issue of estimating Δ. DeepVStripes is a hierarchical extension of VStripes with different values of Δ



Hidden Markov Model

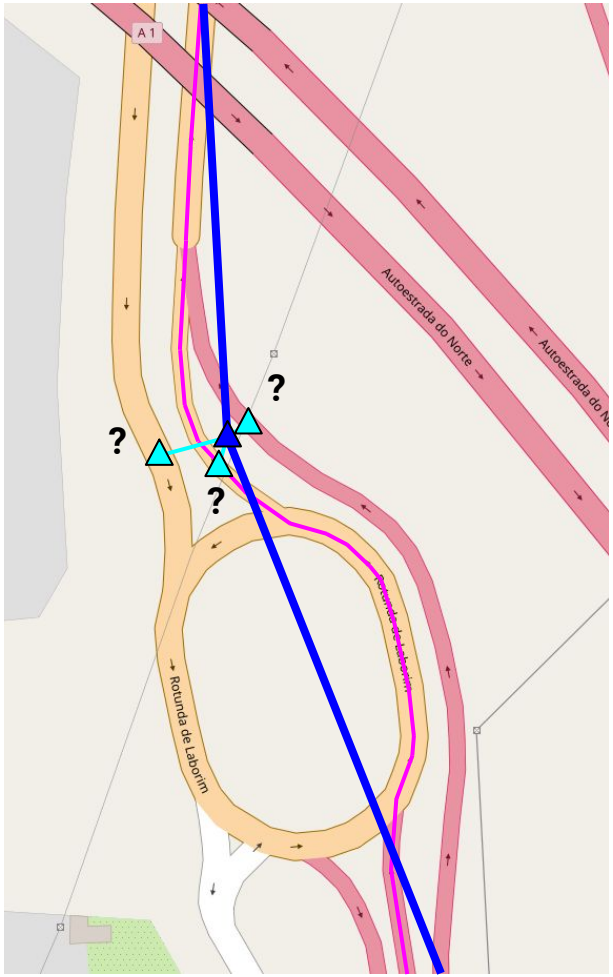
Nearest neighbor provides fast but awful answers: **we not always want to match a GPS location to NN!**

Add contextual information encoding what is reasonable and what is not, for algorithm to decide which of the *k* nearest neighbors (or neighbors within radius *r*) to match



Hidden Markov Model

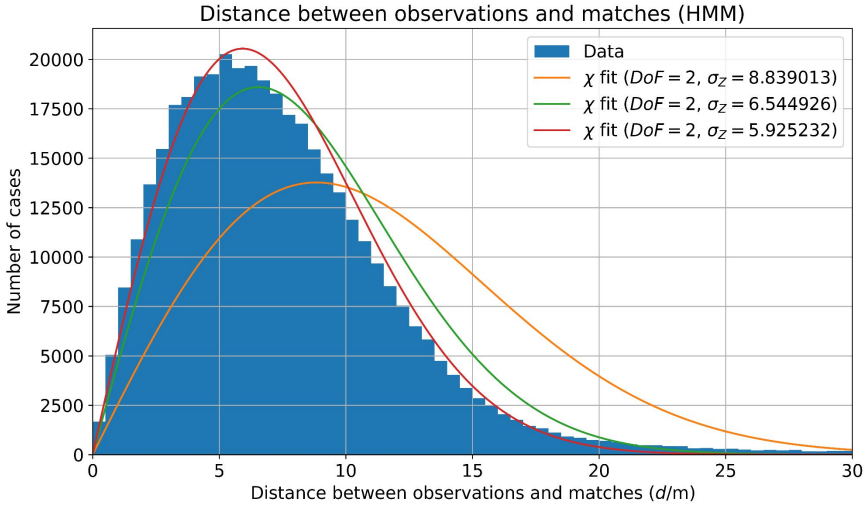
- Markov process X whose states $x_i \in S$ are hidden (real roads where taxi may be)
- We can observe them through another Markov process Y of observations $y_i \in O$ (GPS locations)
- We know the emission matrix B_{tj}: probability of hidden state $s_j \in S$ generating a certain observation $o_t \in O$ (related to GPS error)
- We know the transition matrix A_{ij}: probability of transitioning between hidden states $s_i, s_j \in S$ (some paths in the road network are more reasonable than others)



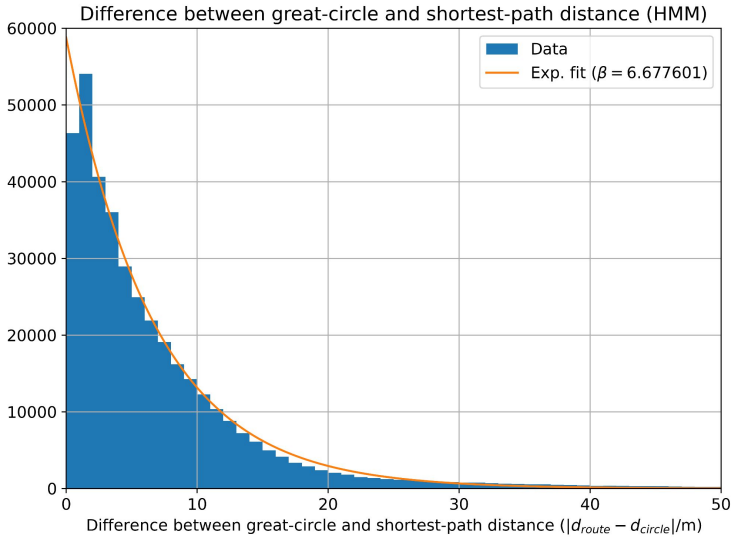
Hidden Markov Model

Now we have to solve the two last problems:

- Probability B_{ij} of hidden state generating an observation?
  - Related to GPS error, which is normal, has zero-mean and std σ_z
  - **P(*o* | *s*) = PDF$_{N(0, \sigma z)}$(dist(*o*, *s*))**



- What is a reasonable path? (aka A_{ij})
  - Great-circle distance between consecutive GPS observations, $\boldsymbol{d_{circle} = dist(y_t, y_{t+1})}$
  - Shortest-path distance between two consecutive matches: $\boldsymbol{d_{route} = ss(x_t, x_{t+1})}$
  - They should be the same. Makes sense, right?
  - $|\boldsymbol{d_{route}-d_{circle}|}$ follows exponential distribution with scale parameter β
  - $\boldsymbol{P(y_t \mid x_t) = PDF_{exp(\beta)}(|d_{route}-d_{circle}|)}$



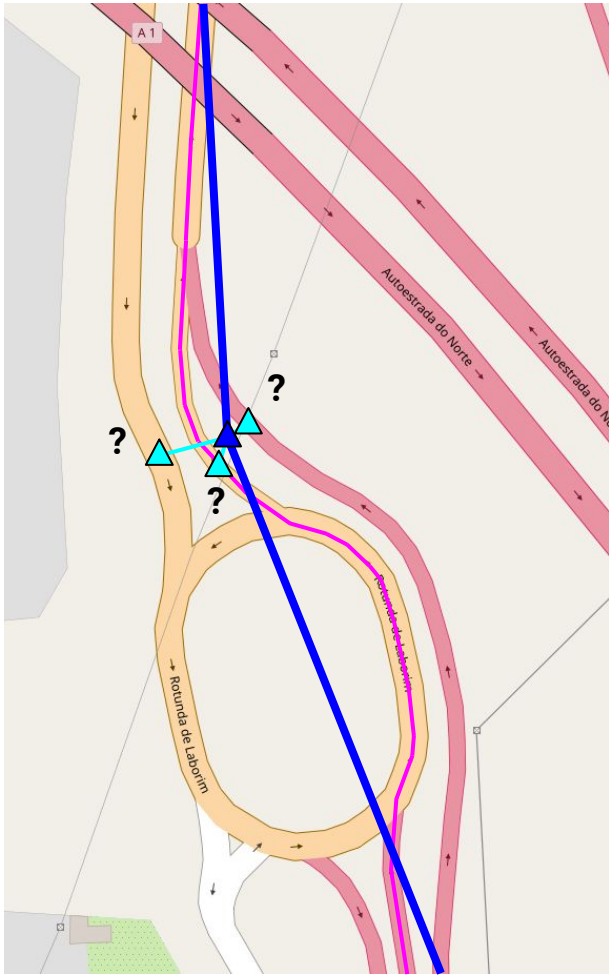
Viterbi algorithm

Given transition matrix A, emission matrix B and vector of initial probabilities Π, determines the **Viterbi path**, which is the most likely sequence of hidden states (aka the most likely path the taxi took)

Simple DP:

- $\mathbf{P}(0, j) = \Pi_j$
- $\mathbf{P}(t, j) = \max_i \mathbf{P}(t-1, i)\, A_{i,j,t}\, B_{t,j}$
- Answer: $\max_i \mathbf{P}(T, i)$

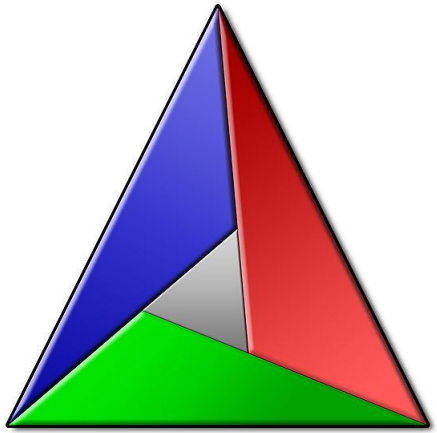
T observations, N possible states:
$O(TN^2)$ time, $O(TN)$ memory (can be optimized)



Implementation

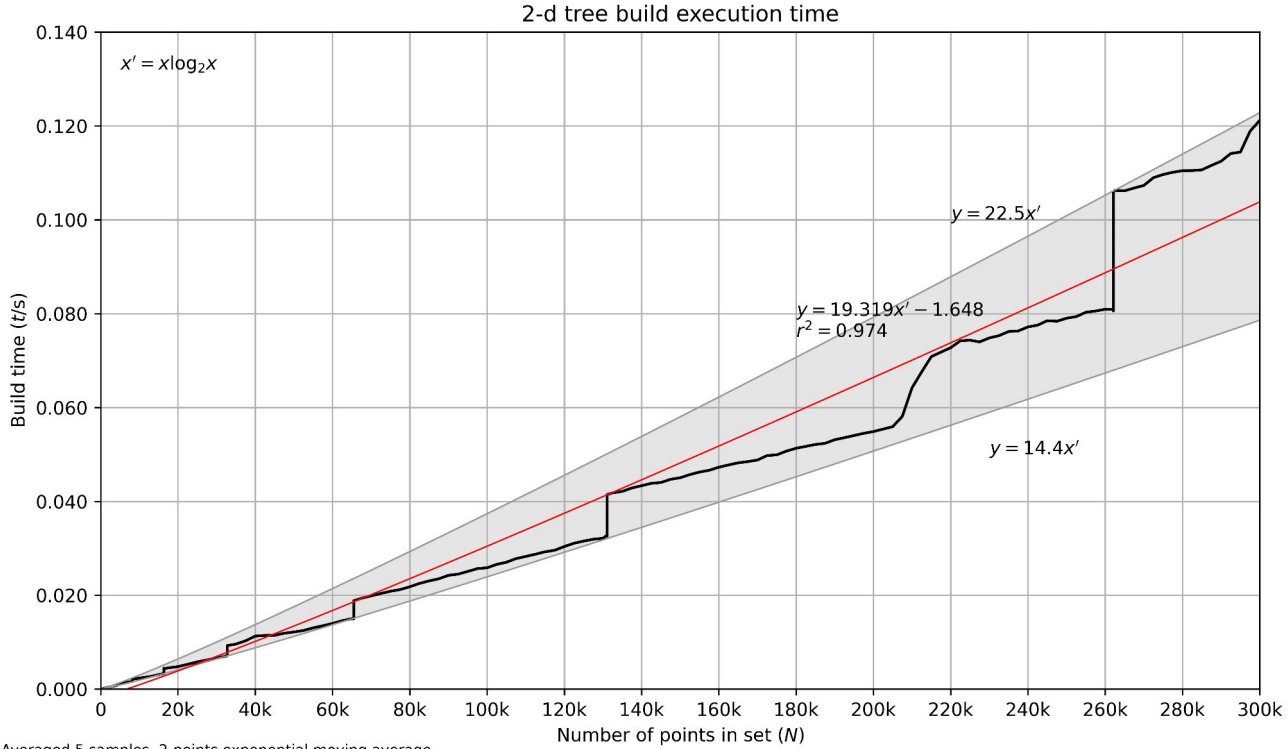
Implementation

- C++
- CMake build system
- SFML library for graphical display



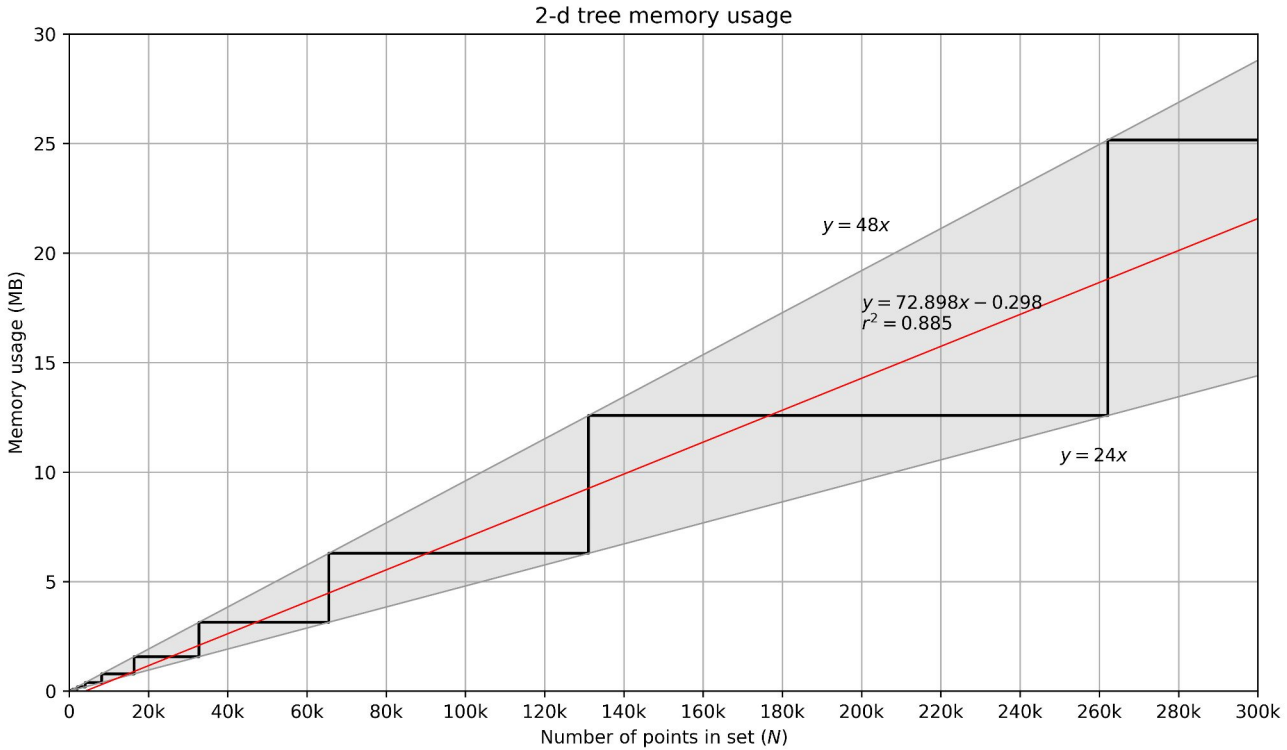
Empirical analysis

2-d trees

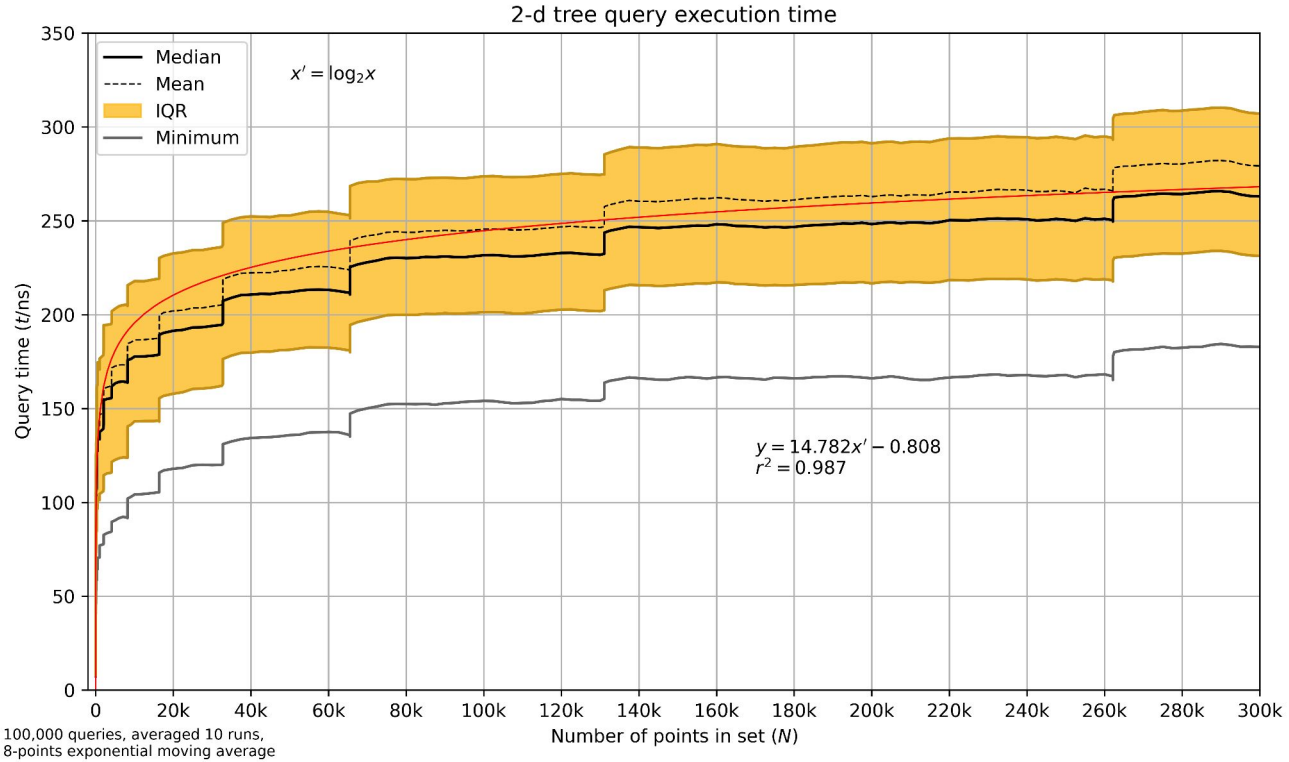


Averaged 5 samples, 2-points exponential moving average

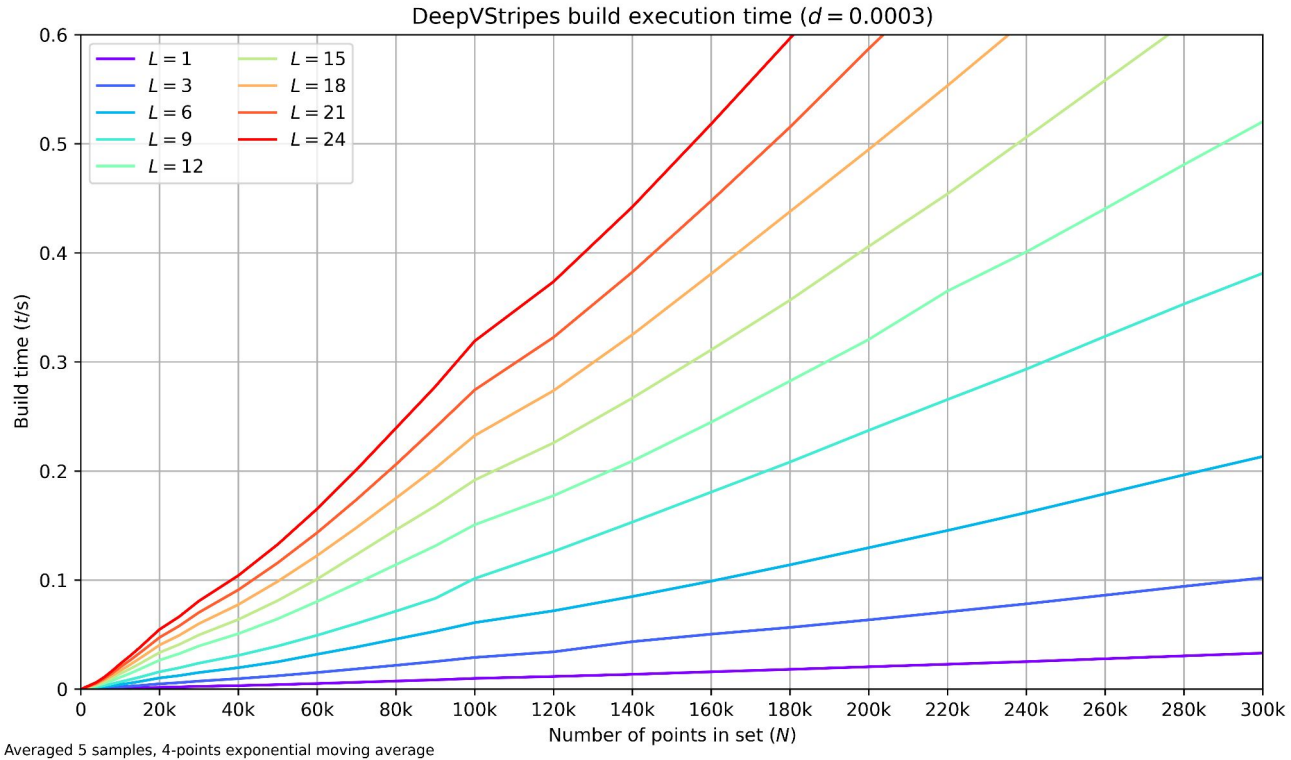
2-d trees


2-d tree memory usage
Memory usage (MB)
Number of points in set (N)
y = 48x
y = 72.898x − 0.298
r² = 0.885
y = 24x
0
5
10
15
20
25
30
0
20k
40k
60k
80k
100k
120k
140k
160k
180k
200k
220k
240k
260k
280k
300k

2-d trees


2-d tree query execution time
Median
Mean
IQR
Minimum
$x' = \log_2 x$
$y = 14.782x' - 0.808$
$r^2 = 0.987$
Query time (t/ns)
Number of points in set (N)
100,000 queries, averaged 10 runs,
8-points exponential moving average
0
50
100
150
200
250
300
350
0
20k
40k
60k
80k
100k
120k
140k
160k
180k
200k
220k
240k
260k
280k
300k

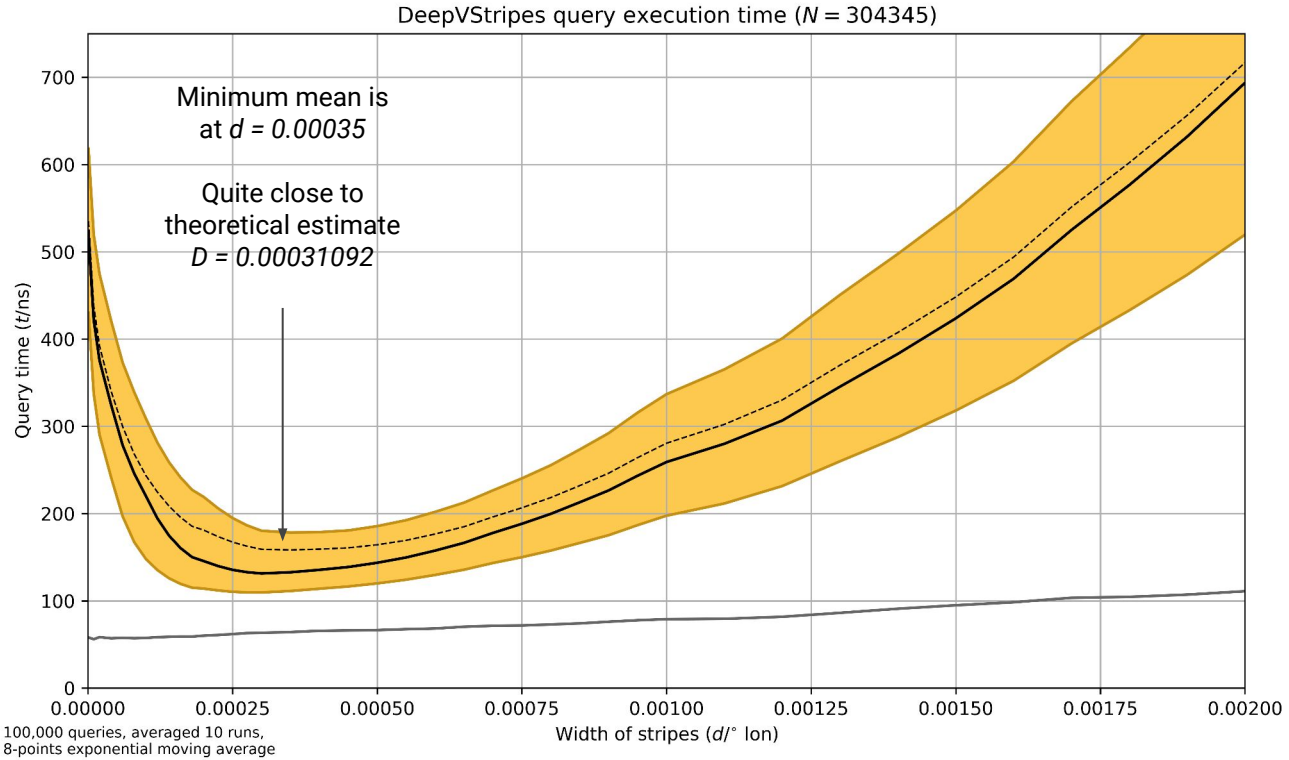
DeepVStripes



Averaged 5 samples, 4-points exponential moving average

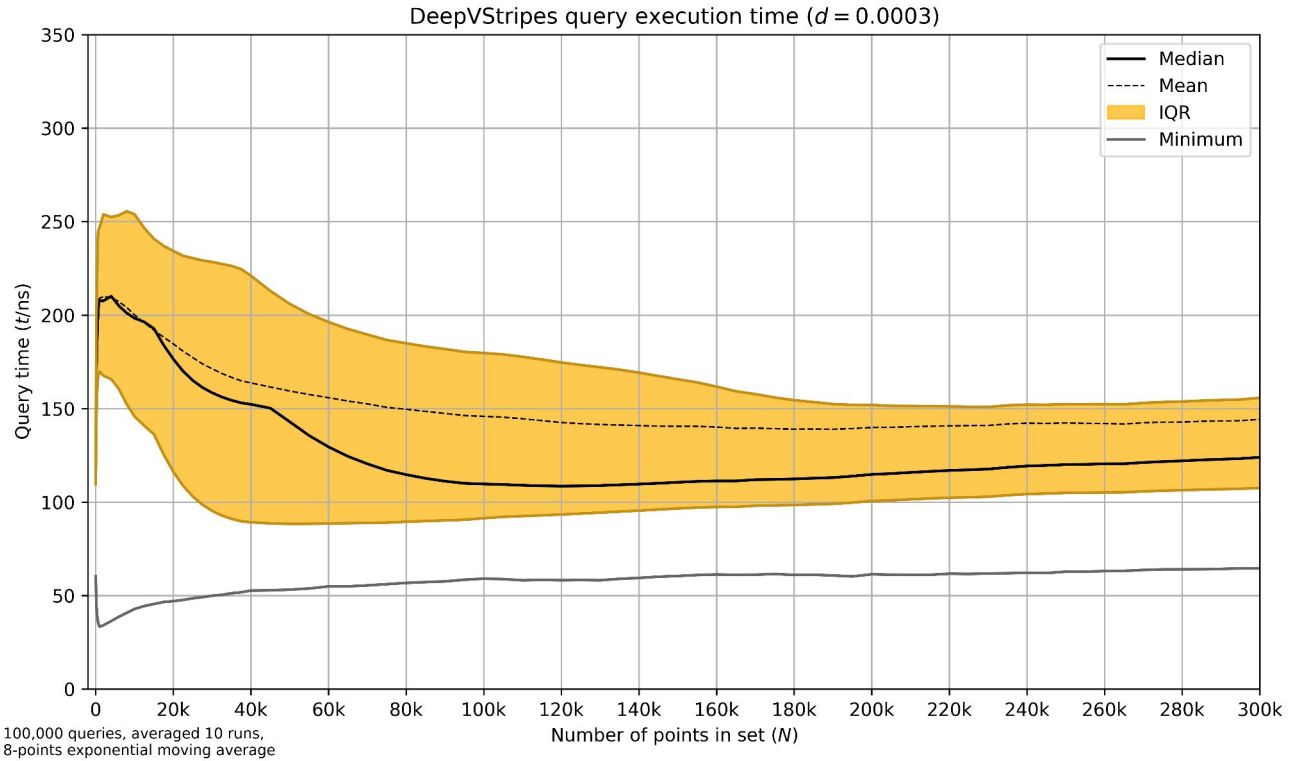
DeepVStripes

Here, climbing up to correct *d* is dominant: ***O*(log(*D*/*d*))**

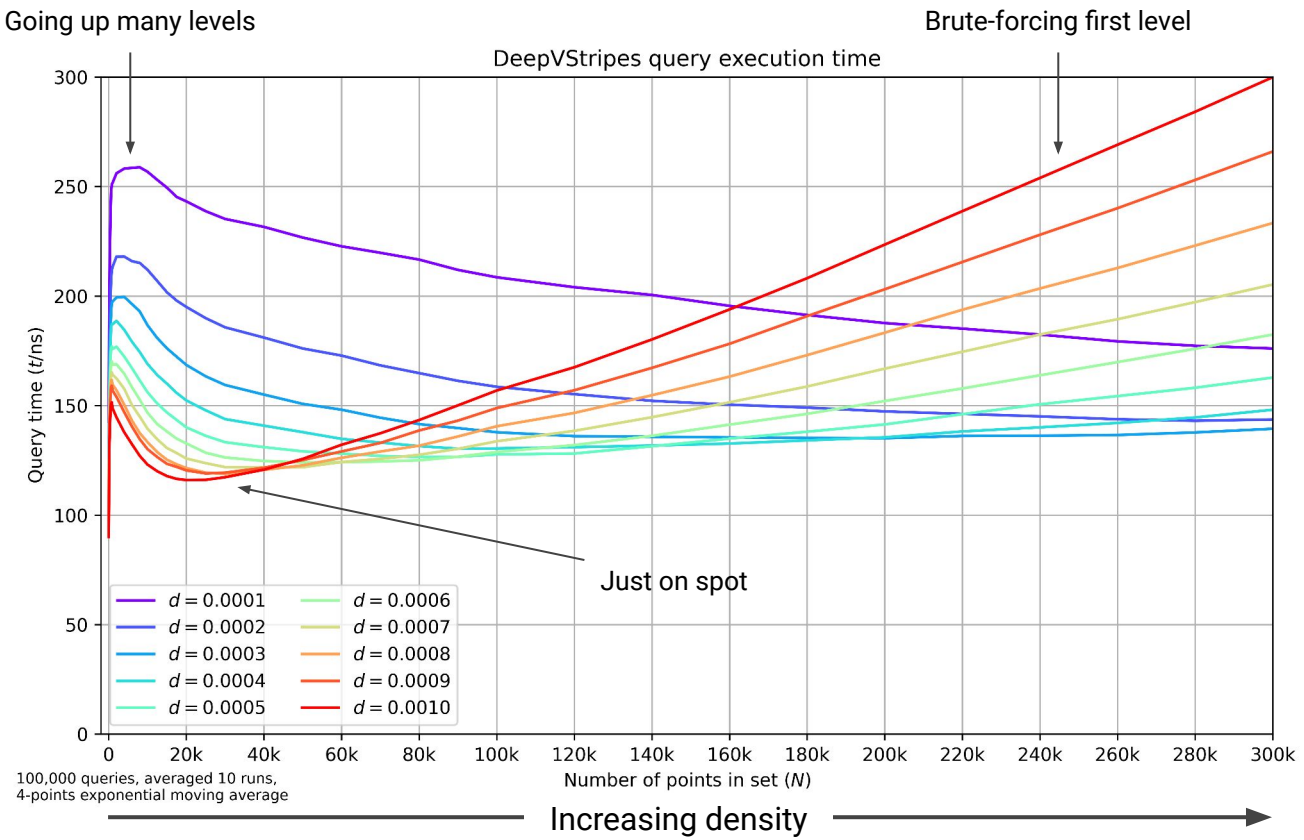


Here, number of points in $4d^2$ is dominant: ***O*(*d*²)**

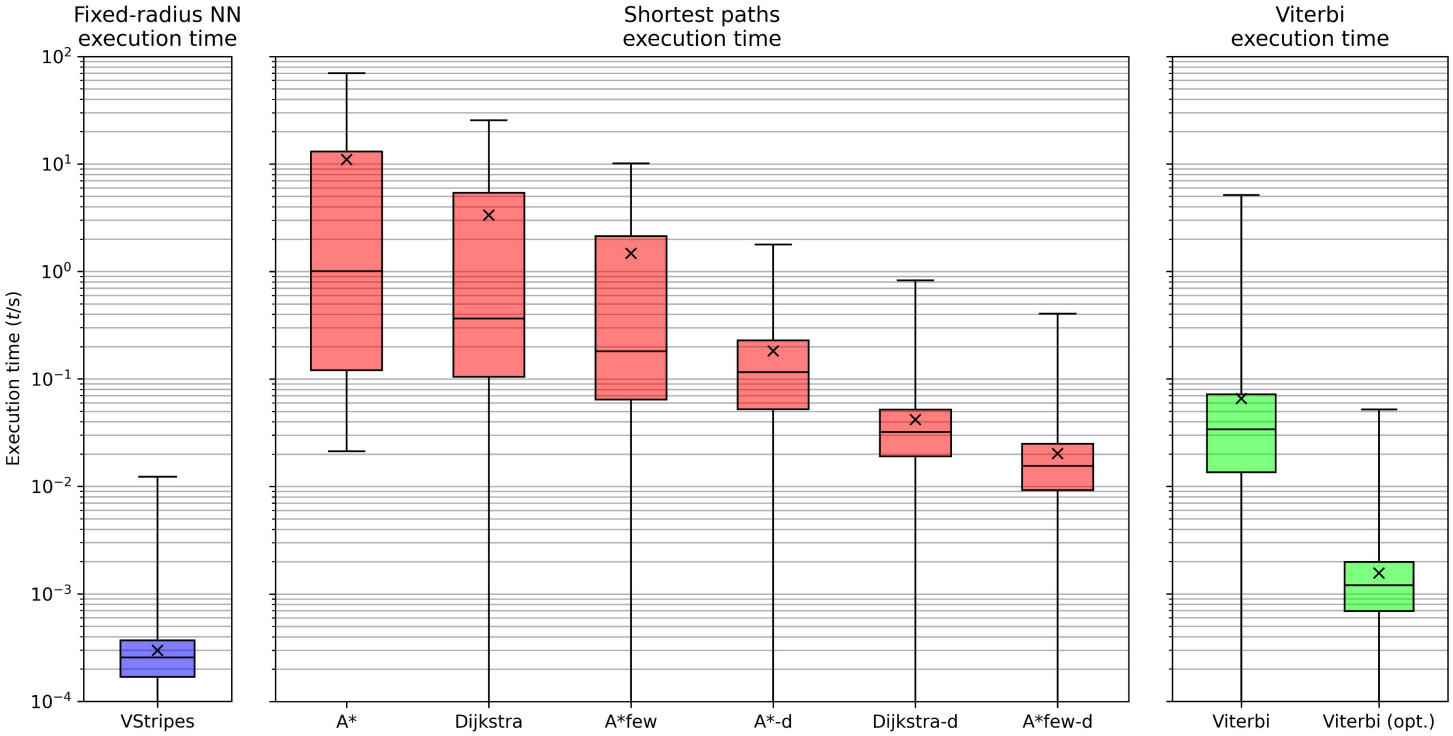
DeepVStripes



DeepVStripes



Hidden Markov Model

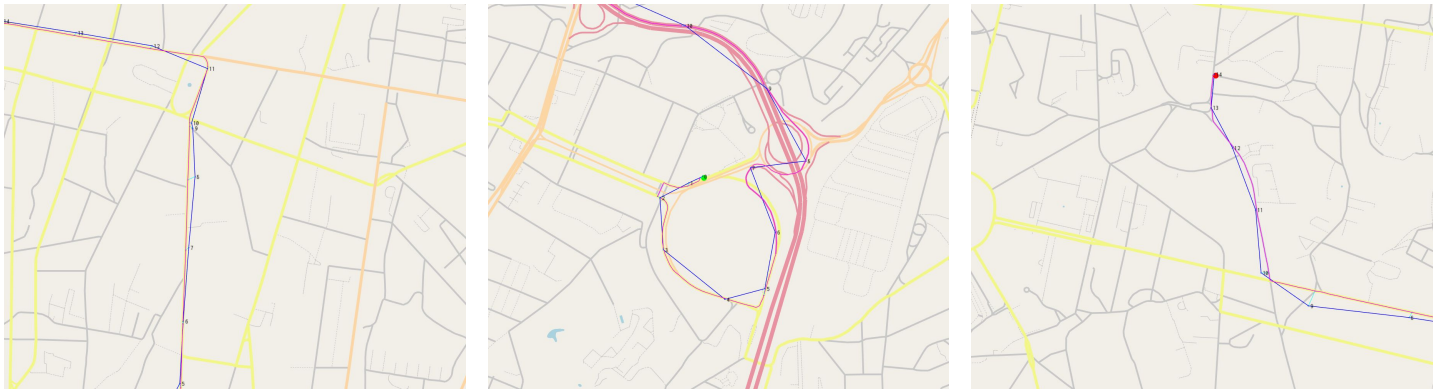


NN vs HMM

Qualitative evaluation

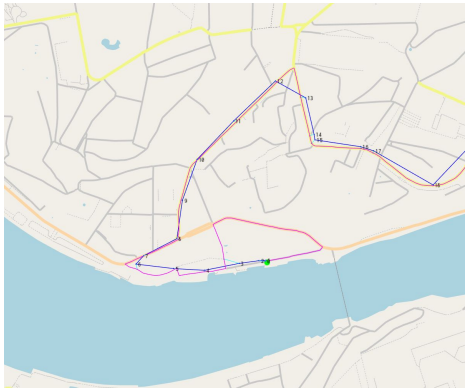
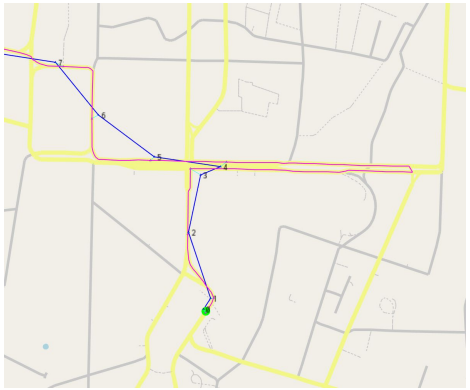
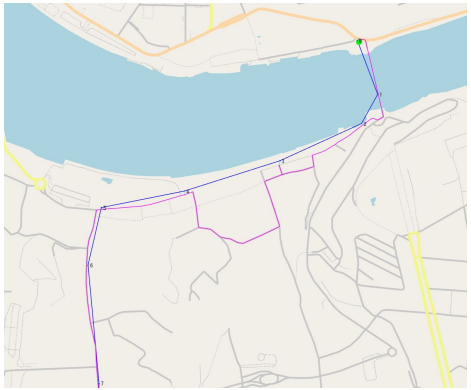
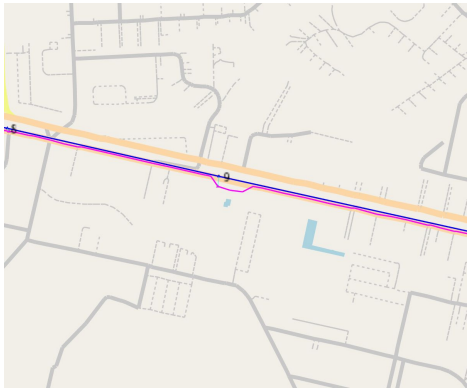
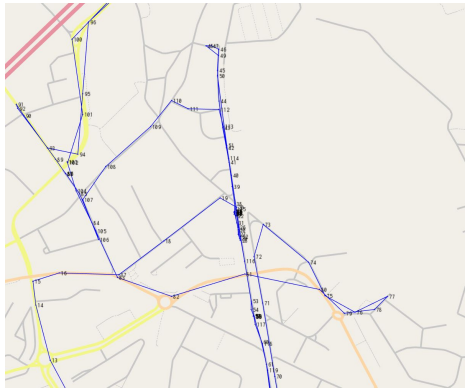
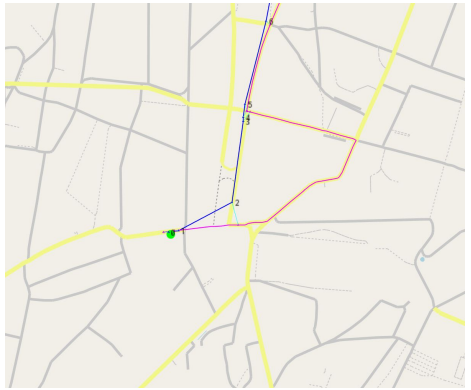
NN

HMM



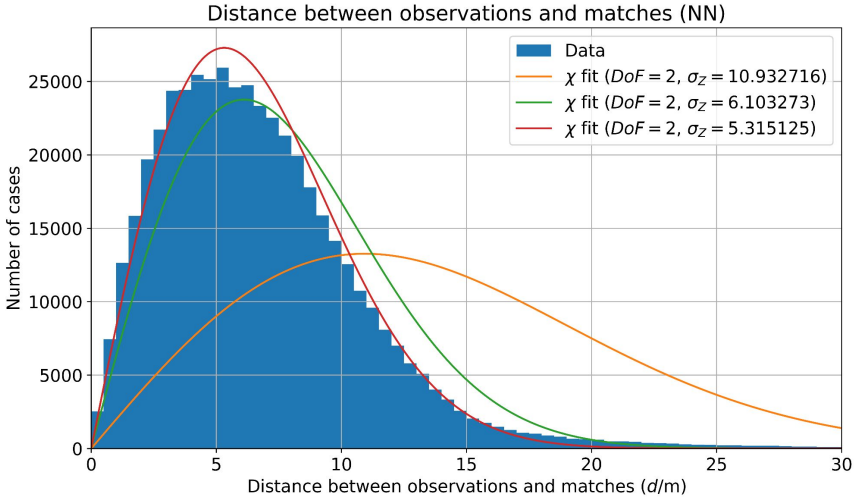
HMM

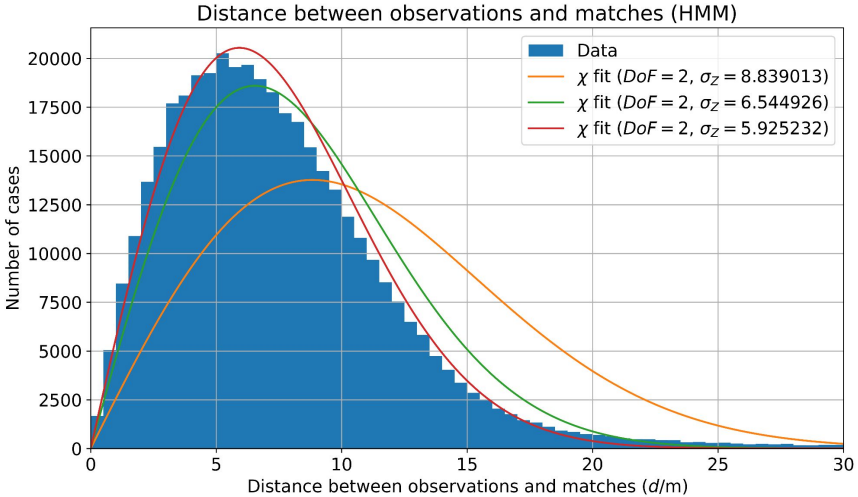
Qualitative evaluation

NN vs HMM

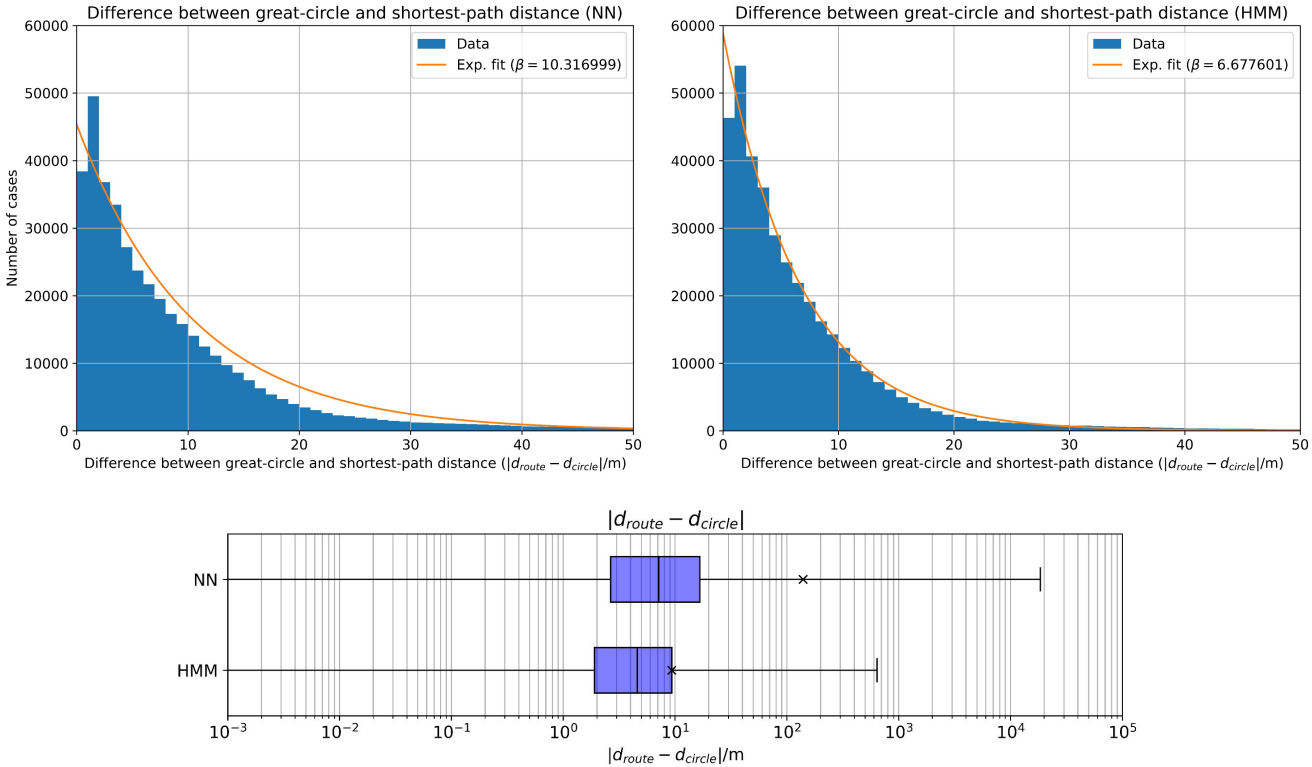
Distance between observations and matches (faithfulness)





NN vs HMM

Distance between great-circle and shortest-path distance (reasonableness)



Conclusion

HMM >> NN

Q&A