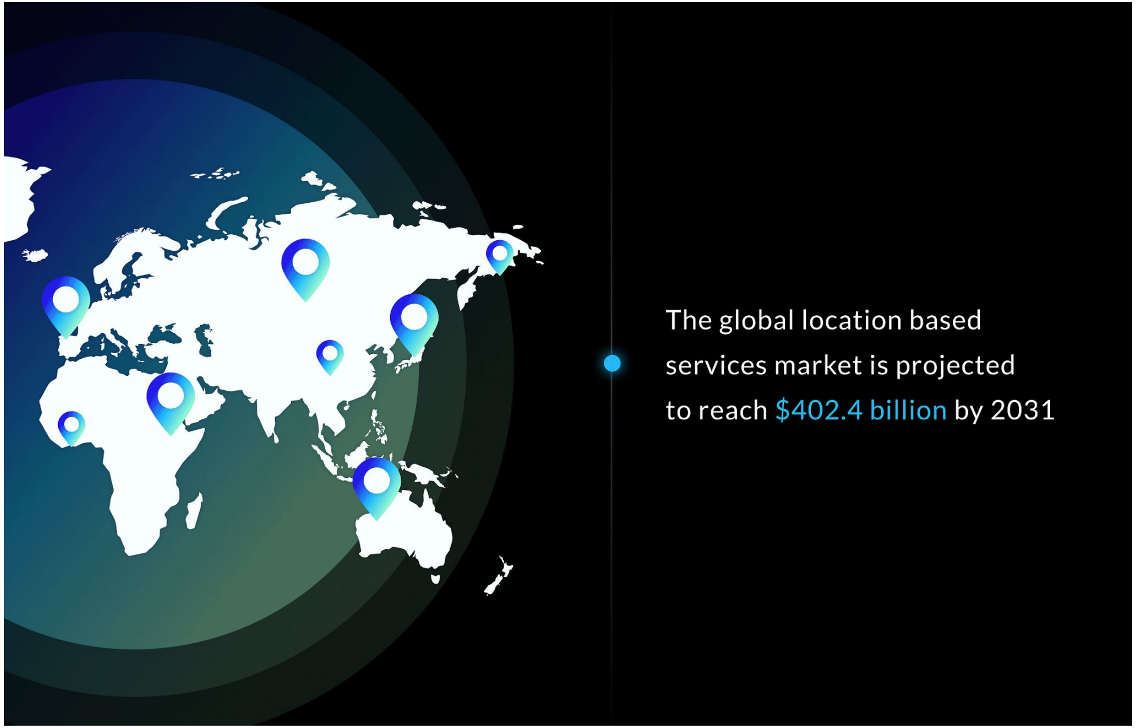
Trajectory based routing / Reachability Analysis

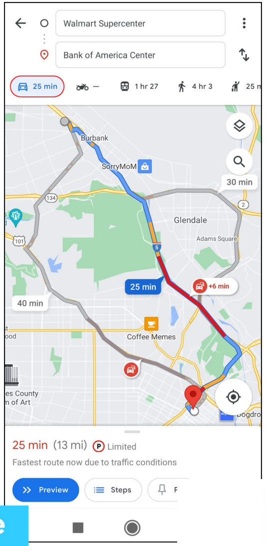
CSCI 587: Lecture 17

10/28/2024

Location-based services are everywhere



Location-based services are everywhere



waze

Azure Maps

Google Maps

WAYMO

TESLA

cruise

enterprise

Uber

kyte.

lyft

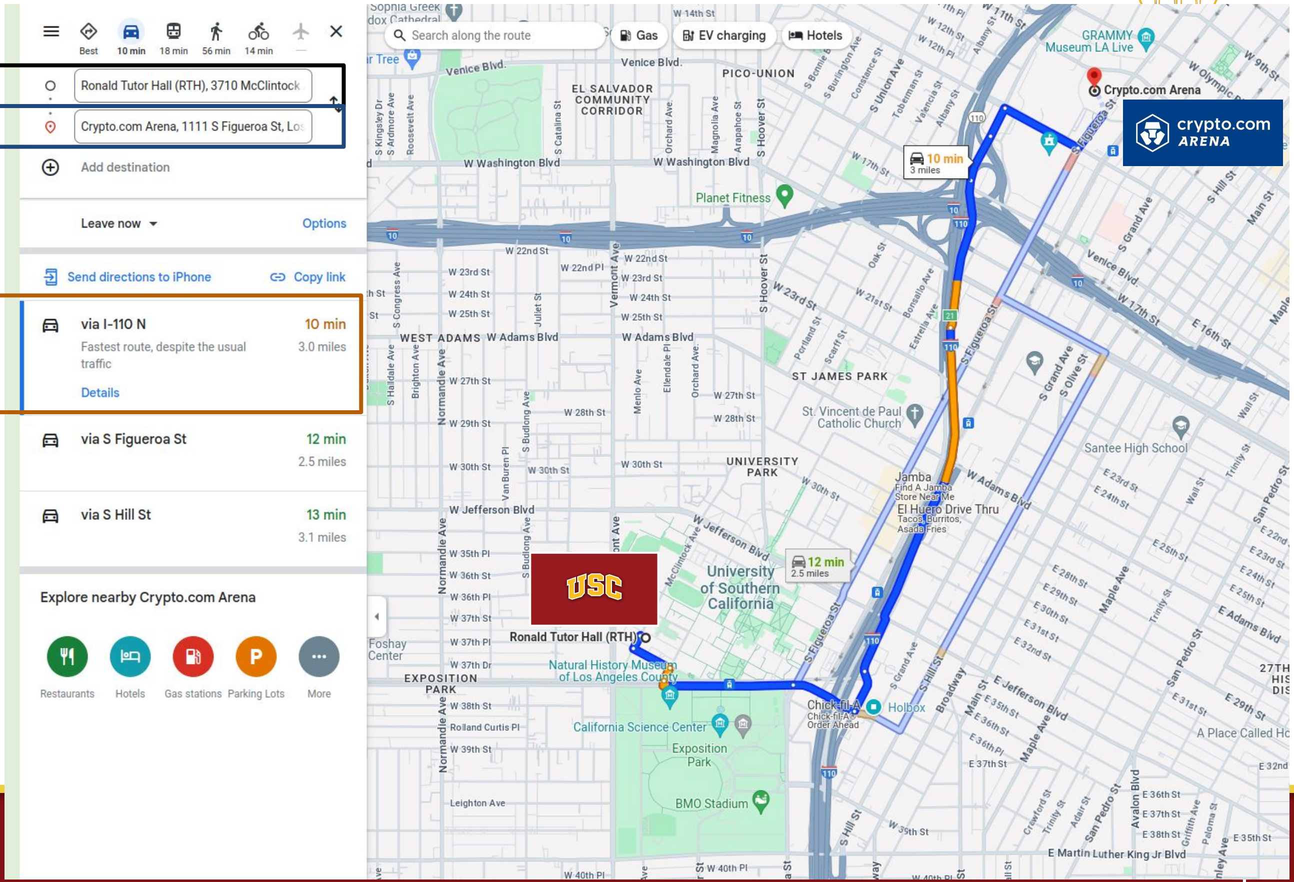
These are only a few examples…

Origin Destination Queries

Origin point

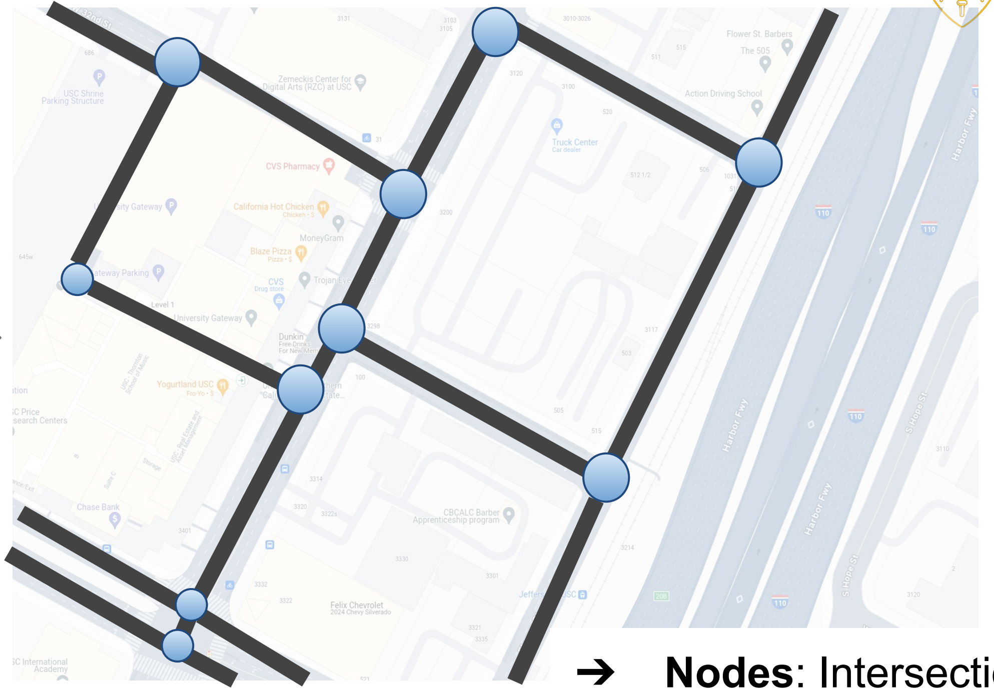
Destination point

Route & Estimated time of arrival



Road Network as a Graph

city road network extracted from OpenStreetMaps (OSM)

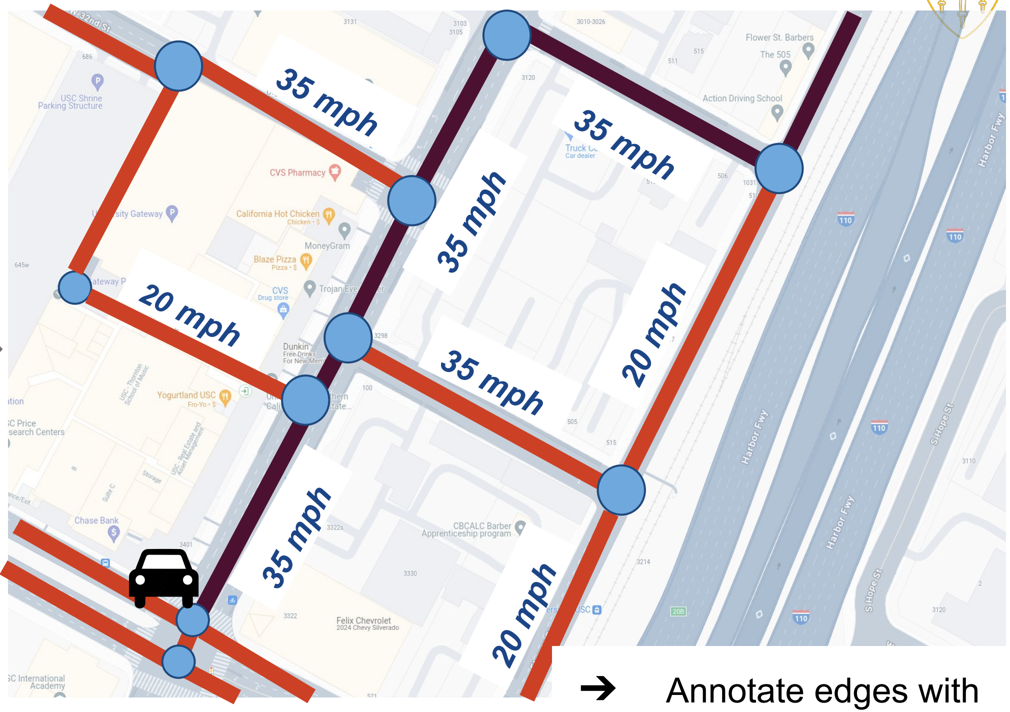


➔ **Nodes**: Intersections

➔ **Edges:** Roads

Road Network as a Graph

city road network extracted from OpenStreetMaps (OSM)

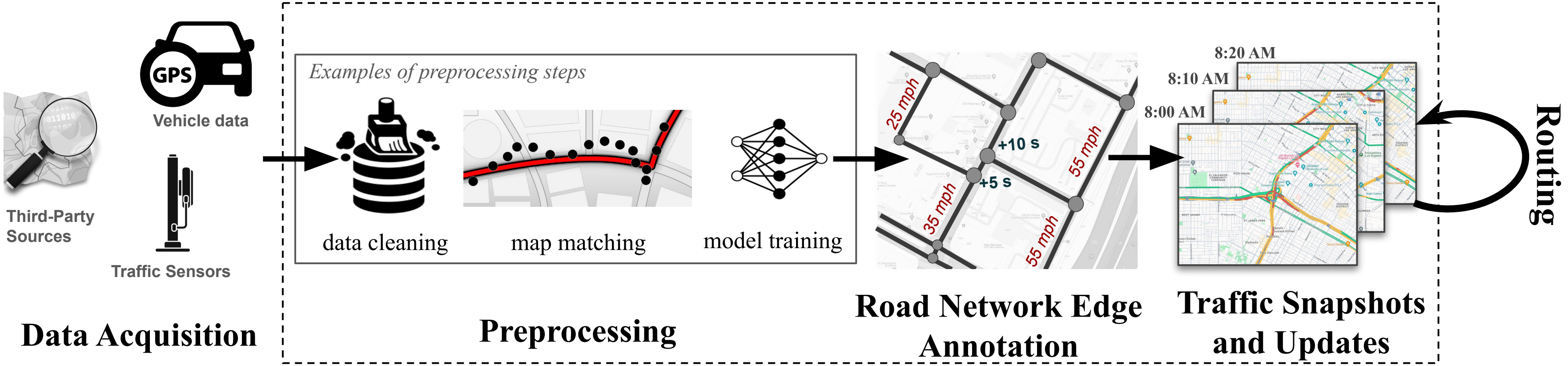


➔ Annotate edges with *weights e.g. road speed limits*

Road Network as a Graph

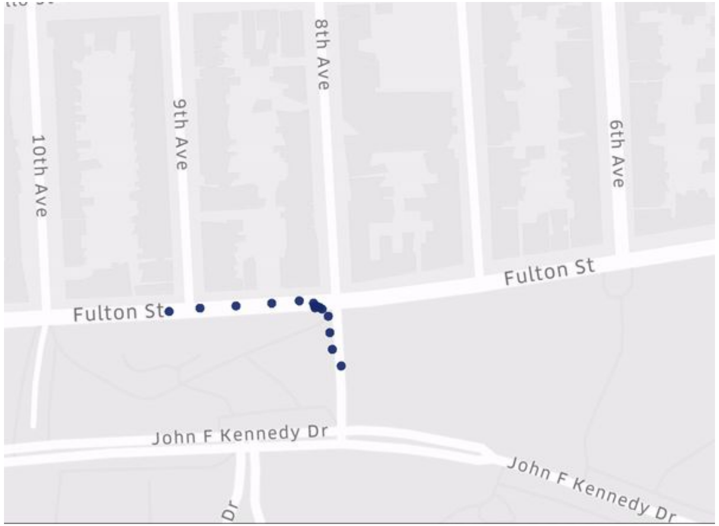
★ How can we get accurate, time dependent weights?

Typical Pipeline of Routing Services

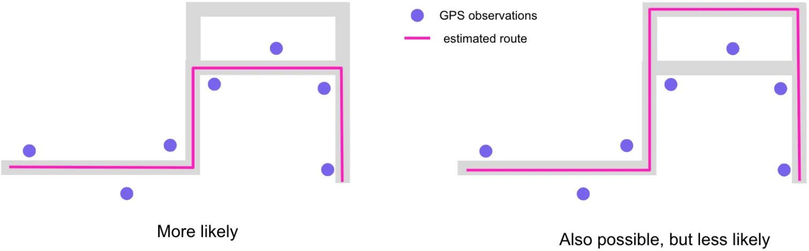


- Large scale, *up-to date GPS data* are continuously collected
- Several cost-intensive preprocessing steps to extract *time dependent "features"*
  - E.g. Map matching: GPS data is aligned with the road network
- Road network edges are dynamically updated (*e.g. every 5 minutes*) and new *traffic snapshots* are created

Map Matching



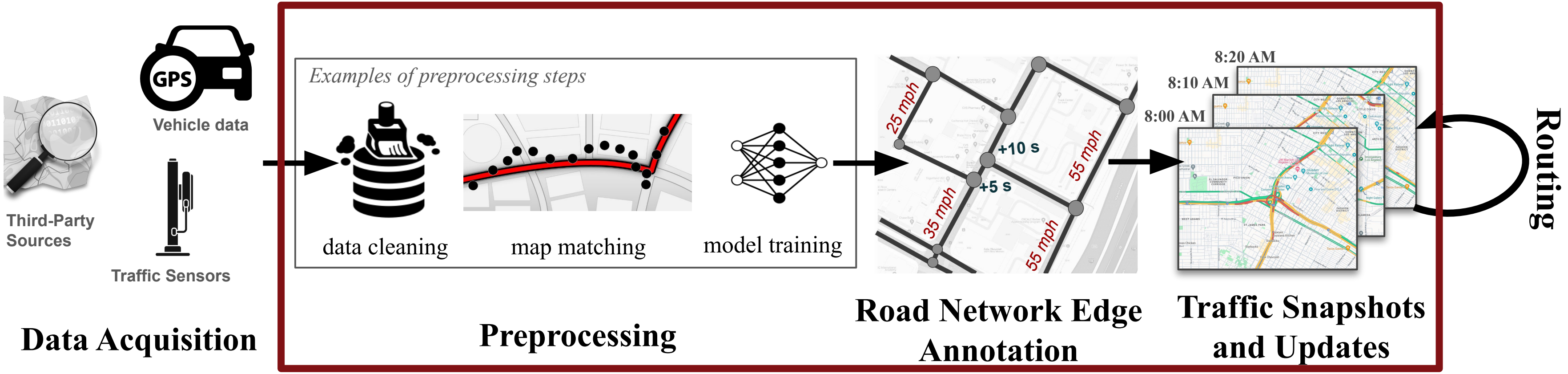
Example of a driver trajectory



- Lyft and Uber use map matching to:
  - To compute the distance travelled by a driver to calculate the fare
  - Dispatch decisions and to display the drivers' cars on the rider app
  - Detect reckless driving

- Approaches for Map Matching
  - Hidden Markov Model: *Newson & Krumm @ SIGSPATIAL '09* [1]
    - DiDi's IJCAI-19 Tutorial [2]
    - Map Matching @ Uber [3]

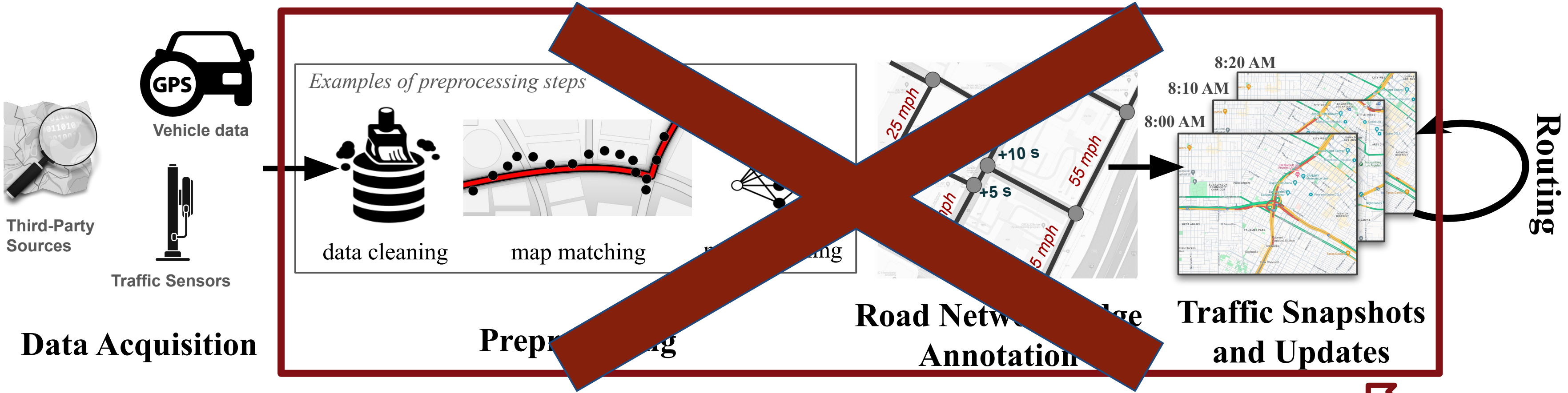
Typical Pipeline of Routing Services



Repeats as new data becomes available

- Large scale, *up-to date GPS data* are continuously collected
- Several cost-intensive preprocessing steps to extract *time dependent "features"*
  - E.g. Map matching: GPS data is aligned with the road network
- Road network edges are dynamically updated (*e.g. every 5 minutes*) and new *traffic snapshots* are created

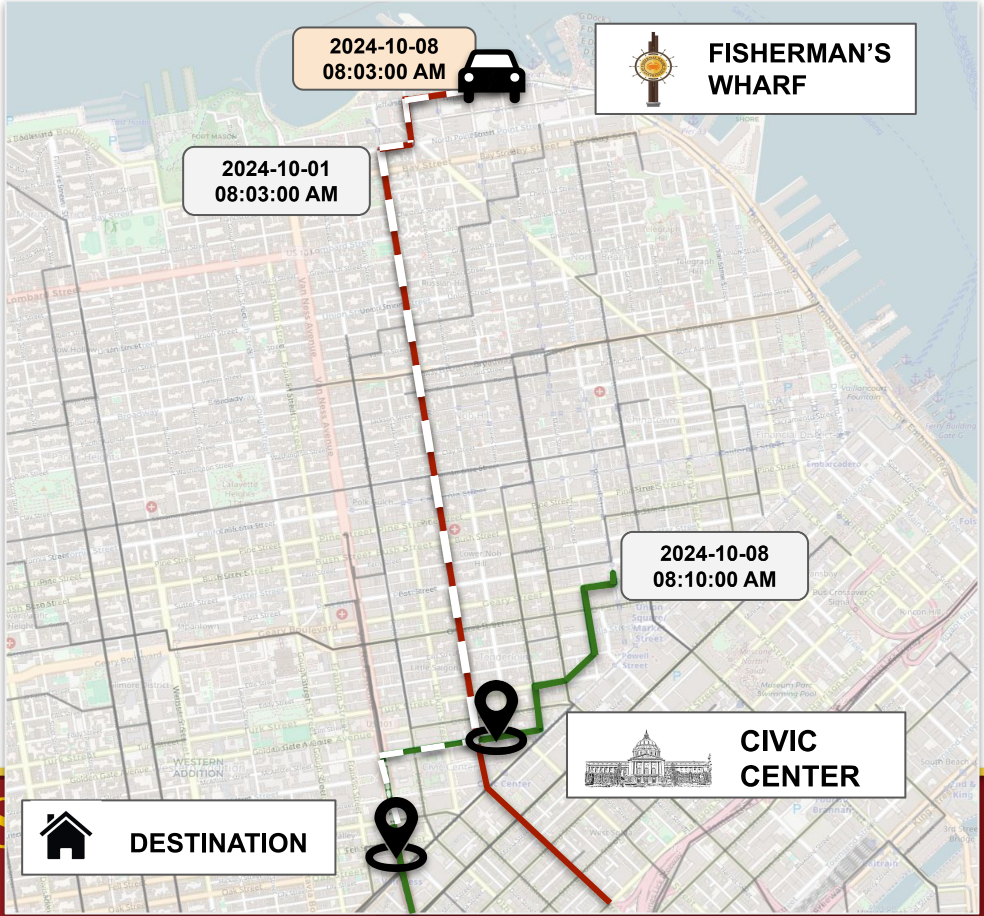
Typical Pipeline of Routing Services



Repeats as new data becomes available

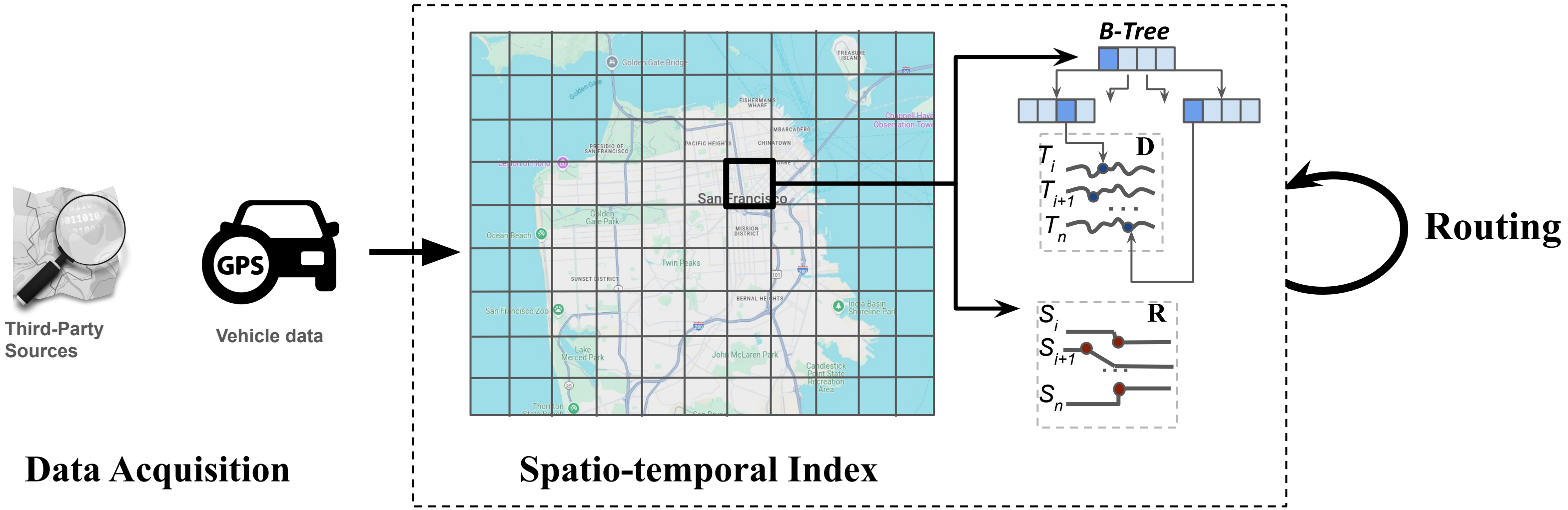
- Large scale, *up-to date GPS data* are continuously collected
- Several cost-intensive preprocessing steps to extract *time dependent "features"*
  - E.g. Map matching: GPS data is aligned with the road network
- Road network edges are dynamically updated (*e.g. every 5 minutes*) and new *traffic snapshots* are created

TrajRoute *(Motivation)*



OR: Fisherman's Wharf
DEST: Home
Time: 08:03:00 AM

TrajRoute: Approach



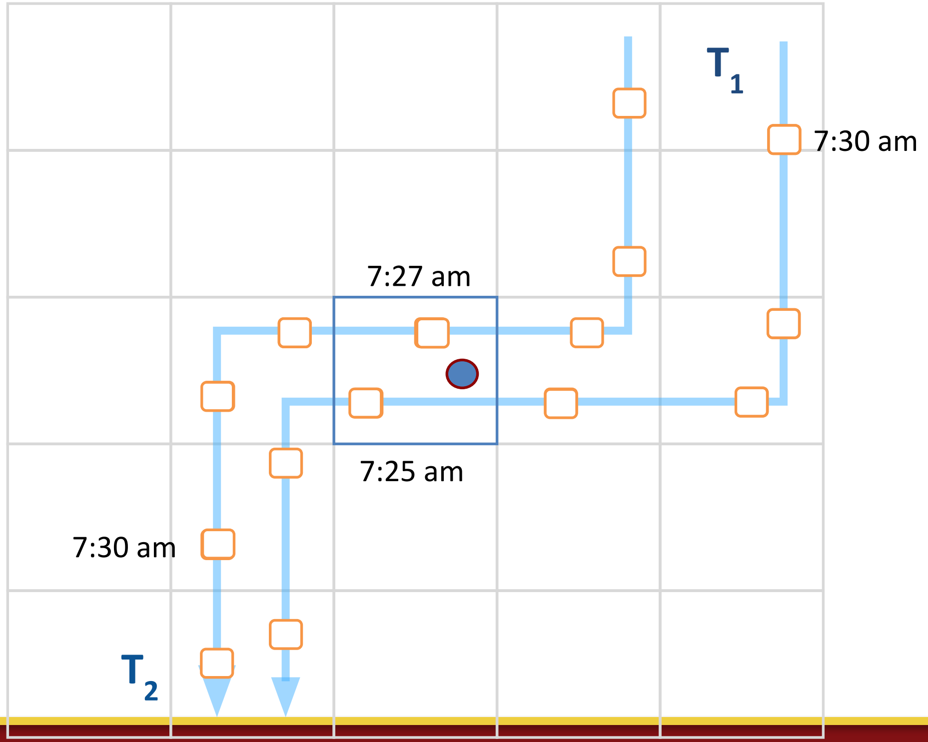
- Routing based on *raw historical trajectories*
    - Ensure that only trajectories that are *spatially* and *temporarily* close to current position are considered
- *Fallback* to the road network when trajectories are not available

TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

Query Point: <p_{OR} , p_{DEST} , dtime>

Current Position(, 7:30am)

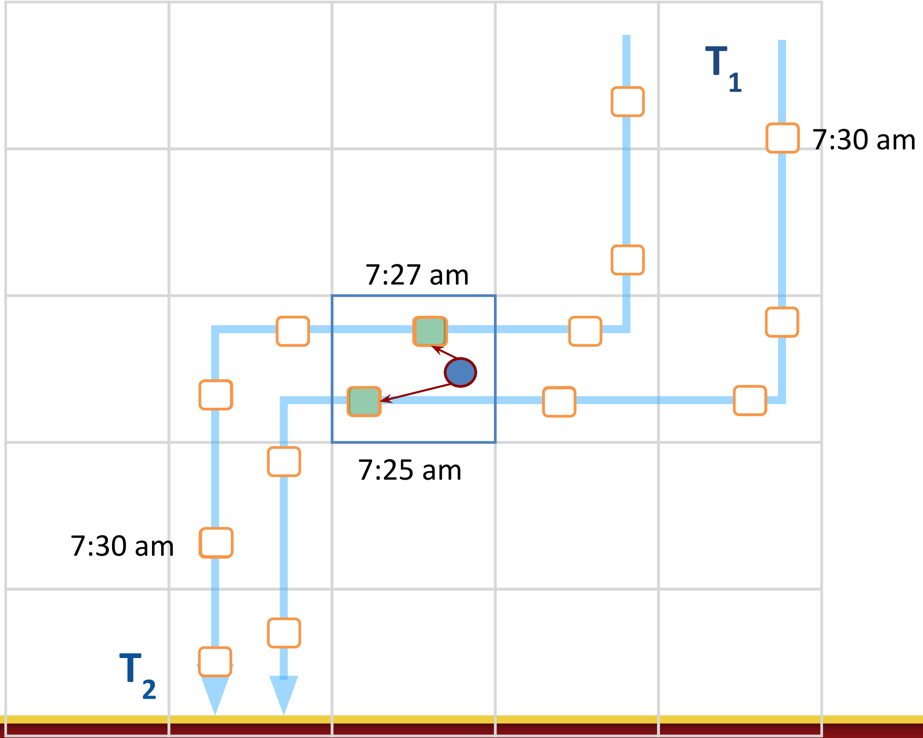


TrajRoute: Approach

GPS Point: <p_J = (lat_J, lon_J), ts_J>

Query Point: <p_{OR} , p_{DEST} , dtime>

Current Position(, 7:30am)



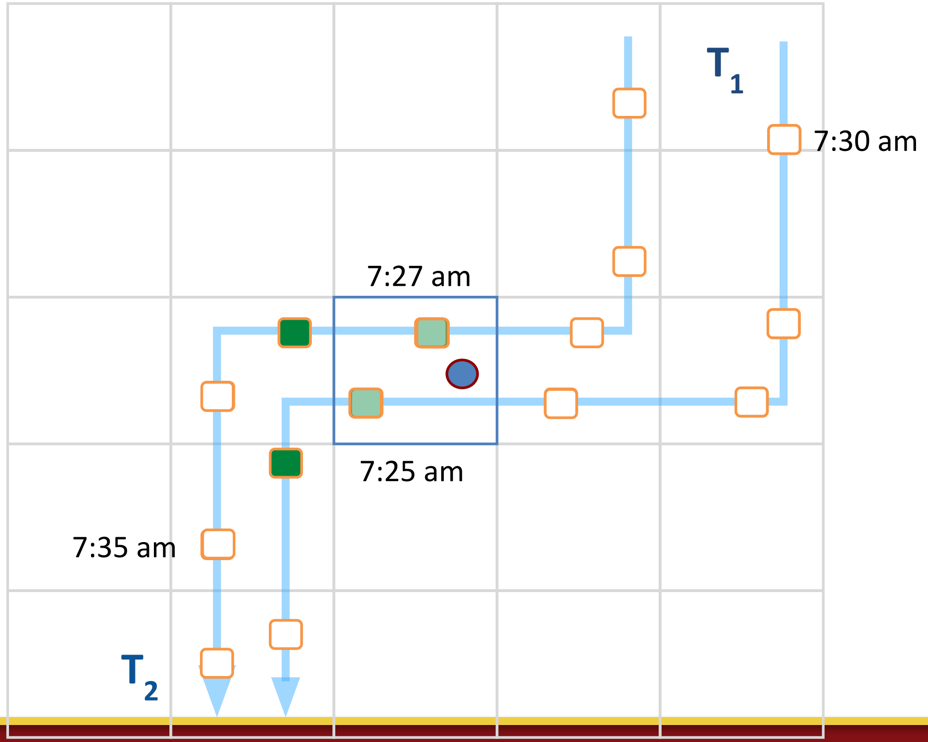
TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

Query Point: <p_{OR} , p_{DEST} , dtime>

Trajectory neighbors to ●

Current Position(● , 7:30am)

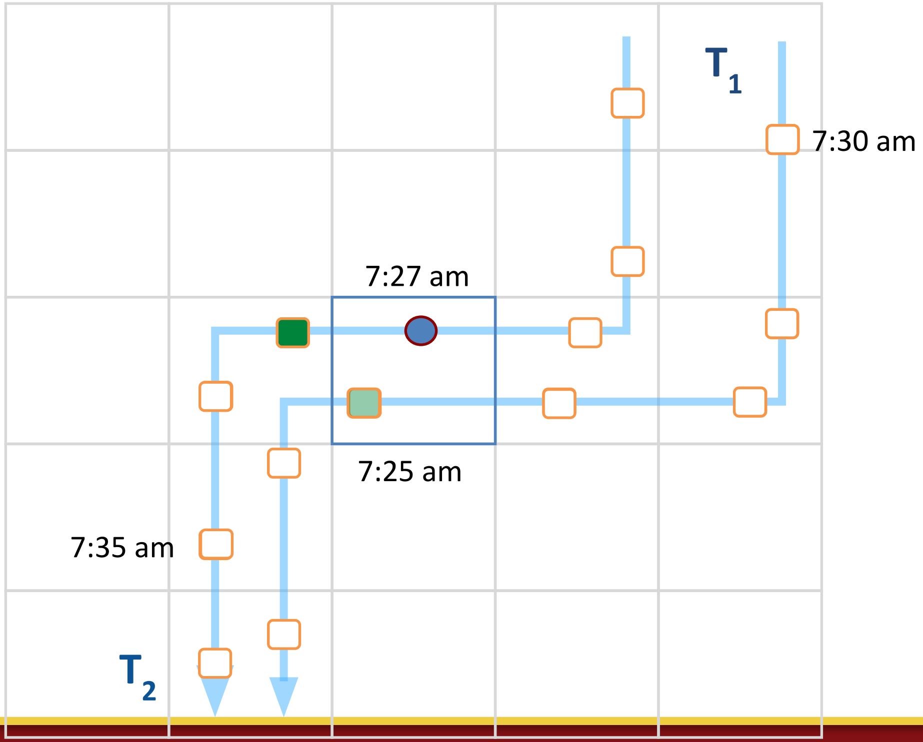


TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

Query Point: <p_{OR} , p_{DEST} , dtime>

Current Position(, 7:30am)



TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

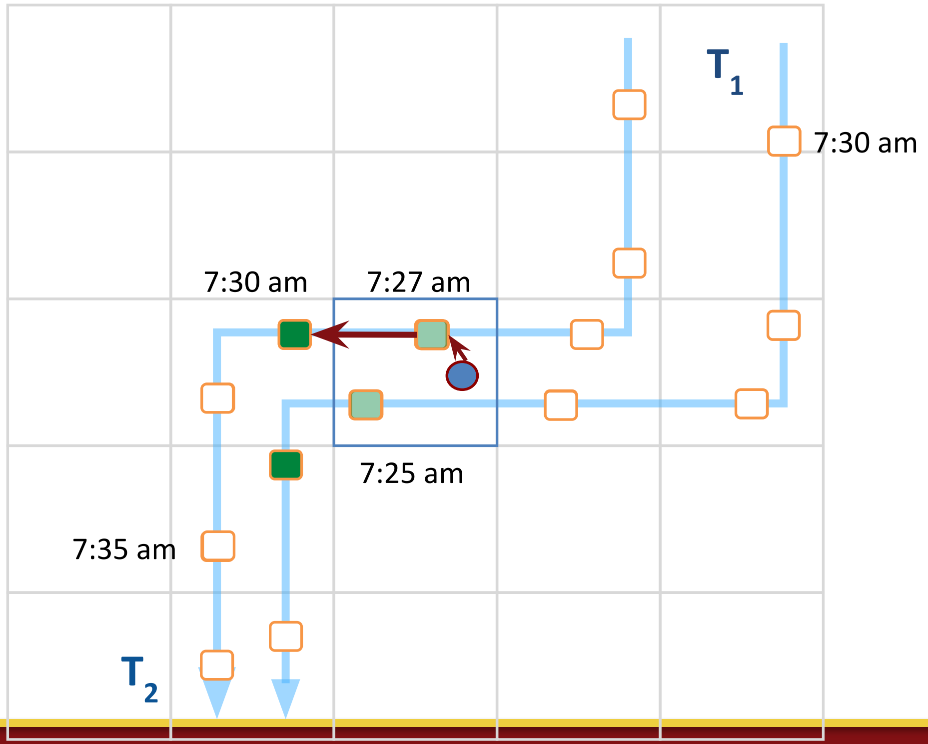
Query Point: <p_{OR}, p_{DEST}, dtime>

Trajectory neighbors to

C_{traj} (,) = Tc + ts( ) - ts( )

- **Tc:** Cost of transition, constant, depends on the size of the cell

Current Position(, 7:30am)

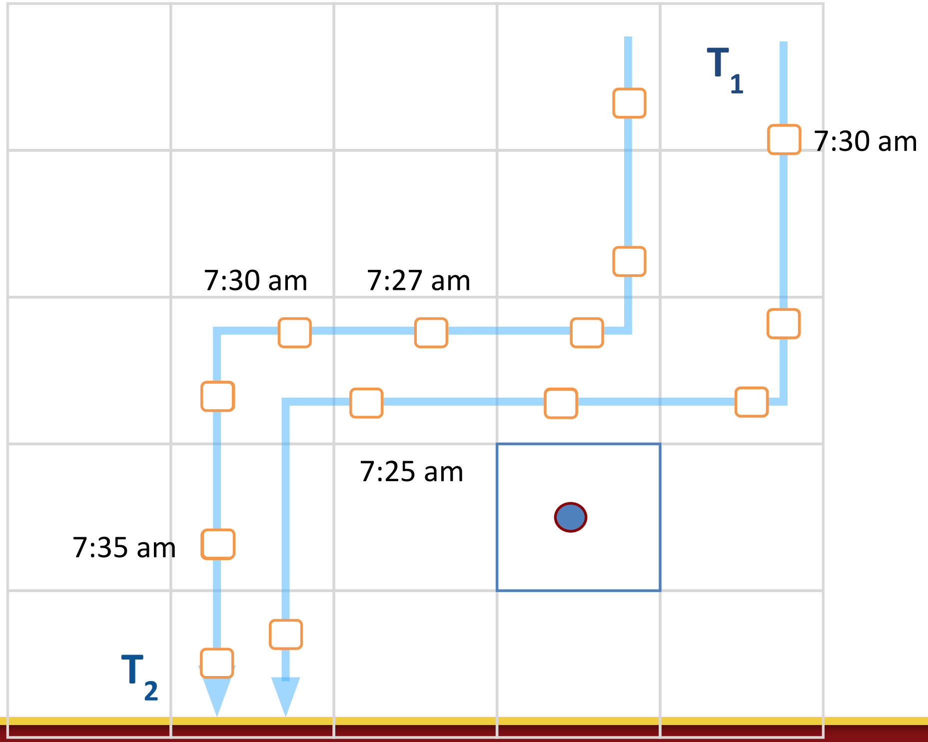


TrajRoute: Approach

GPS Point: <p_J = (lat_J, lon_J), ts_J>

Query Point: <p_{OR} , p_{DEST} , dtime>

Current Position(, 7:30am)



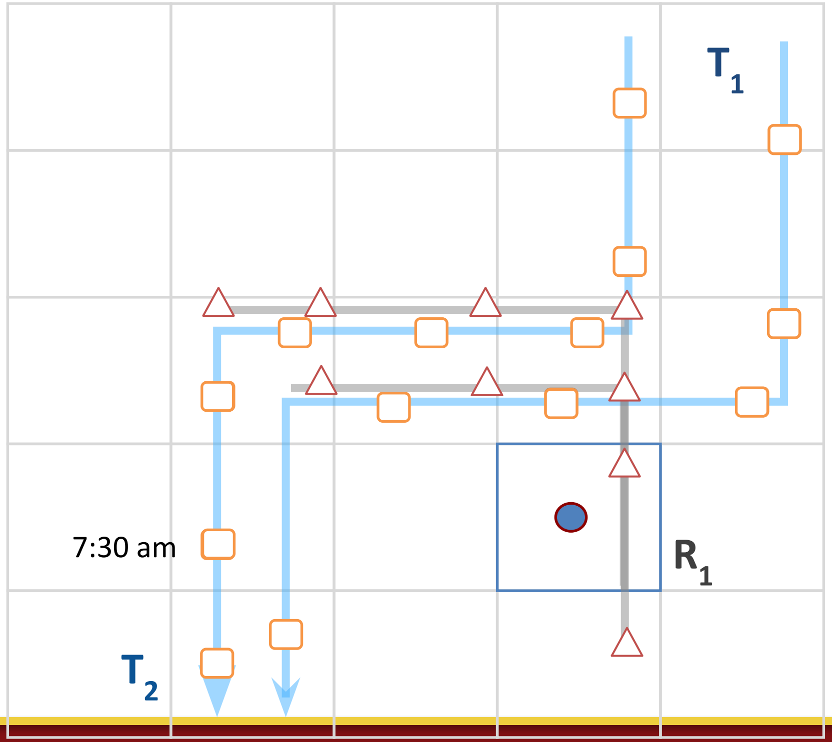
TrajRoute: Approach

□ GPS Point: <p_J = (lat_J, lon_J), ts_J>

△ Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

Current Position(● , 7:30am)



TrajRoute: Approach

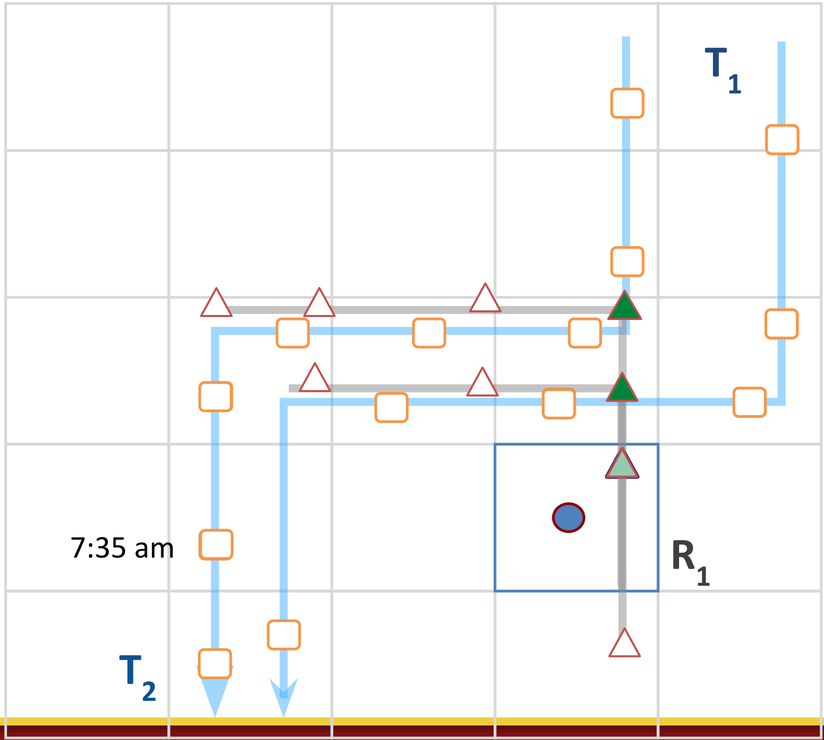
GPS Point: <p_J = (lat_J, lon_J), ts_J>

Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

Road neighbors to

Current Position(, 7:30am)



TrajRoute: Approach

□ GPS Point: <p$_J$ = (lat$_J$, lon$_J$), ts$_J$>

△ Road Point: <p$_I$ = (lat$_I$, lon$_I$)>

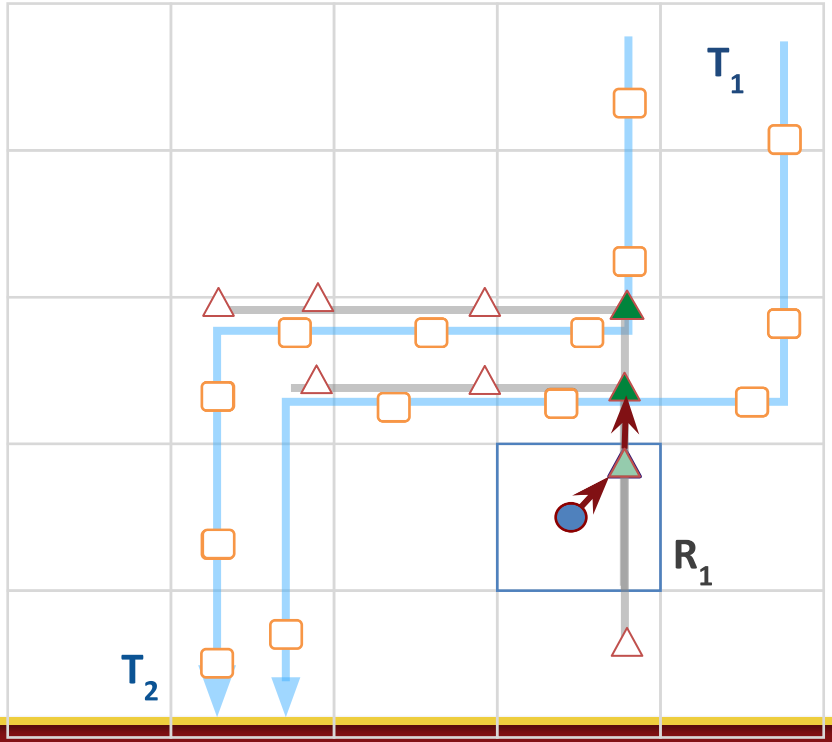
Query Point: <p$_{OR}$, p$_{DEST}$, dtime>

▲ Road neighbors to ●

C_{road} (● , ▲) = τc + [dist(△ , ▲) / v (▲)]

- dist: Haversine distance between road points
- v: Speed limit of road segment

Current Position(● , 7:30am)



TrajRoute: Approach

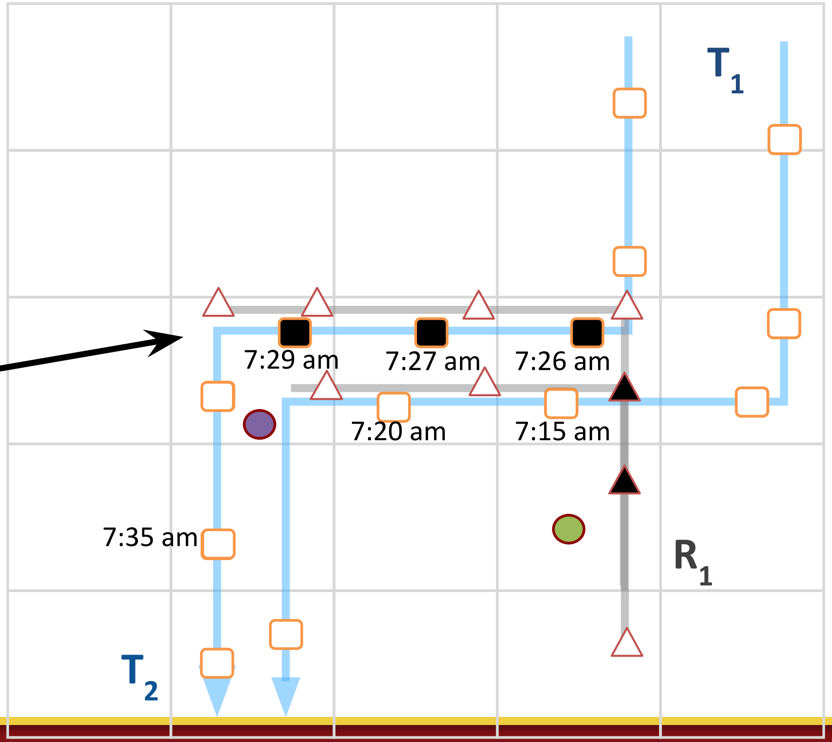
GPS Point: <p_J = (lat_J, lon_J), ts_J>

Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

Ideal Path



TrajRoute: Approach

□ GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

△ Road Point: <$p_I = (lat_I, lon_I)$>

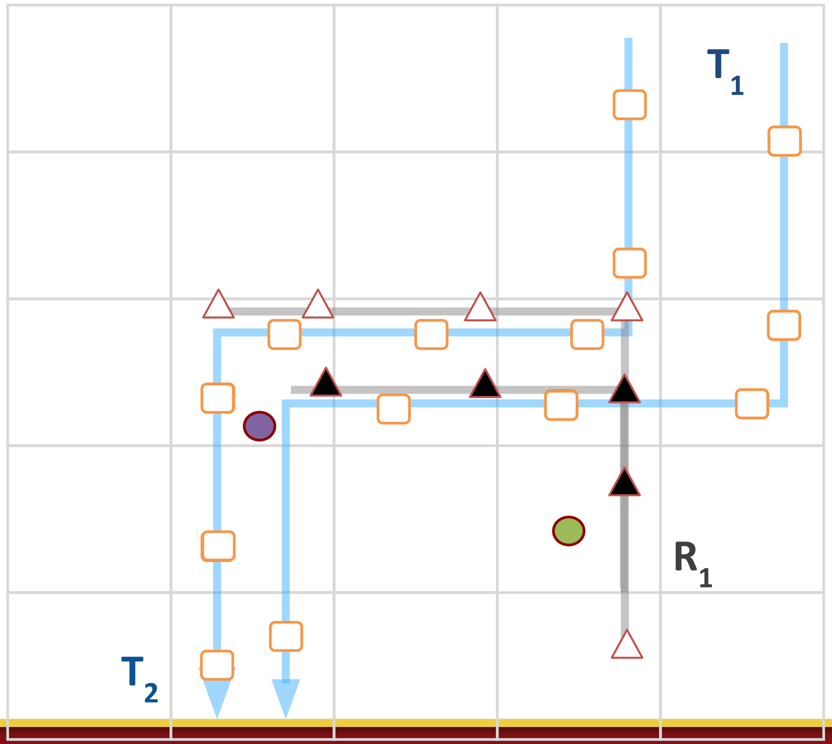
Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

C_{road} is always less than C_{traj}. Does not account for:

- Intersection costs
- Acceleration/Deceleration
- Traffic Lights
- Traffic Congestion etc.

Inherently encoded in trajectory timestamps



TrajRoute: Approach

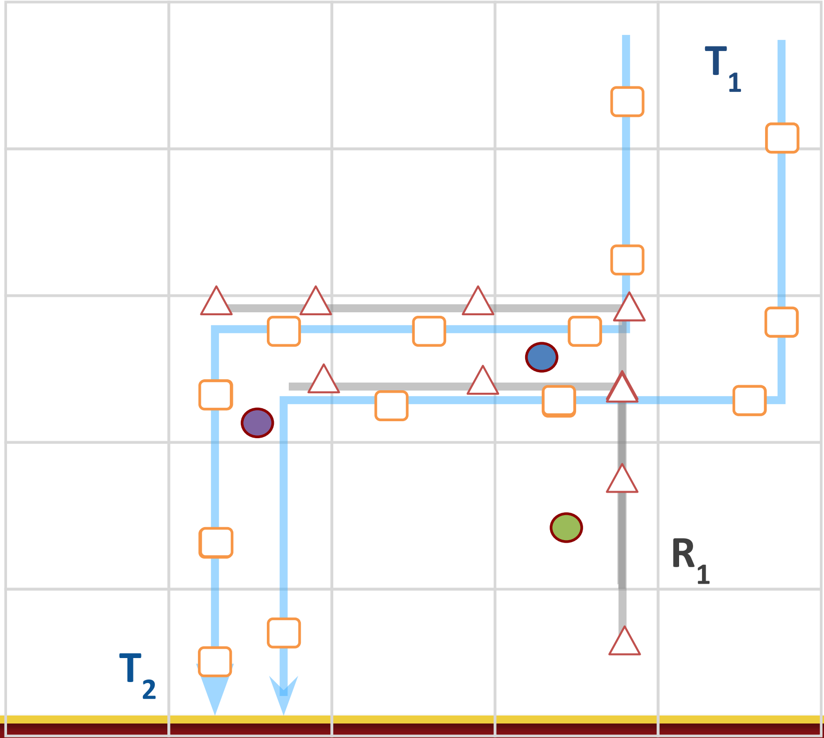
□ GPS Point: <p_J = (lat_J, lon_J), ts_J>

△ Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

Current Position(● , 7:30am)



TrajRoute: Approach

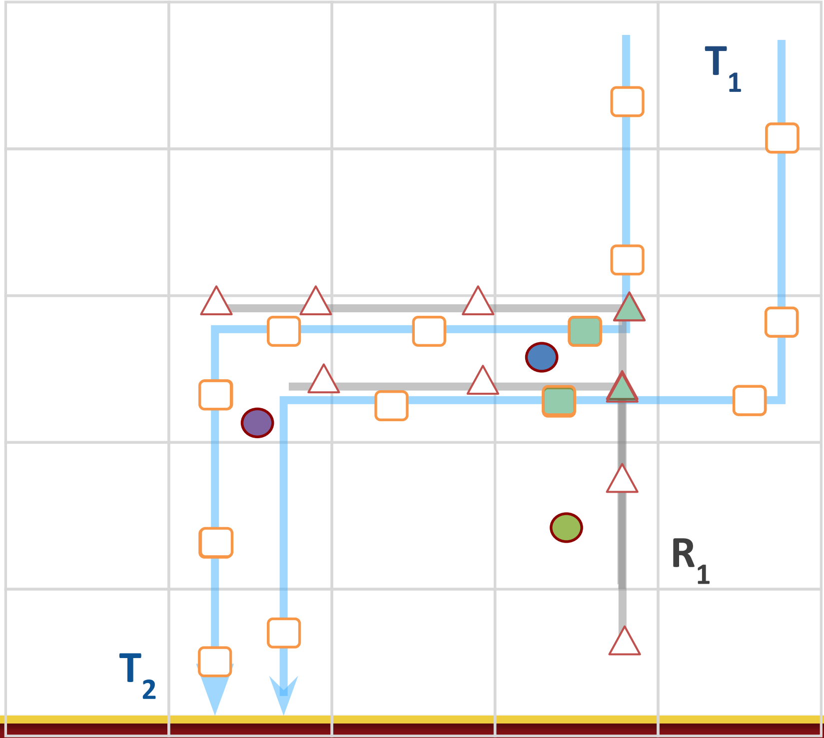
GPS Point: <p_J = (lat_J, lon_J), ts_J>

Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

Current Position(, 7:30am)



TrajRoute: Approach

GPS Point: <p_J = (lat_J, lon_J), ts_J>

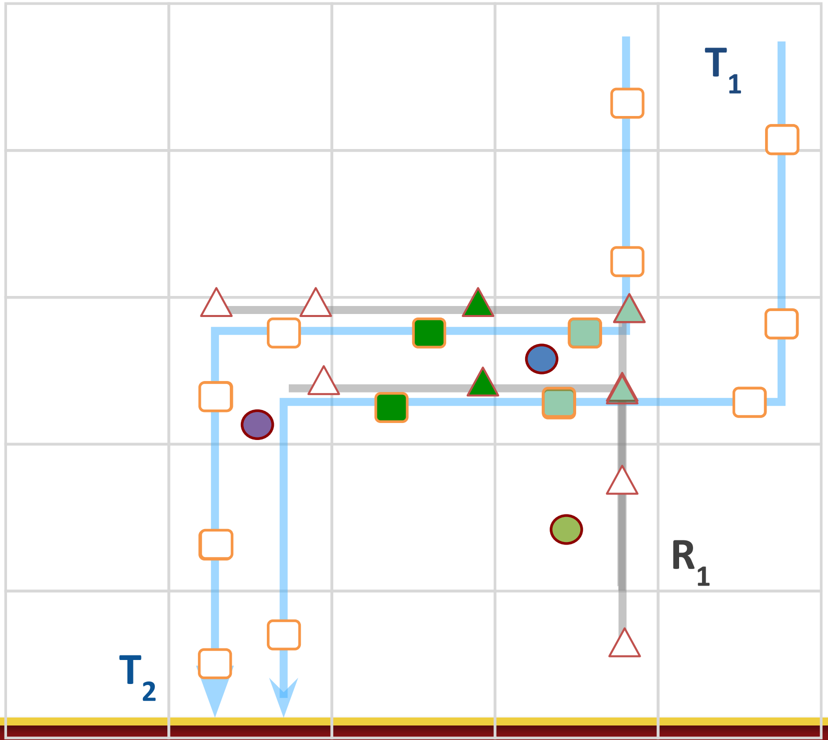
Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

C_{base} = { C_{road} (,), : Road Neighbor ; C_{traj} (,), : Traj. Neighbor }

Current Position(, 7:30am)



TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

Road Point: <$p_I = (lat_I, lon_I)$>

Query Point: <p_{OR} , p_{DEST} , dtime>

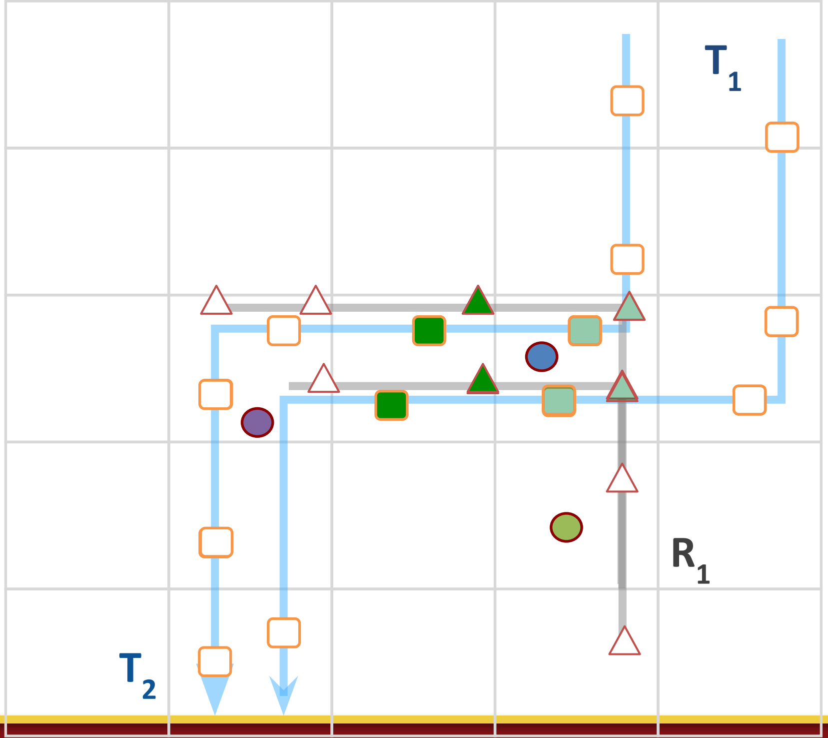
7:30 am

$C_{pref} =$ (1+ α) C_{road} (,), : Road Neighbor

C_{traj} (,), : Traj. Neighbor

- α: road penalty factor (> 0)

Current Position(, 7:30am)



TrajRoute: Approach

GPS Point: $\langle p_J = (lat_J, lon_J), ts_J \rangle$

Road Point: $\langle p_I = (lat_I, lon_I) \rangle$

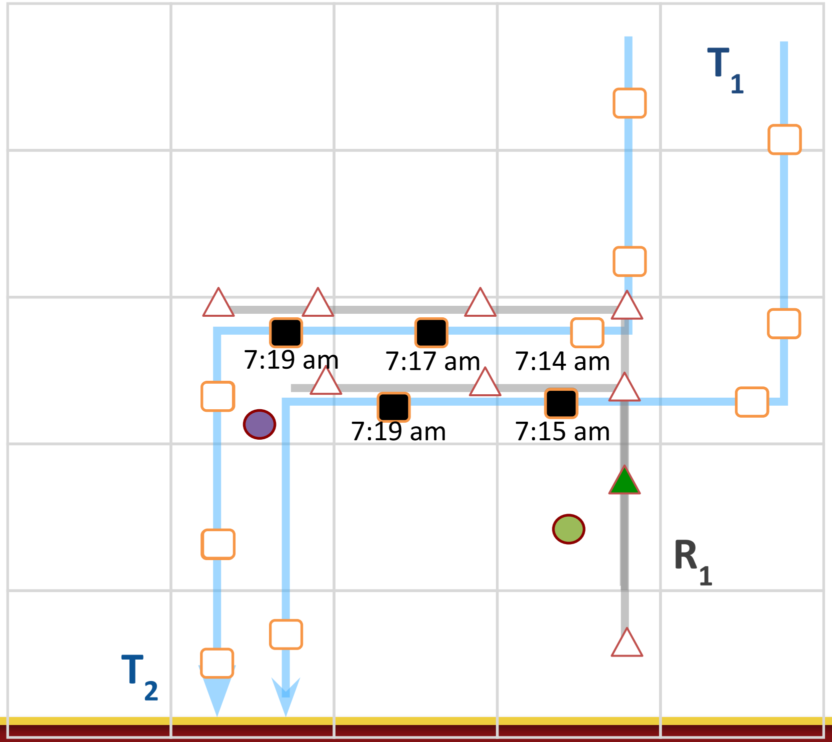
Query Point: $\langle p_{OR}, p_{DEST}, dtime \rangle$

7:30 am

$$C_{pref} = \begin{cases} (1+\alpha)\ C_{road}(\bullet, \blacktriangle), \blacktriangle : \text{Road Neighbor} \\ C_{traj}(\bullet, \blacksquare), \blacksquare : \text{Traj. Neighbor} \end{cases}$$

- α: road penalty factor (> 0)

Current Position(● , 7:30am)



TrajRoute: Approach

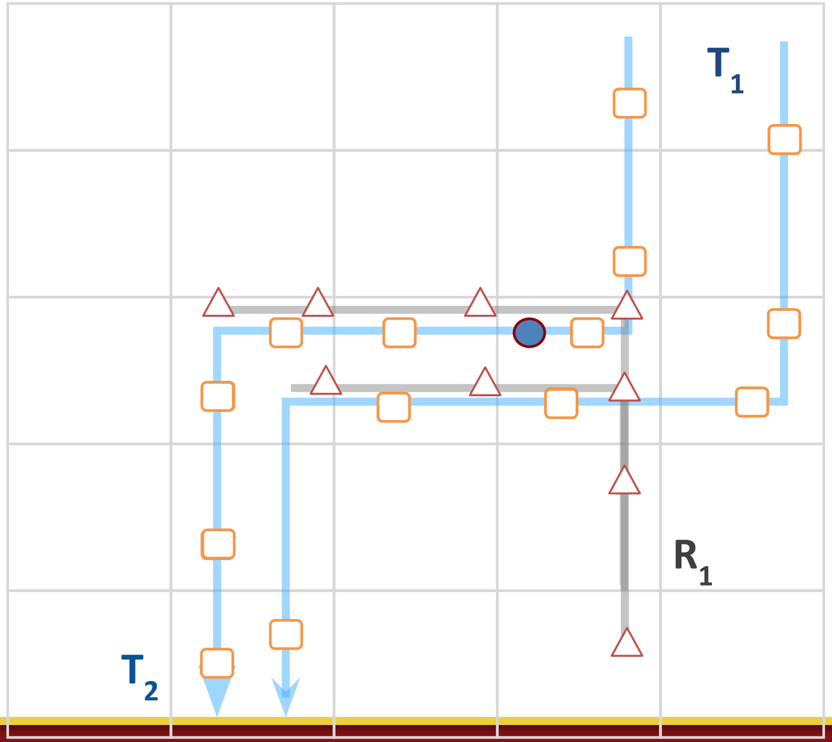
GPS Point: <p_J = (lat_J, lon_J), ts_J>

Road Point: <p_I = (lat_I, lon_I)>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

Current Position(, 7:30am)



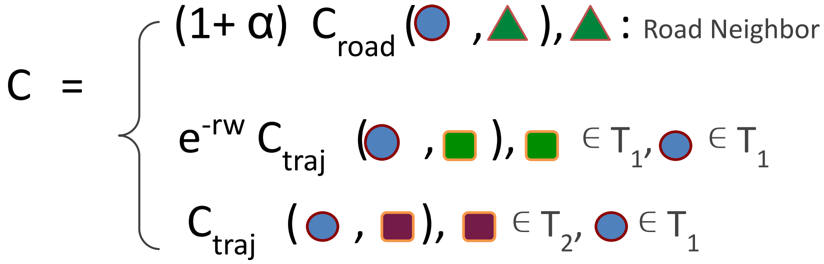
TrajRoute: Approach

GPS Point: <p_J = (lat_J, lon_J), ts_J>

Road Point: <p_I = (lat_I, lon_I)>

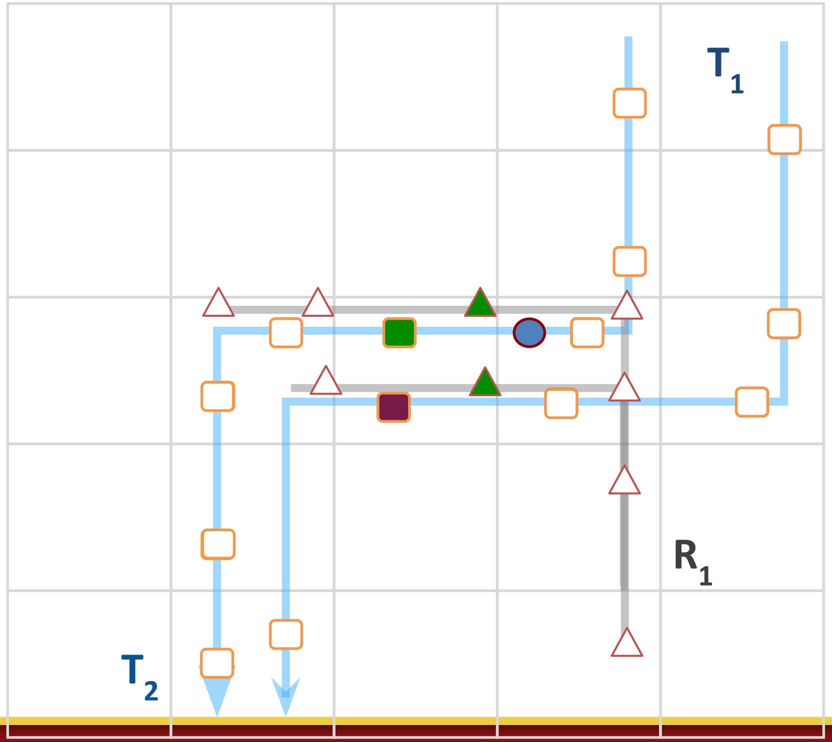
Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am



- α: road penalty factor (> 0)
- rw: continuity reward ∈ [0, 1]

Current Position(, 7:30am)



TrajRoute: Approach

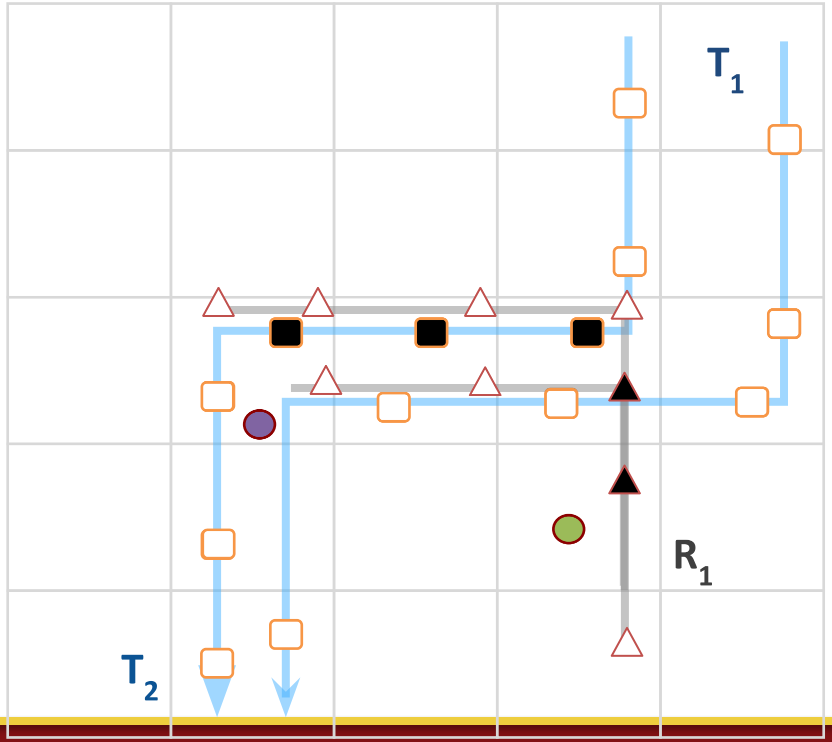
GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

Road Point: <$p_I = (lat_I, lon_I)$>

Query Point: <p_{OR} , p_{DEST} , dtime>

7:30 am

Current Position(, 7:30am)



TrajRoute: Approach

GPS Point: <$p_J = (lat_J, lon_J), ts_J$>

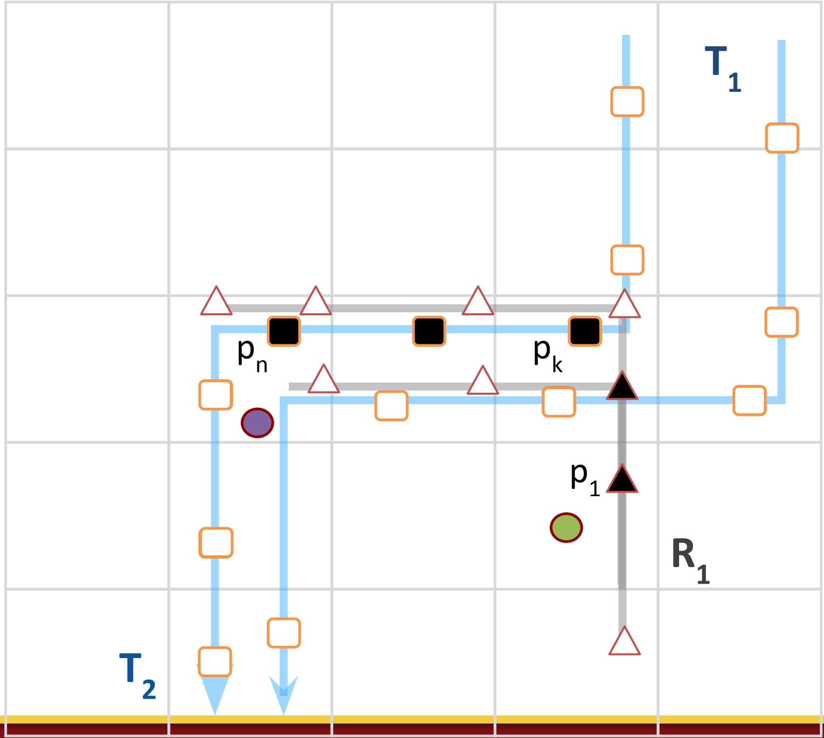
Road Point: <$p_I = (lat_I, lon_I)$>

Query Point: <p_{OR}, p_{DEST}, dtime>

7:30 am

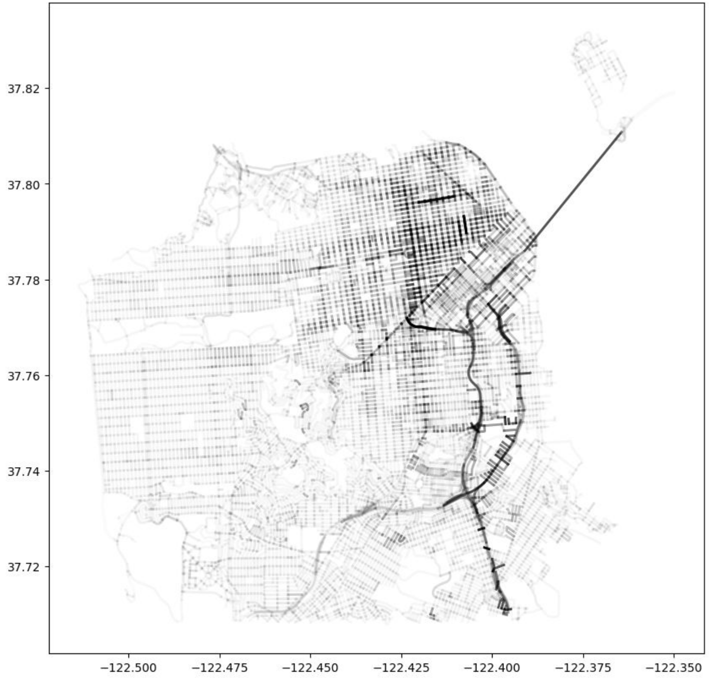
- Any pathfinding algorithm can be applied.
- For Dijkstra:
  - $$g(p_k) = \sum_{i=1}^{|P|} C(p_{i-1}, p_i), \quad p_i \in P$$
- For A*:
  - $$h(p_k) = \frac{dist(p_k, Q.p_{DEST})}{v_{max}}$$ ➡ always underestimates the cost

Current Position(, 7:30am)



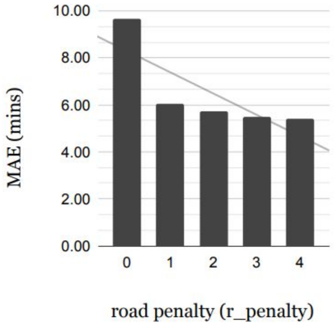
TrajRoute: Experiments

- Data Source
  - San Francisco Taxi Data
  - OSM for road network
- Data Statistics
  - > 1M trajectories, 27.279 roads
  - 99% spatial coverage
  - Peak: ~25%, Off-peak: ~75%
  - Weekend: ~35%, Weekday: ~65%
- Evaluation
  - Random Origin-Destination Queries from trajectories.
  - Comparison of routes with Azure Maps
    - Length of route
    - ETA of route

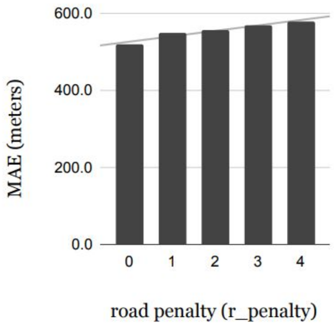


TrajRoute: Experiments

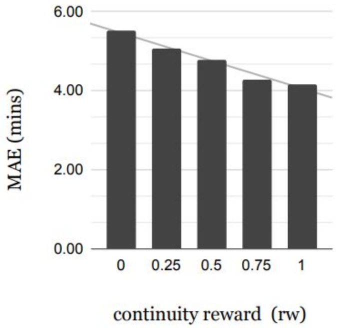
- Data Source
  - San Francisco Taxi Data
  - OSM for road network
- Data Statistics
  - > 1M trajectories, 27.279 roads
  - 99% spatial coverage
  - Peak: ~25%, Off-peak: ~75%
  - Weekend: ~35%, Weekday: ~65%
- Evaluation
  - Random Origin-Destination Queries from trajectories.
  - Comparison of routes with Azure Maps
    - Length of route
    - ETA of route



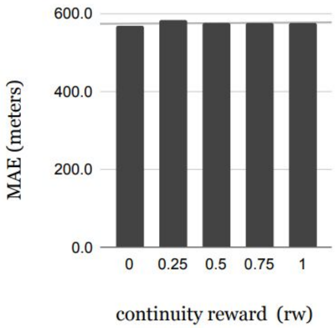
(a) MAE of route travel time



(b) MAE of route distance



(a) MAE of route travel time



(b) MAE of route distance

TrajRoute: Experiments

- Evaluation
  - Random Origin-Destination Queries from trajectories.
  - Comparison of routes with Azure Maps
    - Length of route
    - ETA of route
  - Spatial Coverage
    - Keep trajectories that cover x% of the area
    - Keep *α=3.0* and *rw=0.75* constant

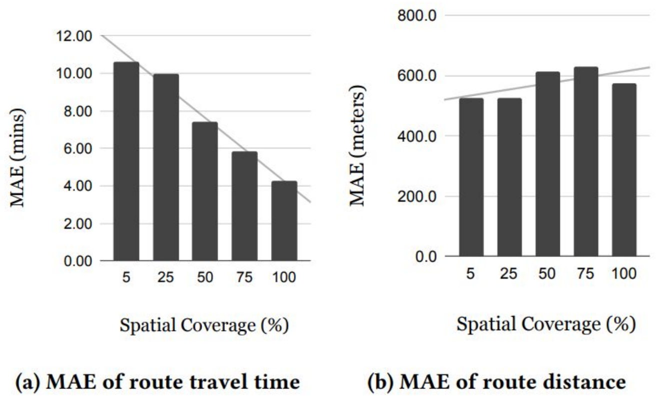
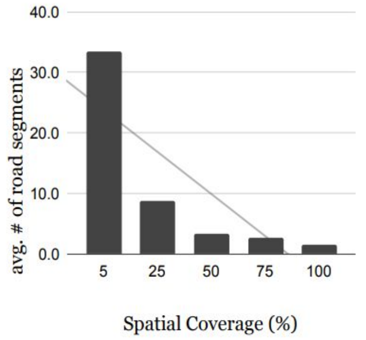


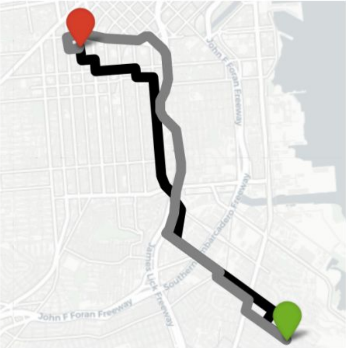
Figure 7: Results for different levels of spatial coverage.



USC Viterbi
School of Engineering
Integrated Media Systems Center

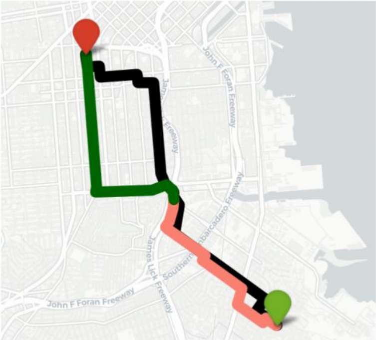
TrajRoute: Experiments

Query Time: *06:01 PM*
Azure Maps ETA: 21 mins



(a) Route for $r_{penalty} = 0$ and $rw = 0$.
TrajRoute ETA: 10 mins.

(b) Route for $r_{penalty} = 3$ and $rw = 0$.
TrajRoute ETA: 16.21 mins.

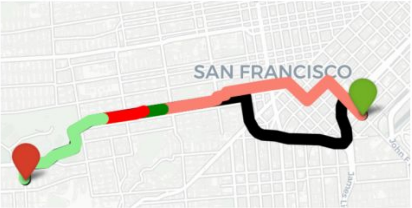


(c) Route for $r_{penalty} = 3$ and $rw = 0.75$.
TrajRoute ETA: 19.53 mins.

Query Time: *01:25 AM*
Azure Maps ETA: 12 mins

(a) Route for $r_{penalty} = 0$ and $rw = 0$.
TrajRoute ETA: 8.3 mins.

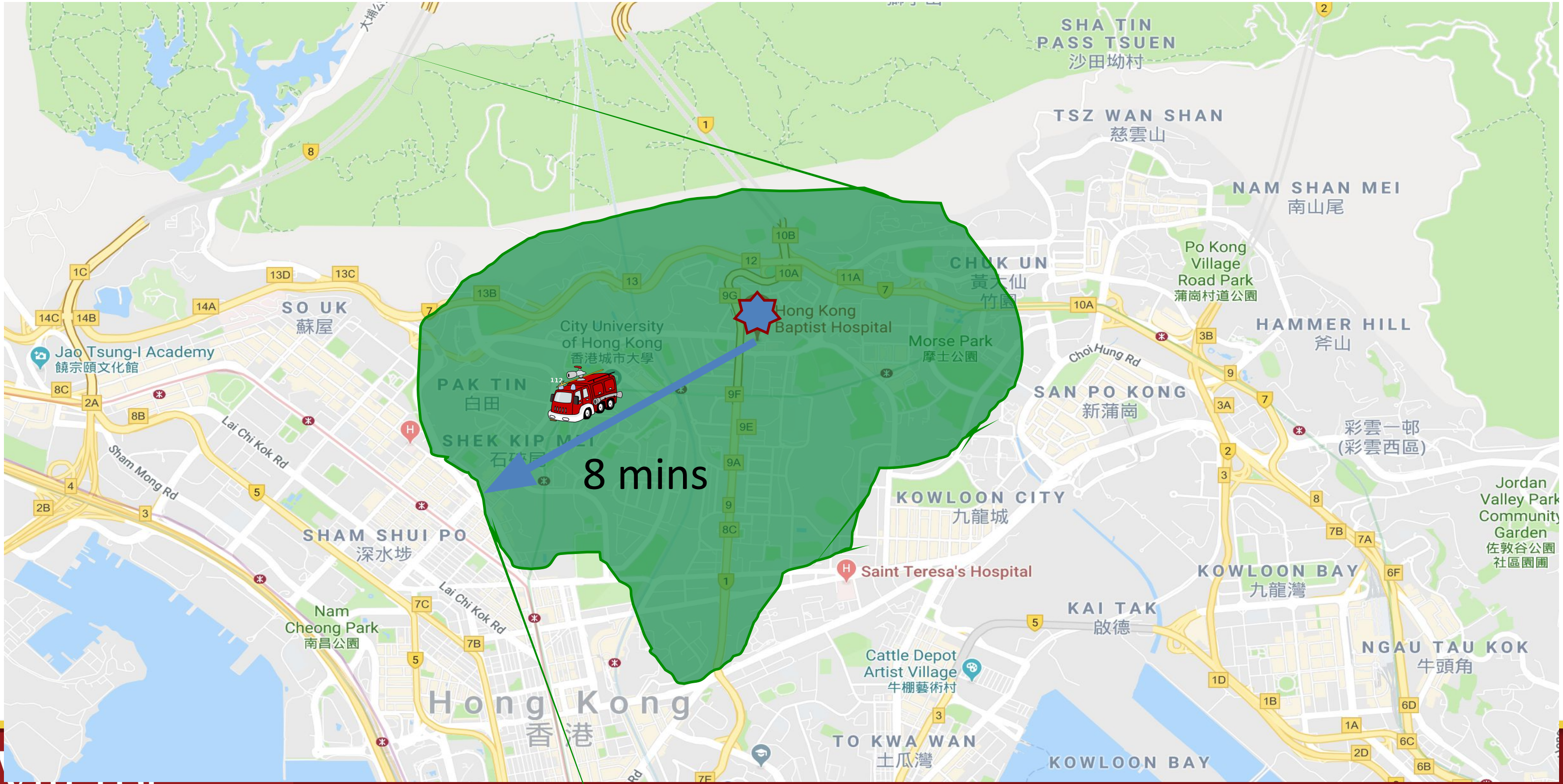
(b) Route for $r_{penalty} = 3$ and $rw = 0$.
TrajRoute ETA: 10.15 mins.



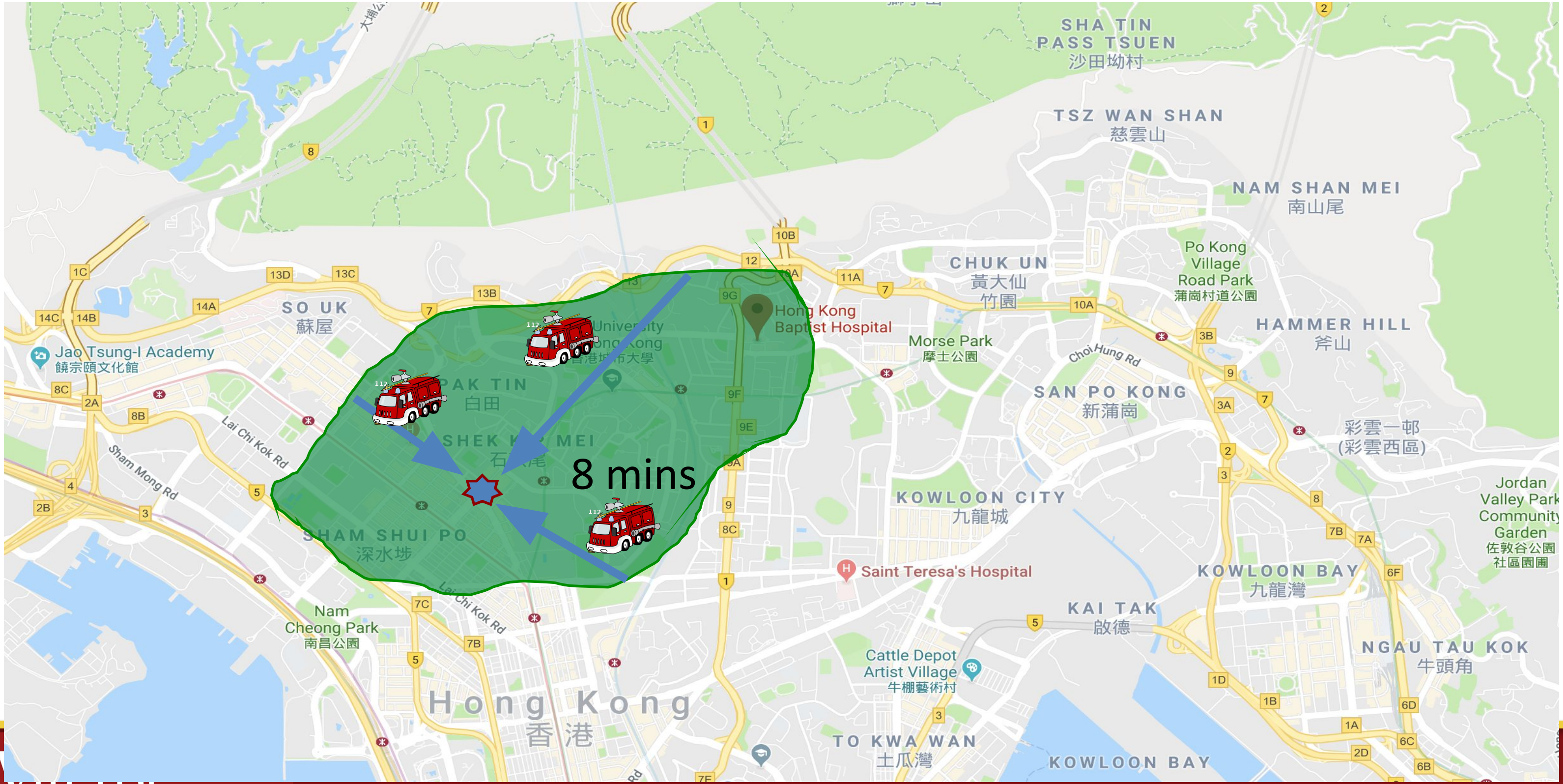
(c) Route for $r_{penalty} = 3$ and $rw = 0.75$.
TrajRoute ETA: 11.58 mins.

Isochrone Maps

**Actual travel times might vary*

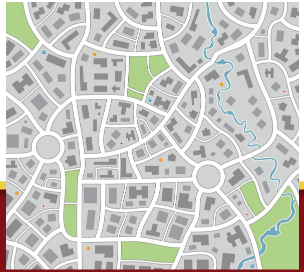


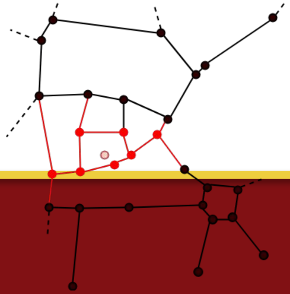
Reverse Reachability Analysis



Graph-based Approaches

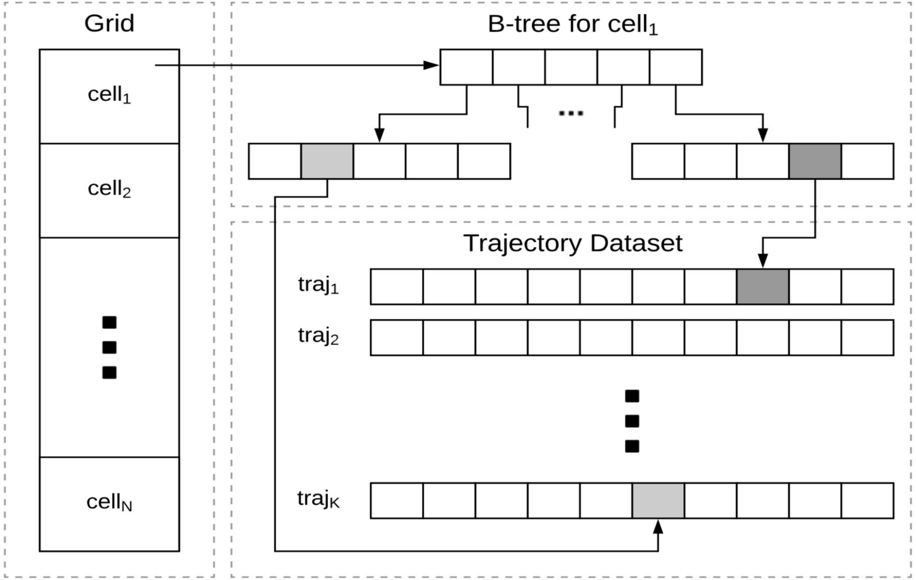
- **Isochrone maps** extensively studied in the databases community
- In **graph theory** defined as the minimal subgraph that can be reached from a query vertex given a limited path cost that is equivalent to travel time.
  - More precisely, it's the set of all reachable vertices, fully traversed edges, and possibly partially traversed edges
- Standard solutions are based on **Dijkstra**'s (or **Dreyfus**) shortest path algorithm
  - **[ICDE'06]** Finding fastest paths on a road network with speed patterns, *Kanoulas et al.*
  - **[GIS'08]** Computing isochrones in multi-modal, schedule-based transport networks, *Bauer et al.*
  - **[EDBT'08]** Finding time-dependent shortest paths over large graphs, *Ding et al.*
  - **[CIKM'11]** Defining isochrones in multi-modal spatial networks, *Gamber et al.*
  - **[SEA'16]** Fast exact computation of isochrones in road networks, *Baum et al.*





Data-Driven Reachability

- Remove the *expensive map-matching* step
  - Can take days to compute time-dependent weights for big data
- Remove the traversal step of *complex* graphs
  - The higher the query time limit the more edges need to be explored
- Compute isochrone maps *directly from data*
  - Only process trajectories that satisfy query criteria
- Support multiple *Reachability Queries*
  - Single-Source & Multi-Source (Normal)
  - Single-Target & Multi-Target (Reverse)

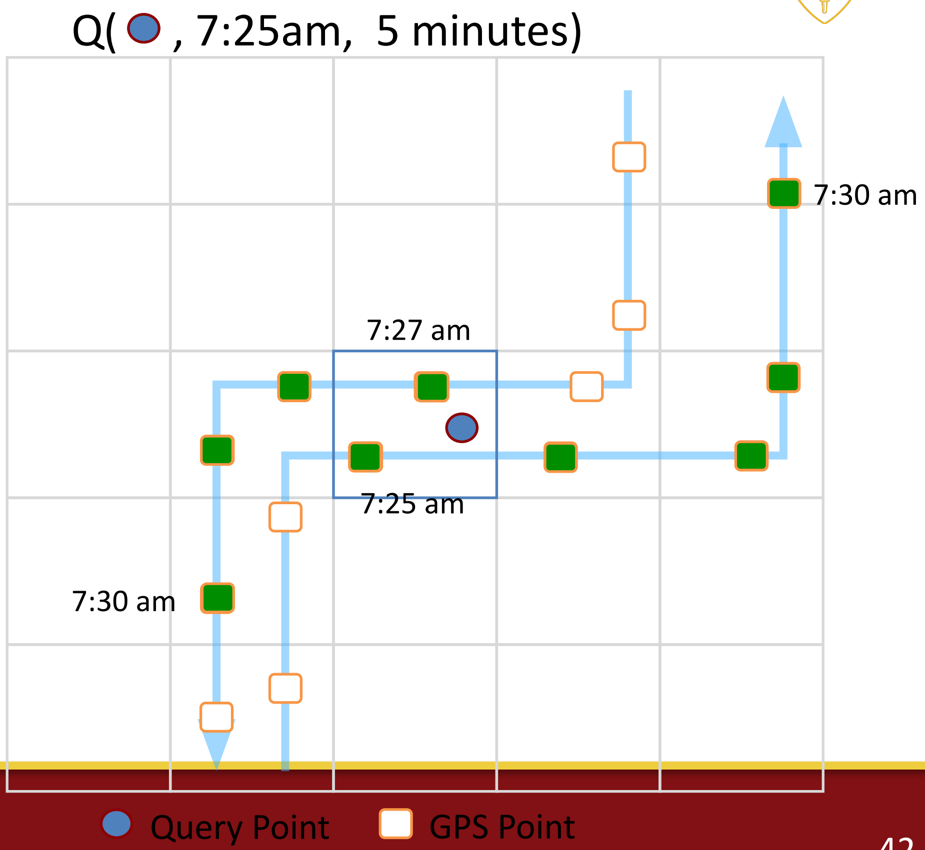


Single-Source & Multi-Source Queries

Reachability Query

- Q(s, t, d)
  - s: source location
  - t: departure time
  - d: time limit in minutes

$c \leftarrow findCell(G, Q.s)$
$r \leftarrow \{\}$ // Initialize result to empty set
for $(traj, i) \in c.gpsInWindow(Q.t, Q.t + Q.d)$ **do**
  while $i < traj.length$ **and** $traj[i].ts \leq Q.t + Q.d$ **do**
    $r \leftarrow r \cup \{traj[i].loc\}$
    $i \leftarrow i + 1$
  end while
end for
return r // Return the set of all reachable points



Single-Target & Multi-Target Queries

Reverse Reachability Query

- Q(q, t, d)
  - q: target location
  - t: arrival time
  - d: time limit in minutes

$c \leftarrow findCell(G, Q.q)$

$r \leftarrow \{\}$ // Initialize result to empty set

for $(traj, i) \in c.gpsInWindow(Q.t - Q.d, Q.t)$ **do**

  while $i \geq 0$ **and** $Q.t - Q.d \leq traj[i].ts$ **do**

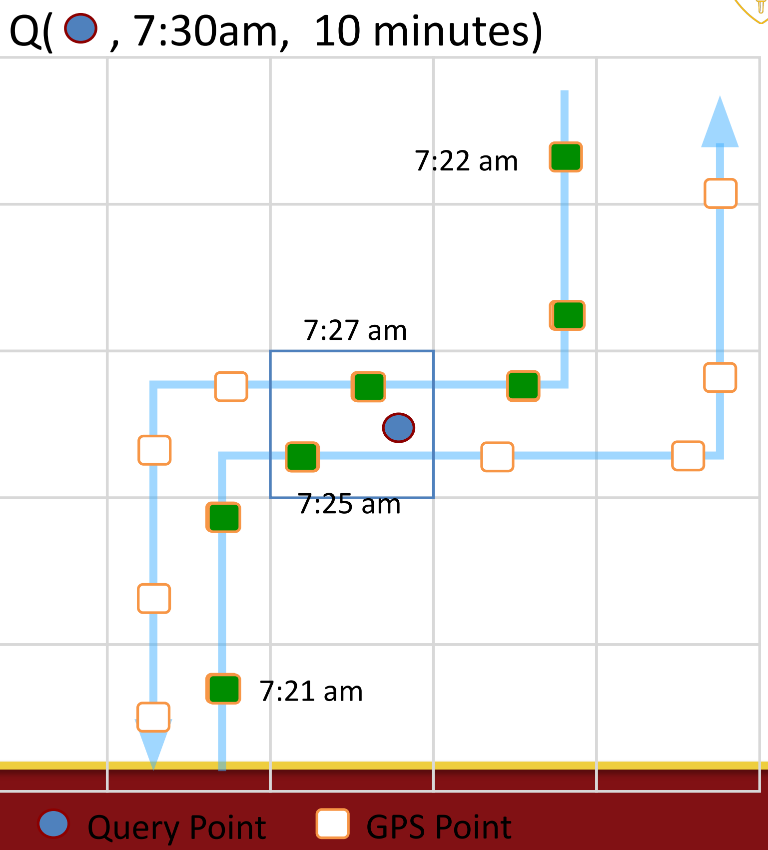
    $r \leftarrow r \cup \{traj[i].loc\}$

    $i \leftarrow i - 1$

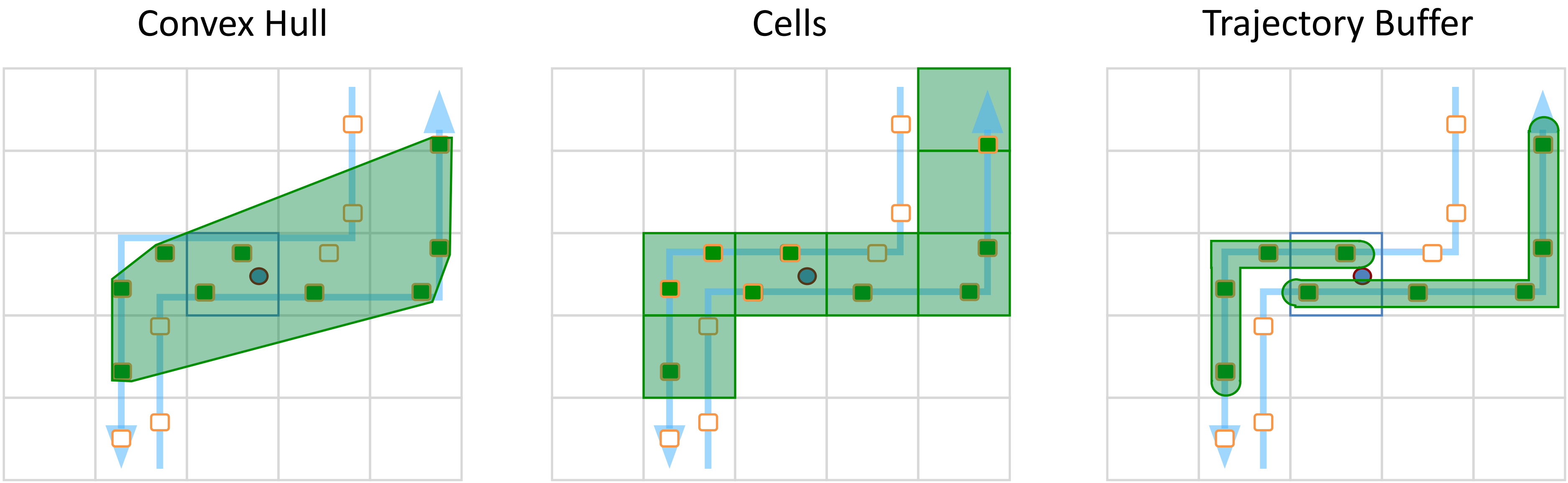
  end while

end for

return r // Return the set of all reachable points



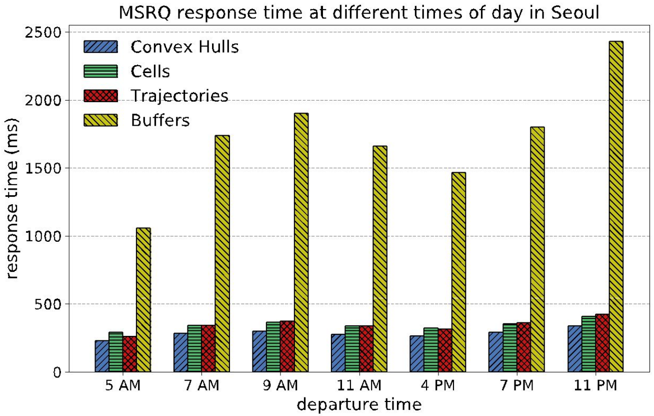
Visualization Methods

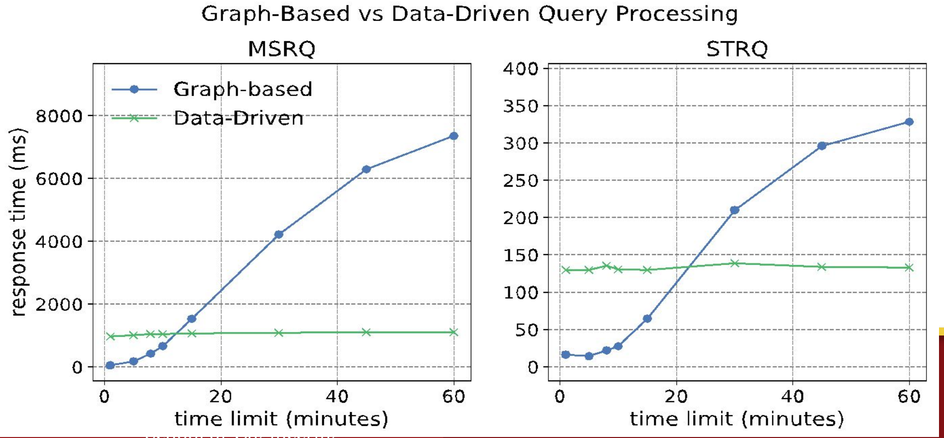


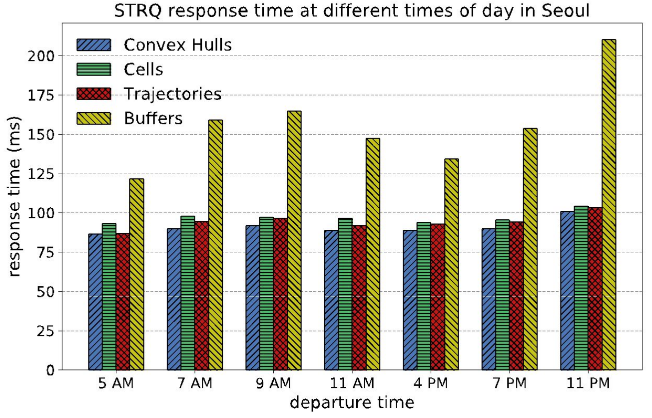
Query Point   GPS Point

Experiments

- Data Source
  - Navicall (Seoul Brand Taxi Call Company)
- Data collection period
  - July 2016 – November 2016
- Data Statistics
  - 5,000 taxies
  - 1 min unit sensing data
  - ~600M readings
  - ~50 GB total







References

[1] Newson P. & Krumm J., "Hidden Markov map matching through noise and sparseness," Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS 09, 336, 2009

[2] DiDi's IJCAI-19 Tutorial: Artificial Intelligence in Transportation (slides 28–40)

[3] Map Matching @ Uber

[4] Anastasiou, C., Huang, C., Kim, S. H., & Shahabi, C. (2019, June). Time-Dependent Reachability Analysis: A Data-Driven Approach. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)* (pp. 138-143). IEEE.

[5] Siampou, MD., Anastasiou, C., Krumm, J., & Shahabi, C. (2024). TrajRoute: Rethinking Routing with a Simple Trajectory-based Approach: Forget the Maps and Traffic!. In *Submission.*