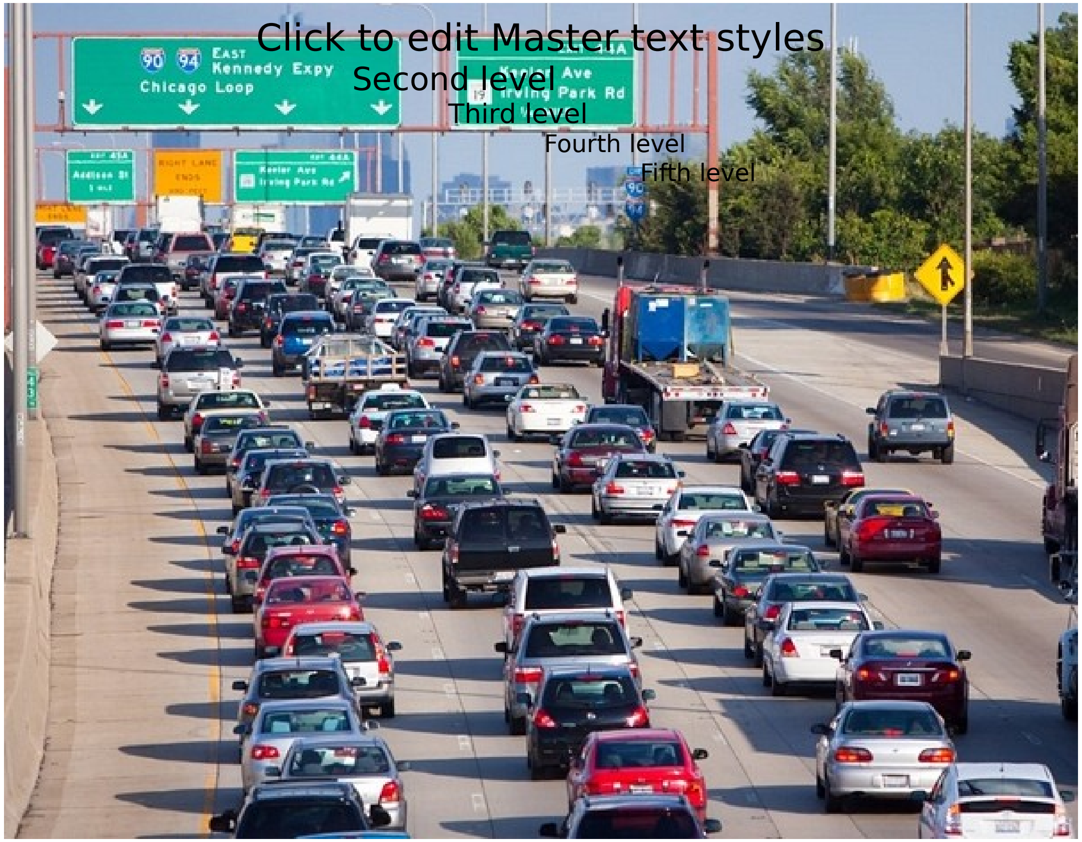
A Continuous Query System for Dynamic Route Planning

Published in ICDE 2011

Nirmesh Malviya (IIT Kanpur, MIT)
Samuel Madden (MIT)
Arnab Bhattacharya (IIT Kanpur)

Traffic congestion is a real problem

- $115 billion congestion cost in 2009 (in USA)
- 34 hours of yearly peak delay for average commuter



Click to edit Master text styles
Second level
Third level
Fourth level
Fifth level

Continuous Routing Queries

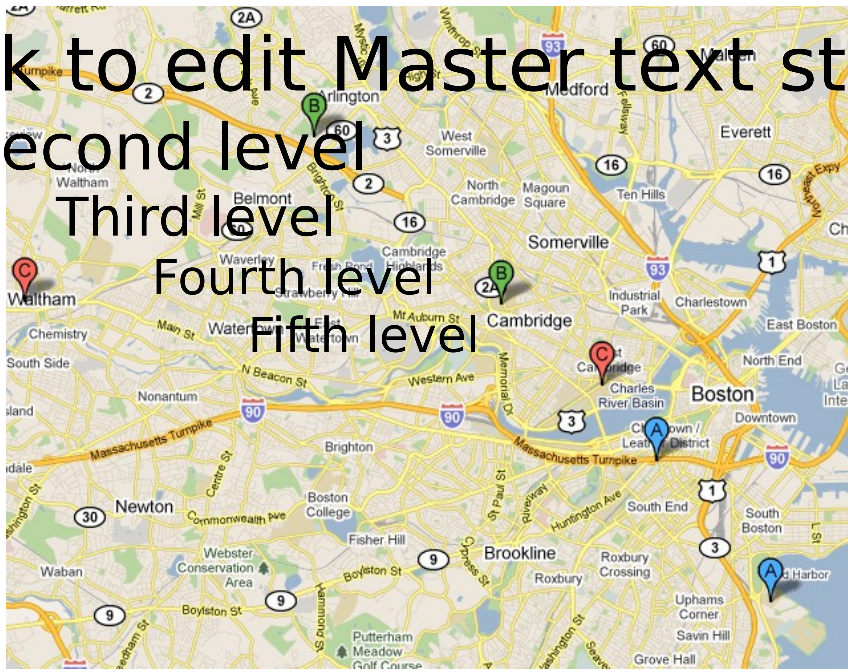
Click to edit Master text styles

Second level

Third level

Fourth level

Fifth level

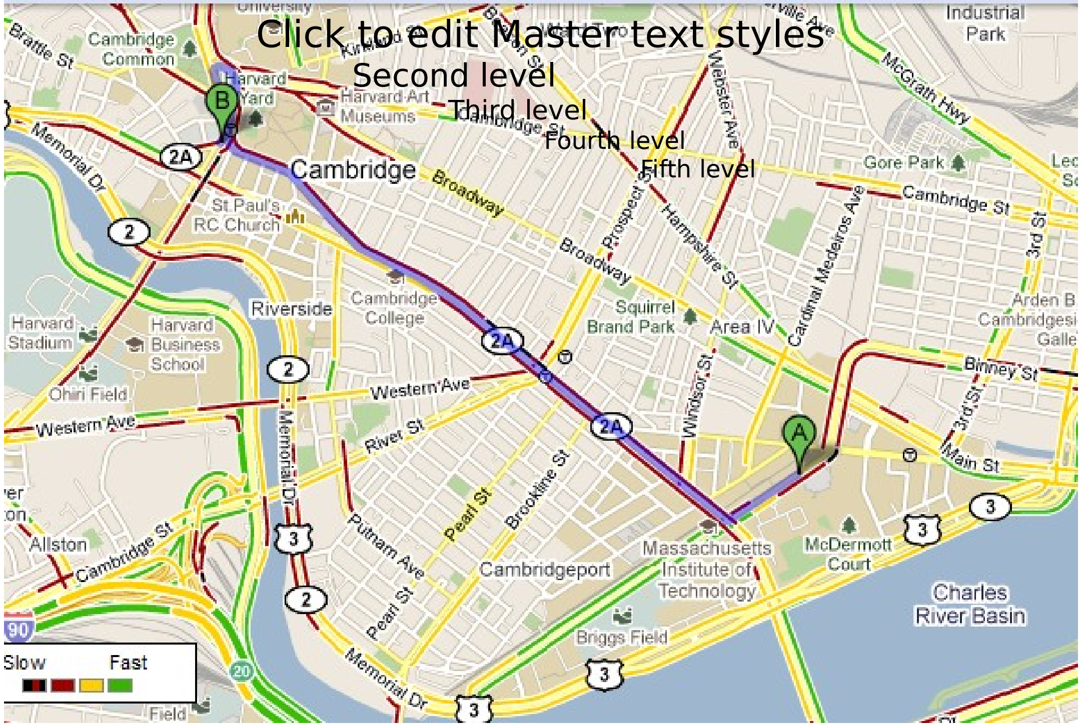


A traffic aware *dynamic* route planning service

- User registers pair <source *s*, destination *t*>
- Continuously monitors *s-t* routes as traffic delays change
- Keeps user updated with a *near-optimal s-t* route
- Scalable

Doesn't Google Maps solve this problem already?

- Routing based on pre-processed edge weights
- Ad-hoc routing only
- Real time traffic layer overlay for visualization



Roadblocks

Can't do repeated shortest-path recalculation

- Not *scalable*

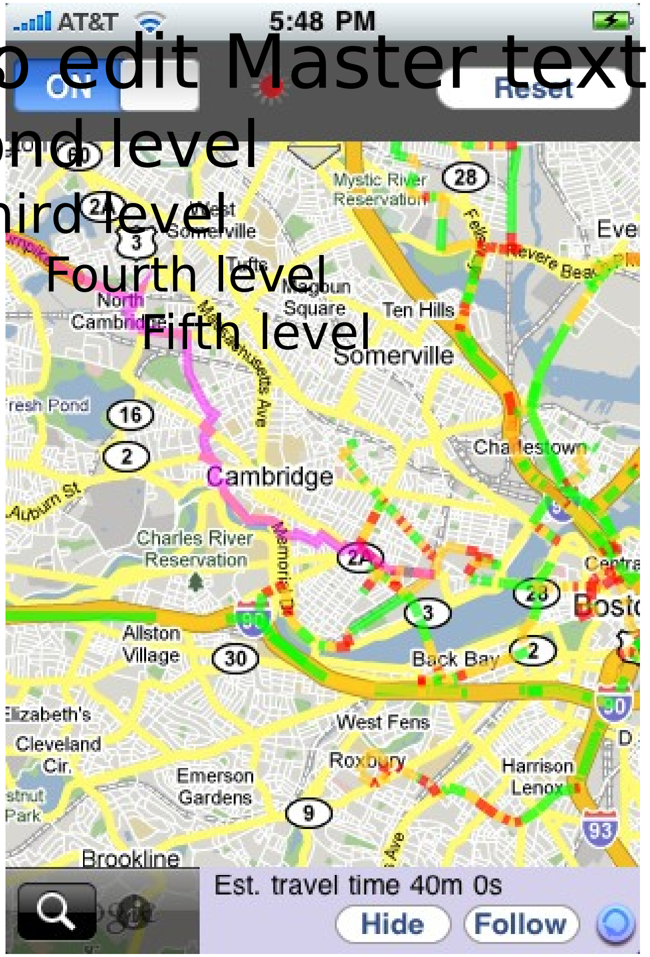
Academic work on *incremental* SP algorithms

- High space overhead (memory-resident)
- High computation overhead
- Tries to get optimal (which is not critical here)

Solution: A Continuous Query System

Click to edit Master text styles
- Second level
  - Third level
    - Fourth level
      - Fifth level

Click to edit Master text styles
- Second level
  - Third level
    - Fourth level
      - Fifth level



Key Ideas

Traffic updates affect only a small part of the road network
Mix of pre-computation and on-the-fly route calculation
Update only when there are significant delay changes

Our Algorithms

K-Candidate-Paths

Proximity-based approach

K candidate paths

Pre-compute K different routes
Dynamically select the best candidate as delays change
- Re-computation limited to K routes

Want changes in traffic delays to not adversely affect all K routes simultaneously
Different strategies for computing K paths

Yen's algorithm

Find the shortest path
Pick a vertex v from the path
Delete edges joining v in the path
Find the shortest path again

K-Variance

Perturb randomly the edge weights
For every perturbation, find the shortest path
Sample the edge cost from a Gaussian distribution
- Parameters learnt from history

Run the algorithm until K *distinct* paths are found
Lots of overlap in paths

Partially overlapping paths

Click to edit Master text styles

Second level

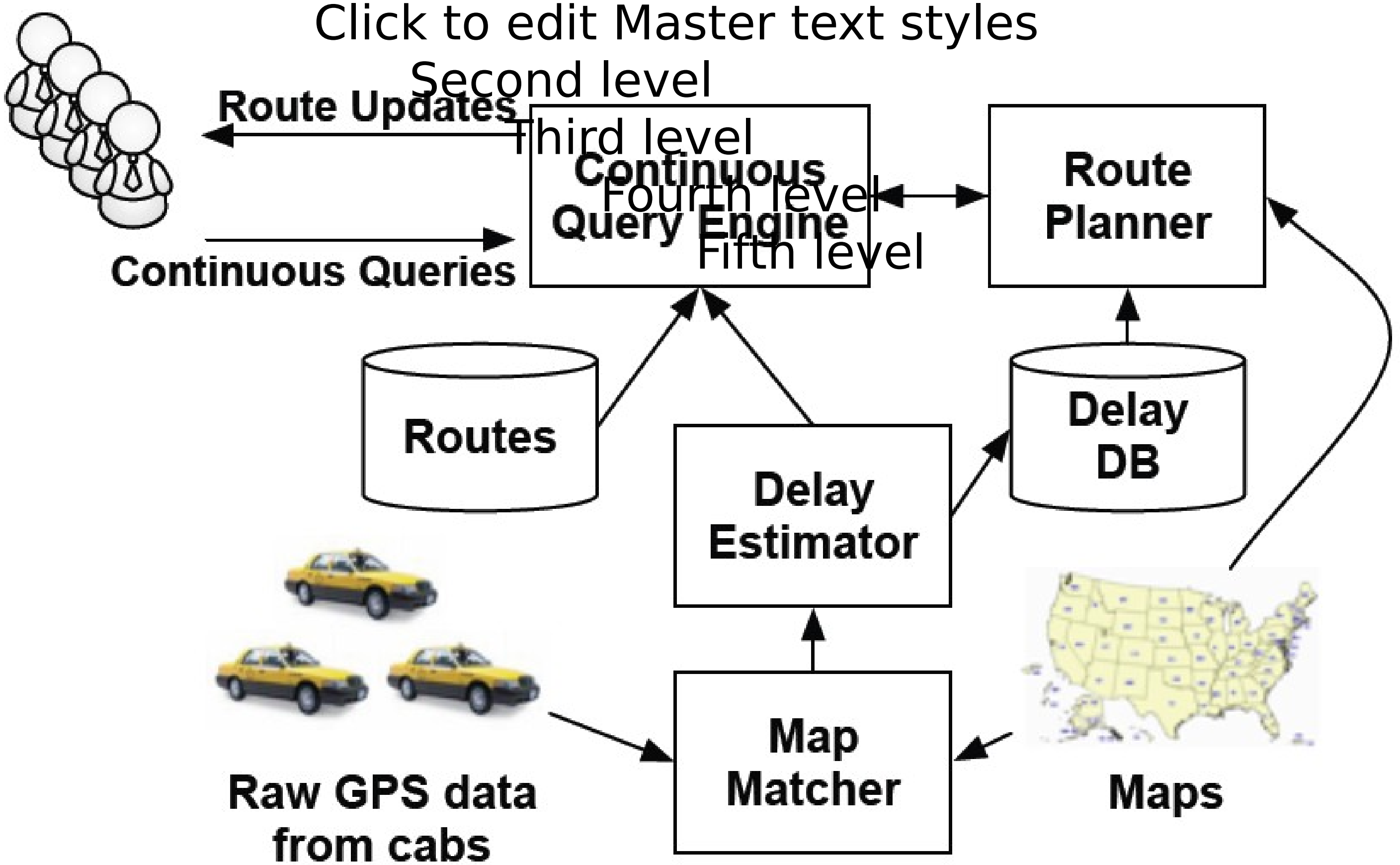
Third level

Fourth level

Fifth level

System Architecture

Click to edit Master text styles
Second level
Third level
Fourth level
Fifth level



K-Statistical

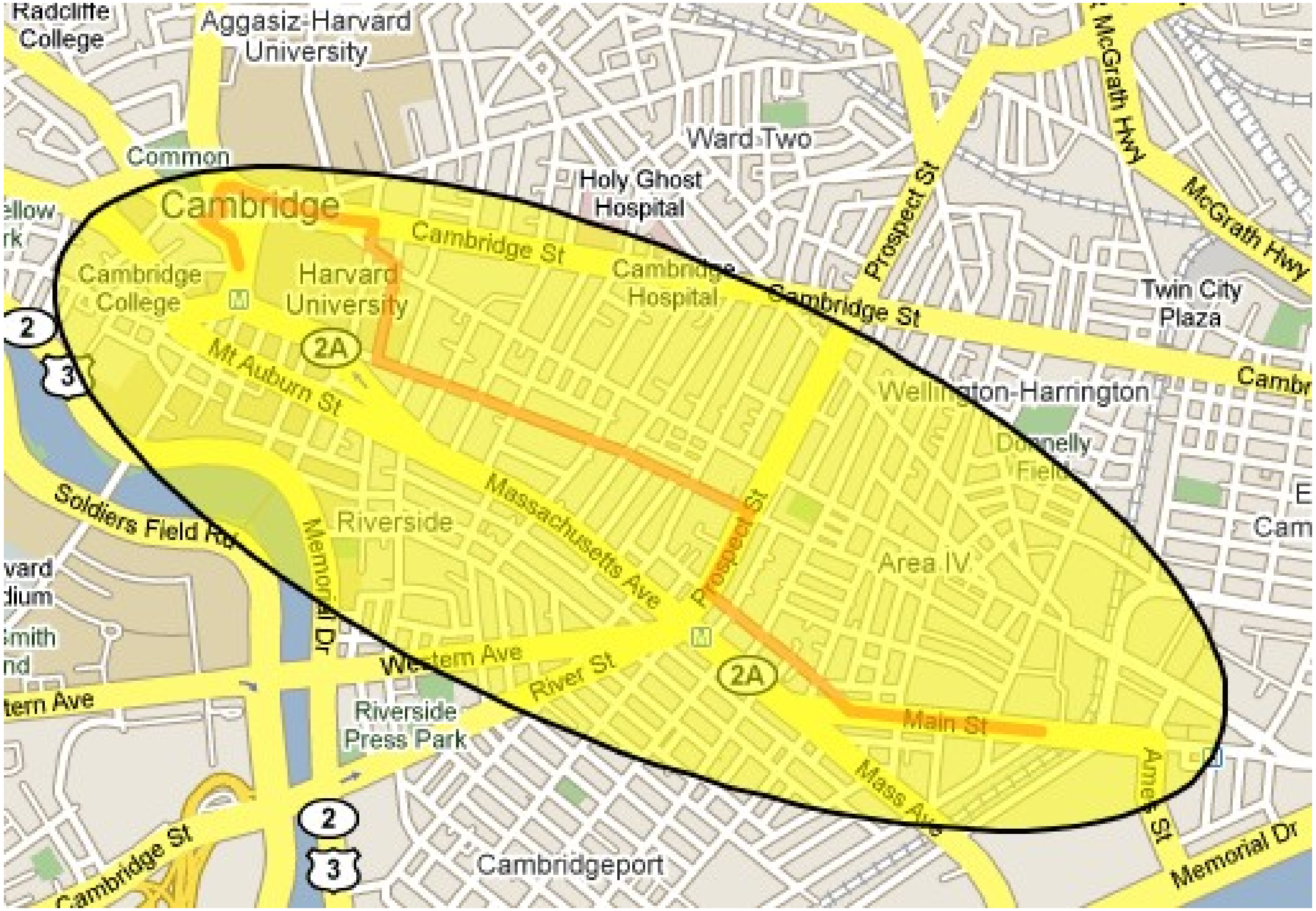
Variant of Yen's K shortest loopless paths algorithm
- Loopy paths, i.e., paths with cycle don't make sense

Delete each edge of the path found in the previous iterations with some probability
Re-do the shortest path calculation on the modified graph

Proximity-based

Compute shortest paths in a constrained region
Constraint on proximity
Shape of constrained region is ellipse
- Polygons take more space
- Ellipses better represent human behavior than rectangles

Proximity-based

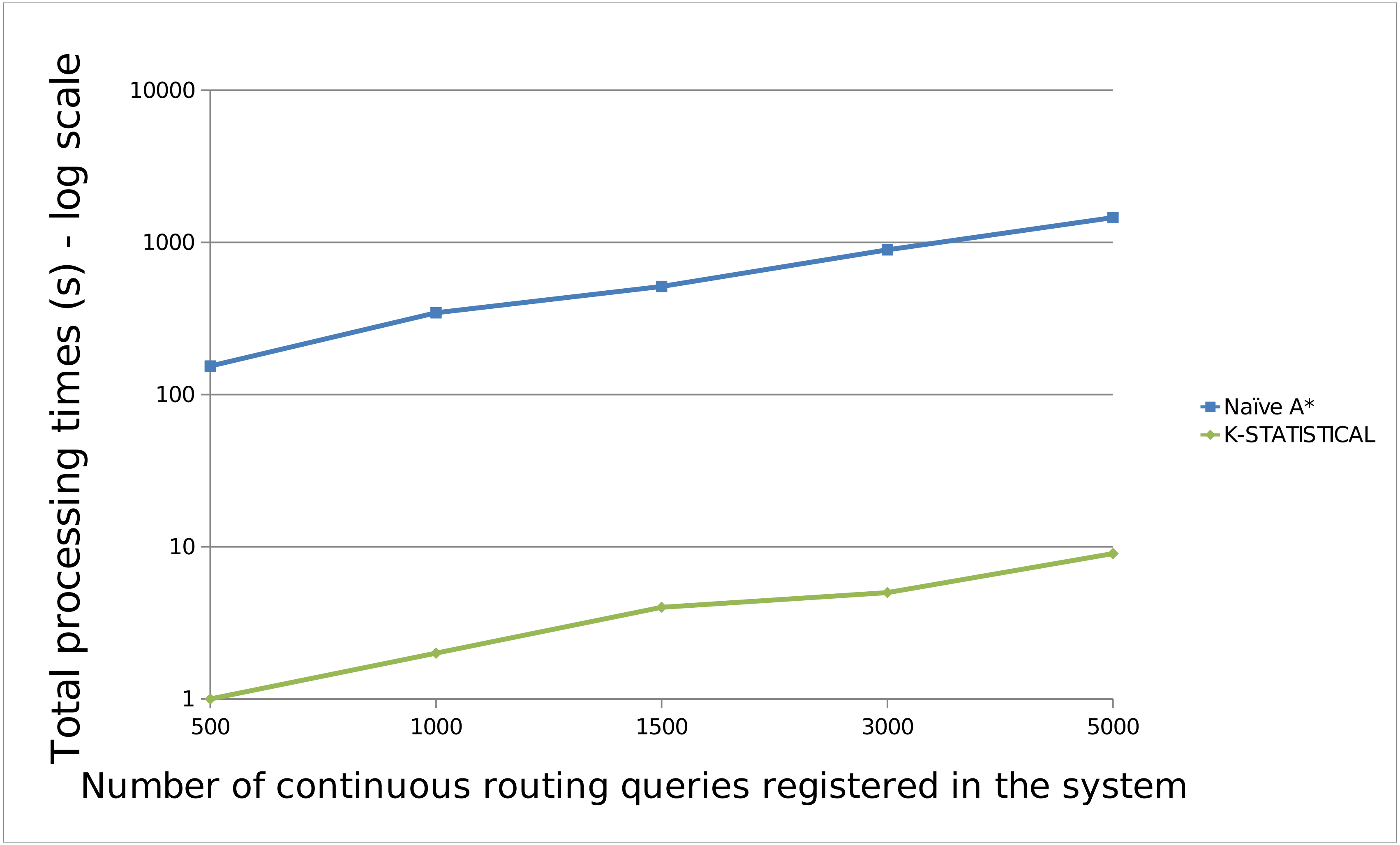


Results

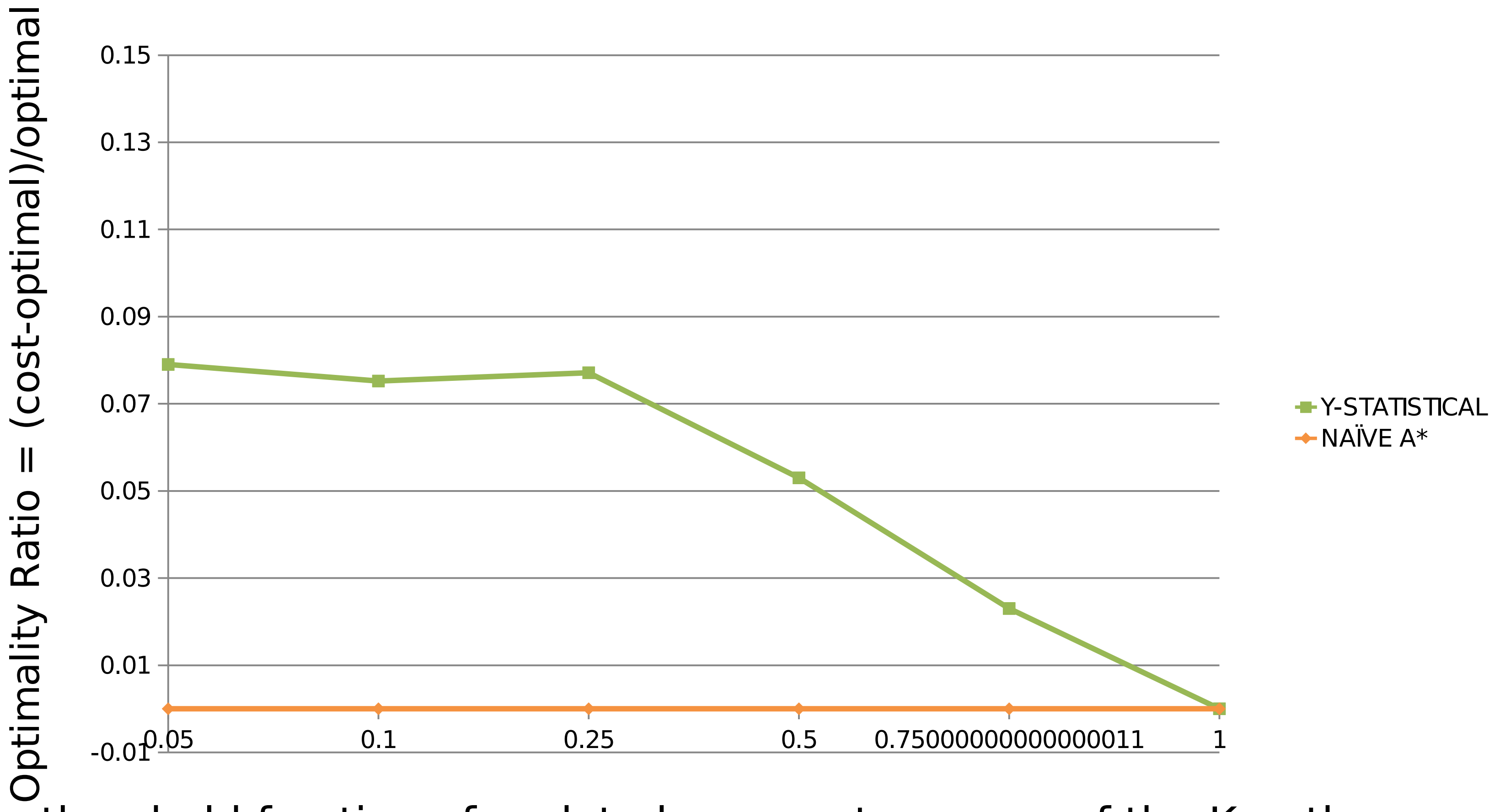
7,000 drives chosen from CarTel log containing 150,000 real drives
Replayed data from 2 months of our traffic delay database
Crucial parameters:

- ϵ: captures number of segments with delay updates
- Υ: captures degree of change in segment delay
- K: set to 5

Two orders of magnitude faster

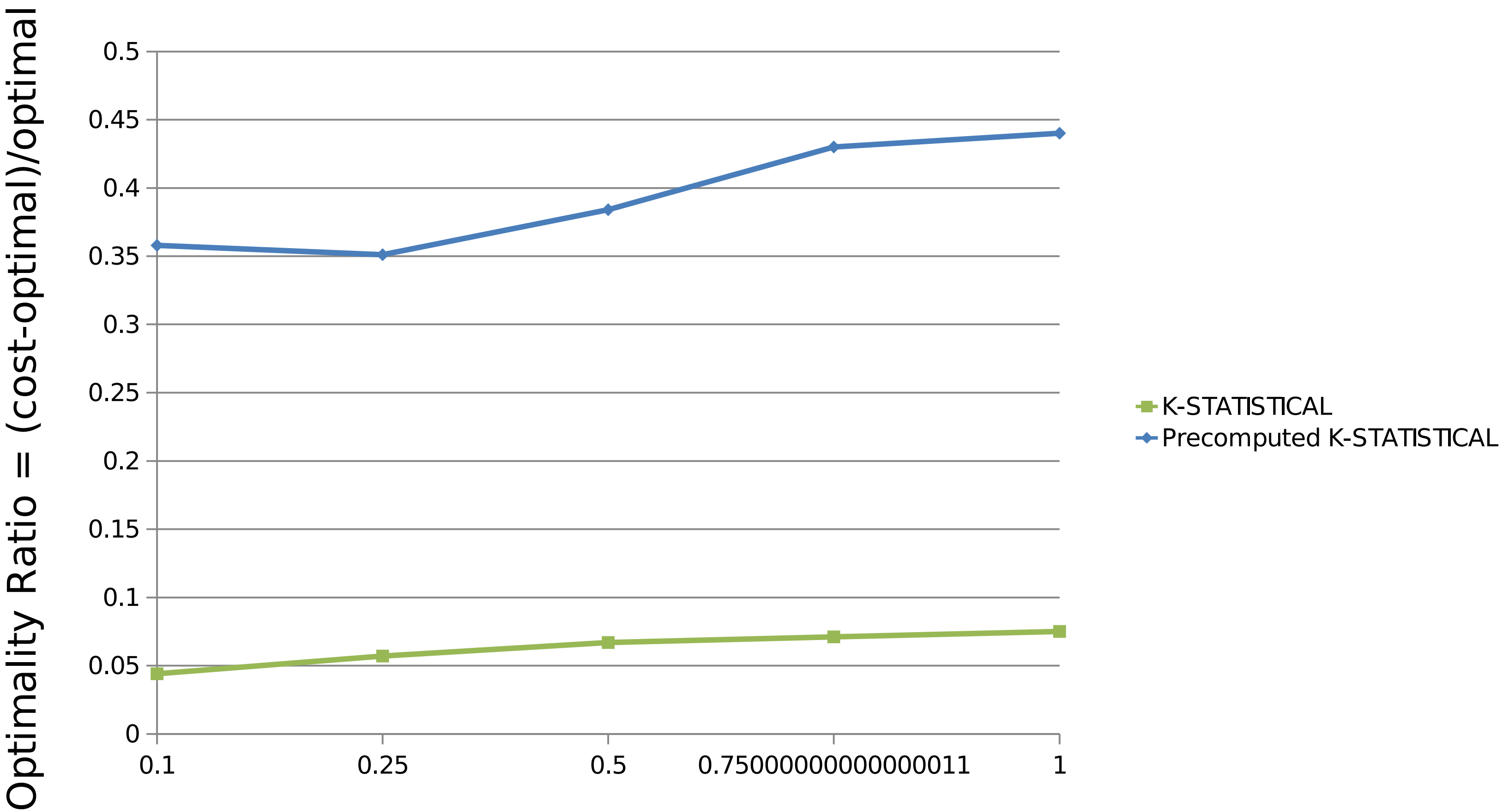


New routes are near optimal



ϵ : threshold fraction of updated segments on any of the K paths

Previously reported routes are near optimal on no update



ϵ : threshold fraction of updated segments on any of the K paths

Conclusions

- Fast and scalable continuous routing scheme which is accurate
- Routes delivered by our techniques are within 4-7% of the optimal
  - average optimality gap for pre-computed routes over 30% in all cases

Conclusions

Fast and scalable continuous routing scheme which is accurate

Routes delivered by our techniques are within 4-7% of the optimal

average optimality gap for pre-computed routes over 30% in all cases

THANK YOU