Part 2 - 02: Urban Data Management (2) (Spatio-Temporal Data)

Long Cheng

Assistant Professor

c.long@ntu.edu.sg

Urban Data: Categorization

Spatial Data

Point-of-interest

Road network

Spatio-temporal Data

Vehicle trajectories

Air quality

Network Data

Road network

Road traffic

Urban Data Management

Preprocessing

Indexing

Query Processing

Spatial Data

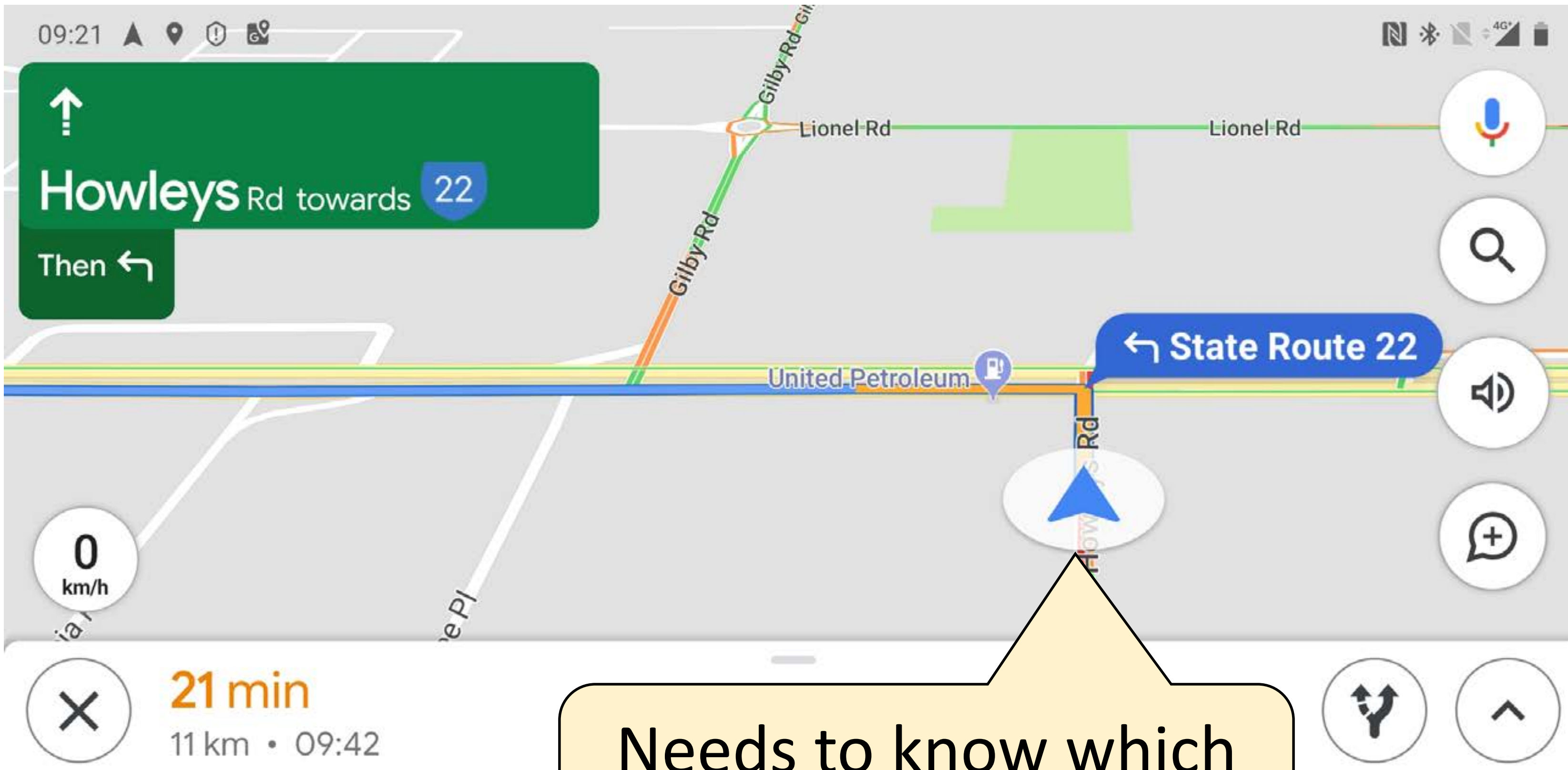
Spatio-Temporal Data

Network Data

Trajectory Data Preprocessing

Map Matching	**Trajectory Simplification**	**Others**

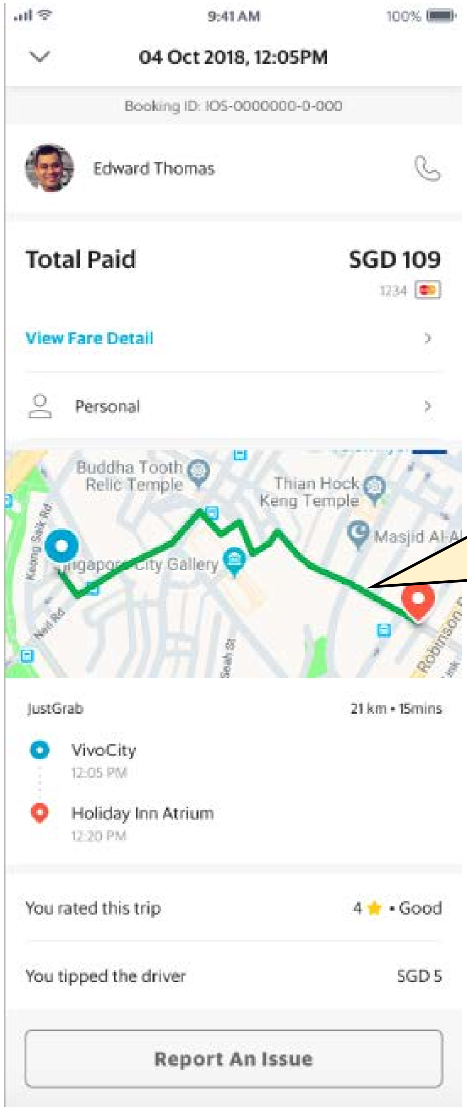
Application Scenario 1: Driving Navigation



Source: https://support.google.com/maps/thread/14562336?hl=en

Needs to know which road segment I am on

Application Scenario 2: Taxi Fare Calculation



Needs to know which route I traveled

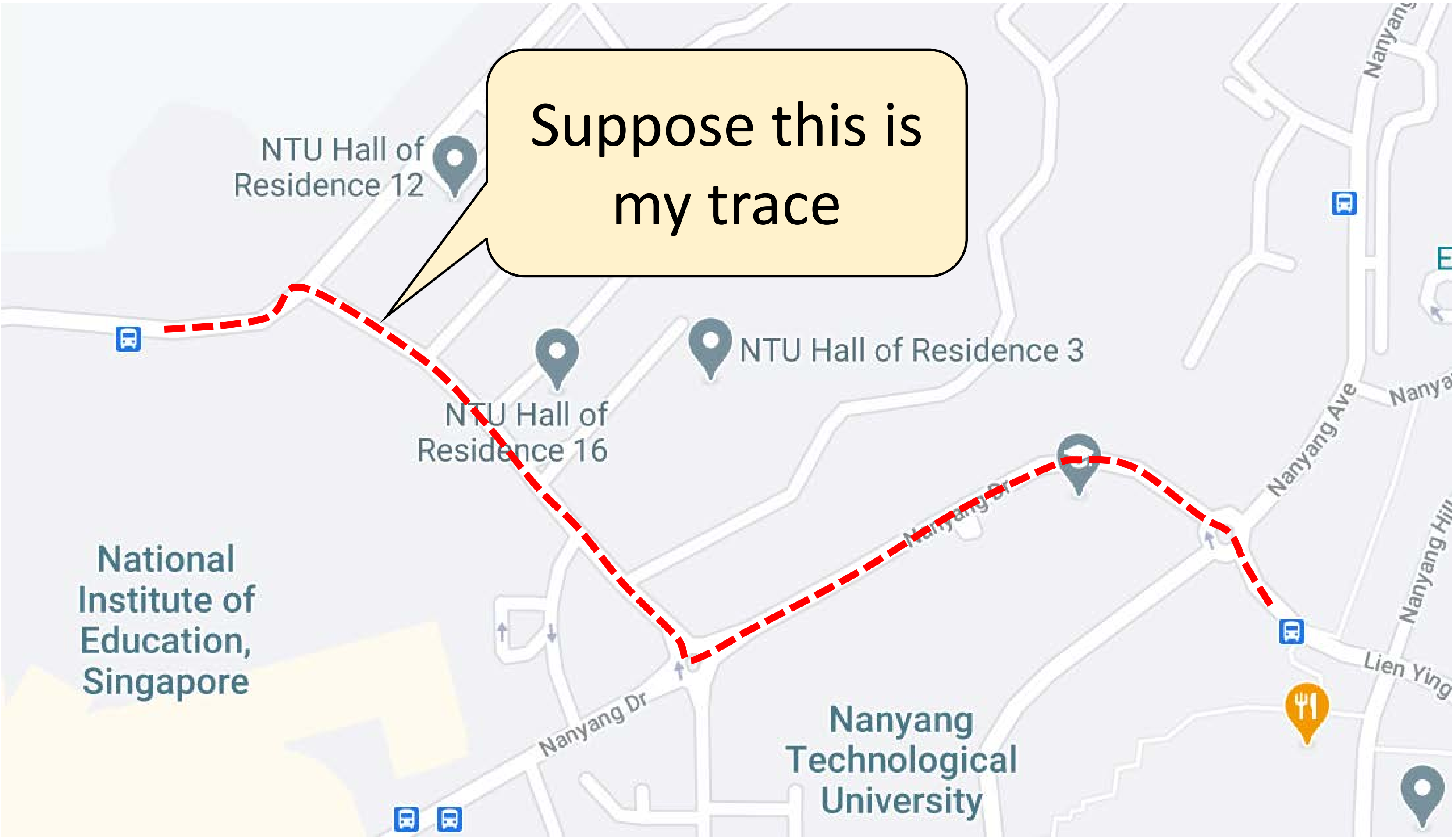
Source: https://help.grab.com/passenger/en-sg/360001392388-Checking-your-trip-history

Map Matching: Motivation

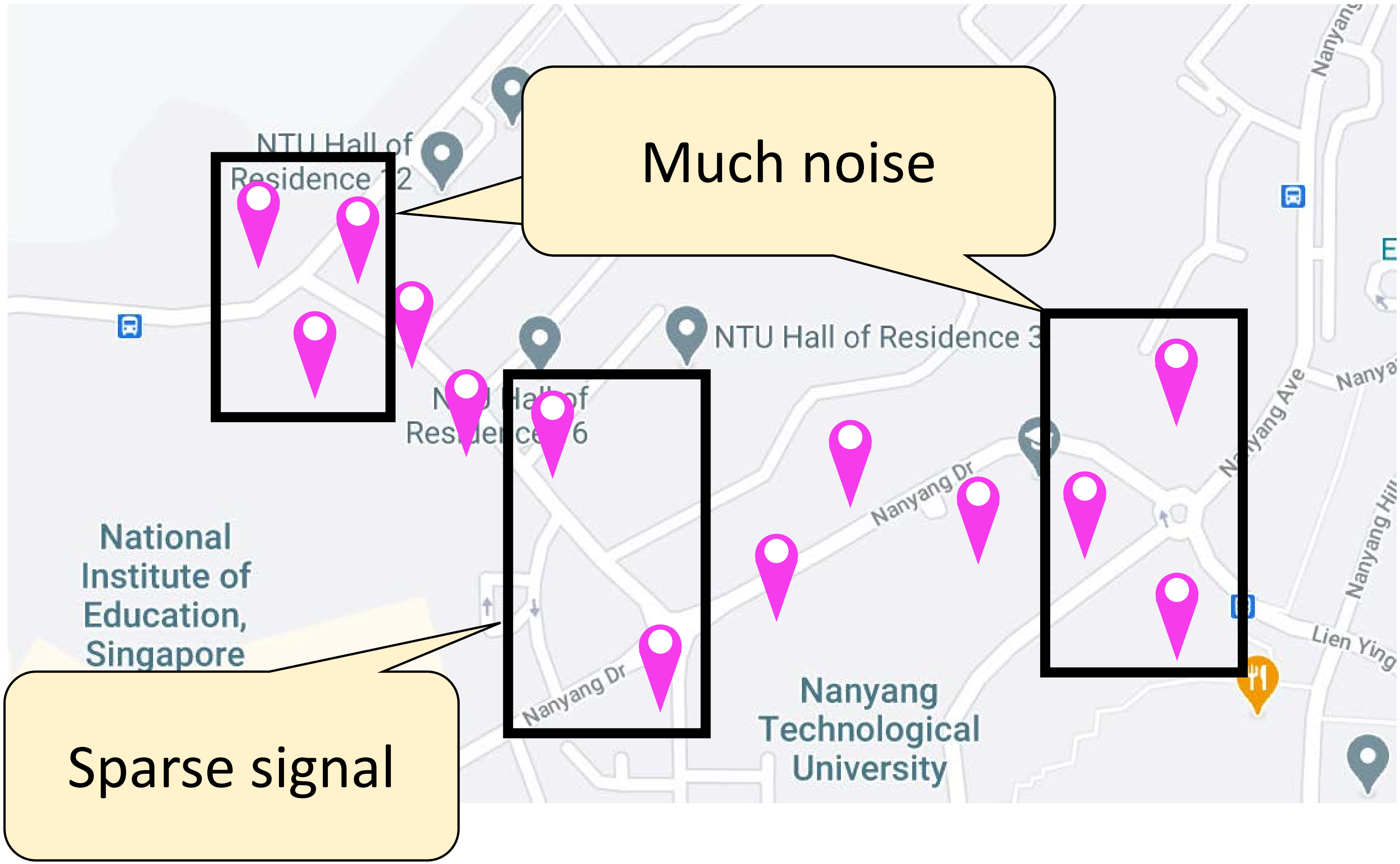
These applications are built on GPS signals and require the route information

Going from GPS signals to routes is necessary… (**Map matching**)

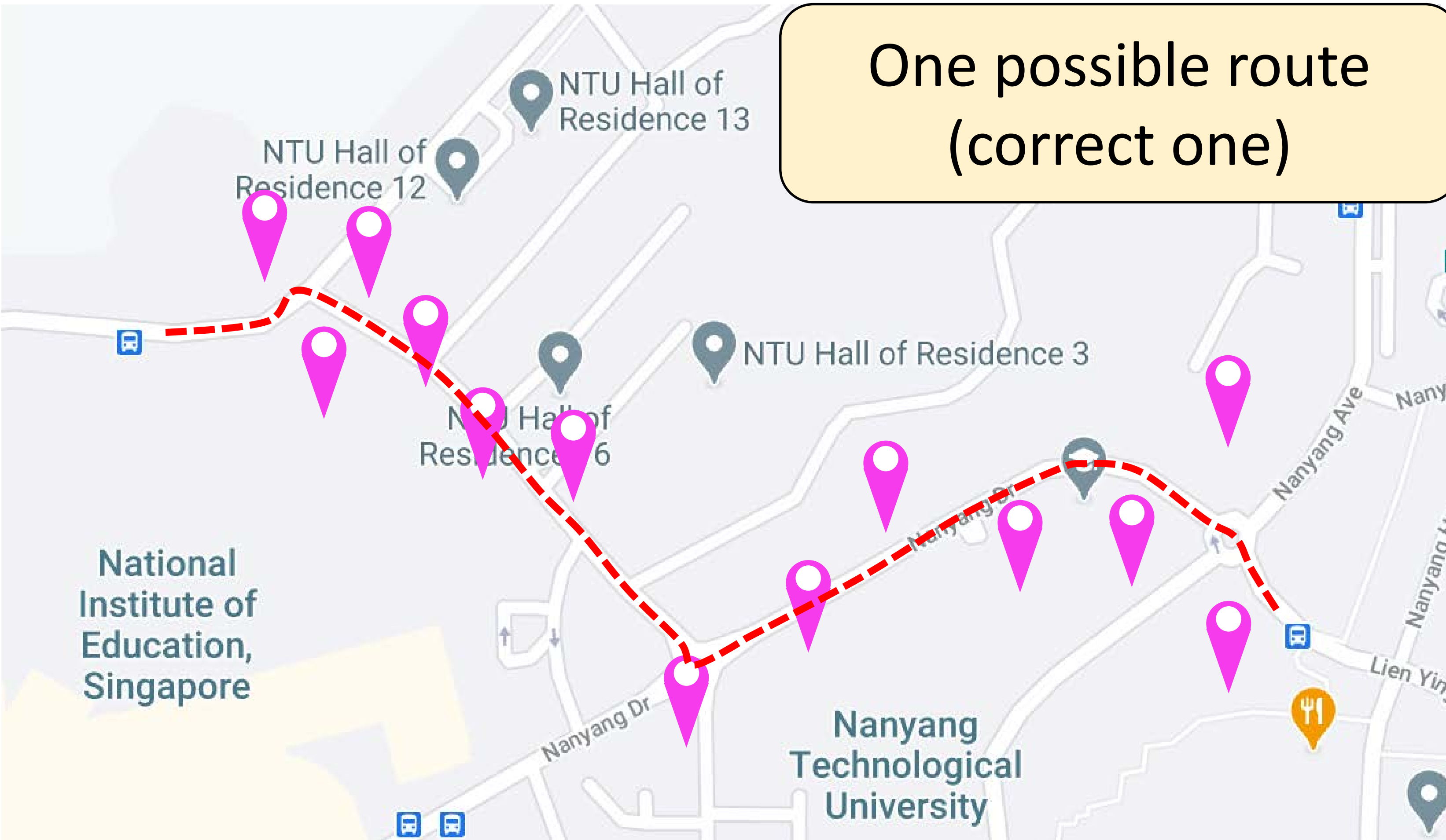
Map Matching: Motivation



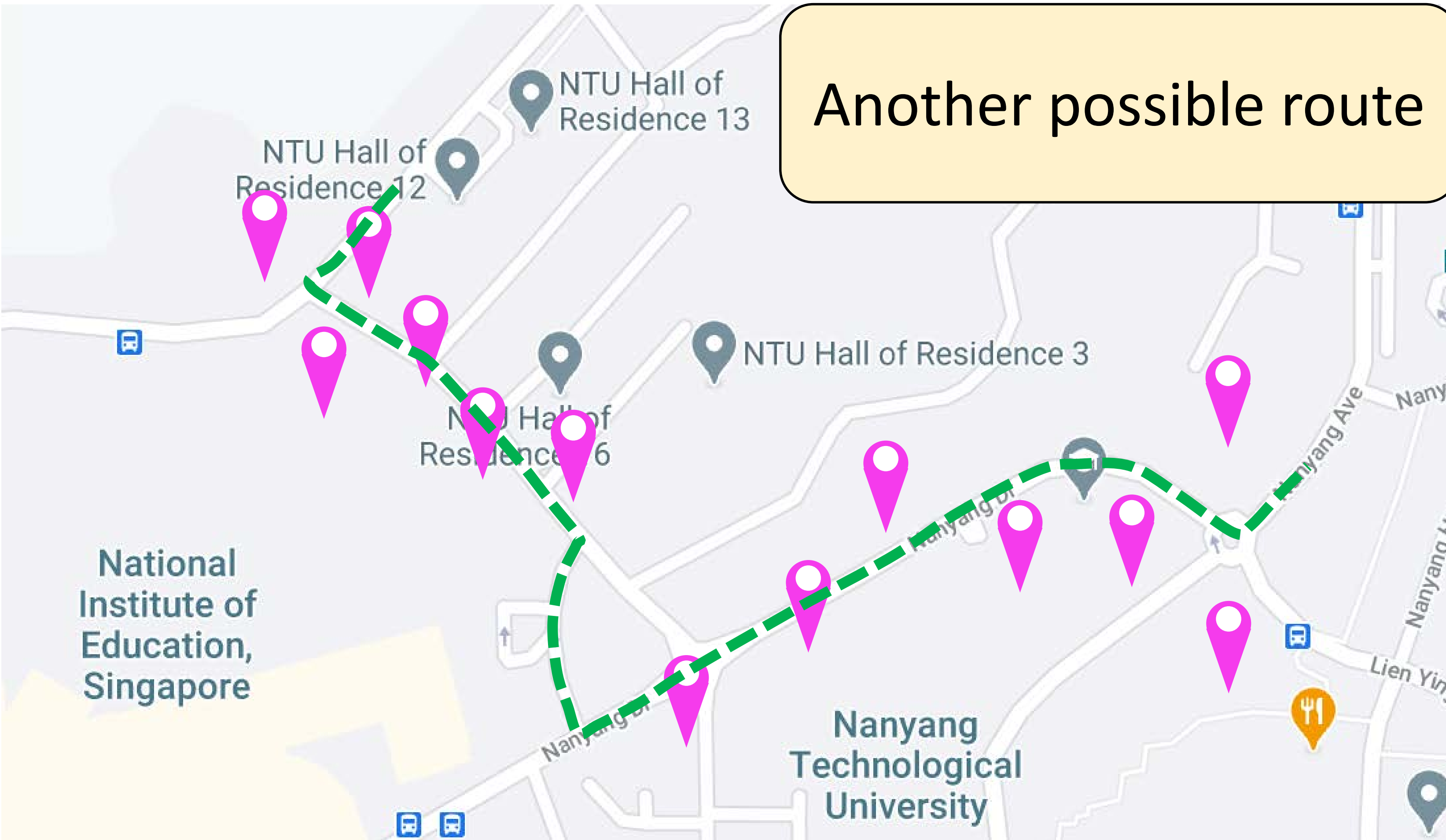
Map Matching: Motivation



Map Matching: Motivation



Map Matching: Motivation



Another possible route

Map Matching: Definition

Map matching:

- **Input:** a sequence of GPS signals
- **Output:** a sequence of road segments

Map Matching: Methods

Geometric Analysis

- Point-to-point
- Point-to-curve
- Curve-to-curve

Topological Analysis

- Edge connectivity
- Edge shape
- …

Advanced Algorithms

- Hidden Markov Model (HMM)
- Kalman Filter
- …

Map Matching with HMM

Year 2009

Hidden Markov map matching through noise and sparseness

P Newson, J Krumm - Proceedings of the 17th ACM SIGSPATIAL …, 2009 - dl.acm.org

The problem of matching measured latitude/longitude points to roads is becoming increasingly important. This paper describes a novel, principled map matching algorithm that uses a Hidden Markov Model (HMM) to find the most likely road route represented by a time-stamped sequence of latitude/longitude pairs. The HMM elegantly accounts for measurement noise and the layout of the road network. We test our algorithm on ground truth data collected from a GPS receiver in a vehicle. Our test shows how the algorithm …

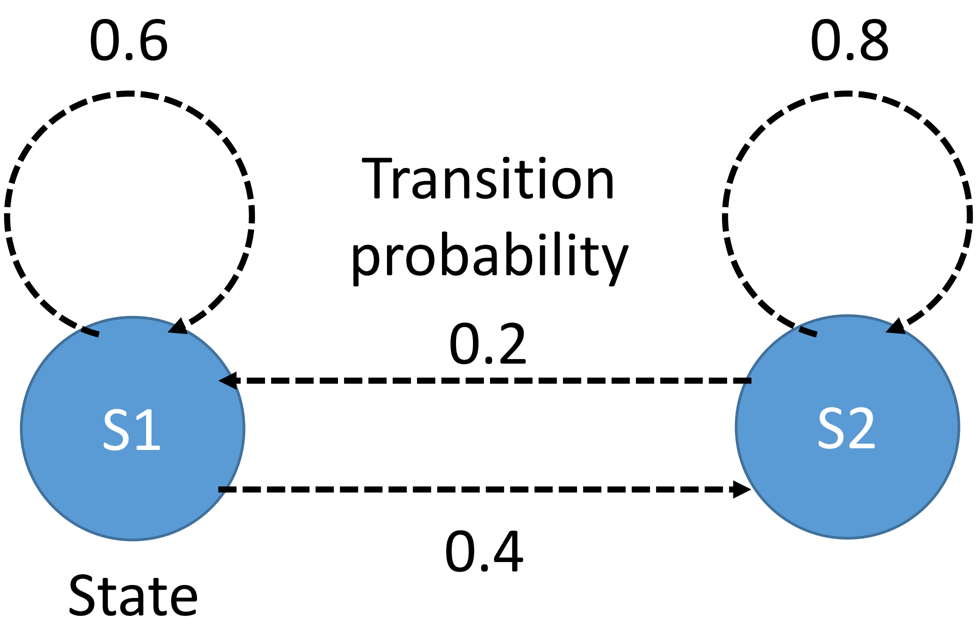
☆ ❞ Cited by 742 Related articles All 7 versions

Cited by 742

HMM Background

Markov process:

- Random process (of a sequence states)
- The probability of the current state depends only on the previous state **(Markov property)**

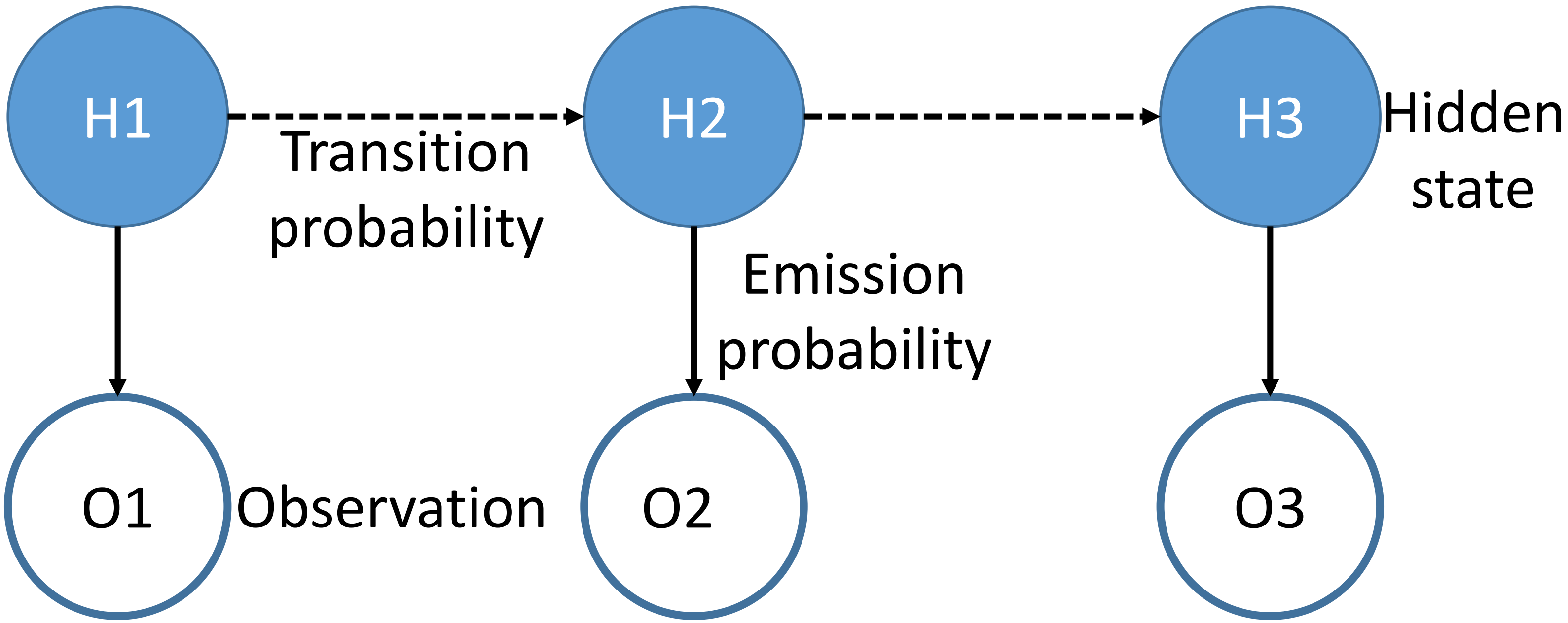


Markov process

HMM Background

Hidden Markov Model (HMM):

- Markov process + unobservable state
- Observations, which depend only on the current state, are visible



Hidden Markov Model

HMM Background

HMM Inference problem:

- **Input:** a sequence of **observations** and an HMM
- **Output:** the most likely **hidden state** sequence

HMM Background

Sequence of observations:

y1 y2 y1 y3 y4

Most likely sequence of hidden states?

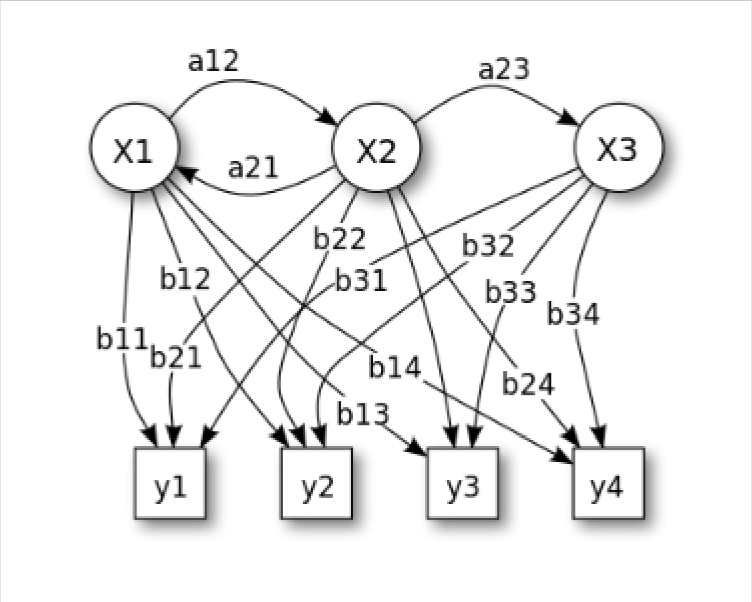


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Source: https://en.wikipedia.org/wiki/Hidden_Markov_model

HMM Background

Sequence of observations:

y1 y2 y1 y3 y4

x1 x2 x1 x2 x3
x2 x1 x2 x1 x2
x1 x2 x1 x2 x1
...

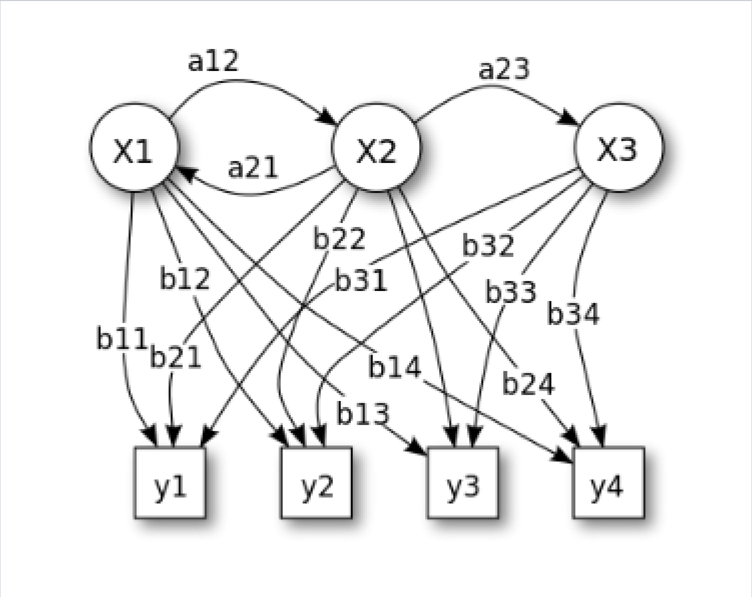


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Source: https://en.wikipedia.org/wiki/Hidden_Markov_model

HMM Background

Sequence of observations:

y1 y2 y1 y3 y4

Viterbi Algorithm

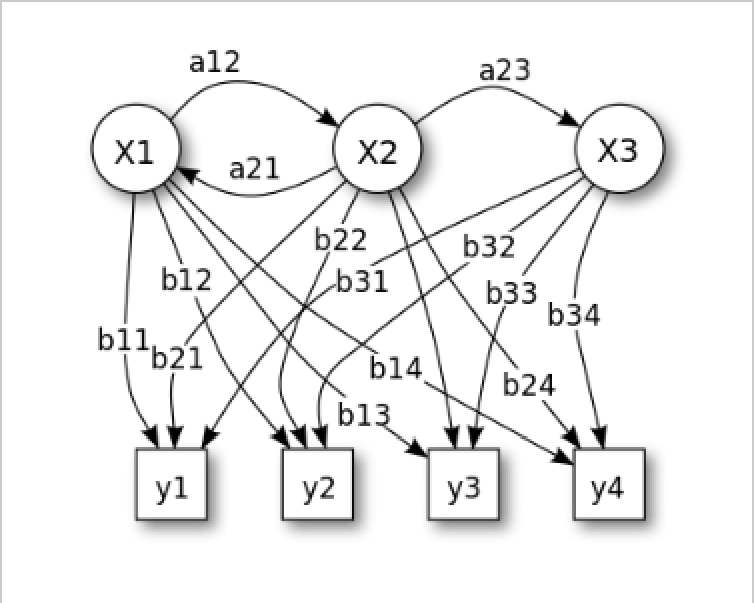


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Source: https://en.wikipedia.org/wiki/Hidden_Markov_model

HMM Background

```
function VITERBI(O, S, Π, Y, A, B) : X
    for each state i = 1, 2, ..., K do
        T1[i, 1] ← π_i · B_{iy_1}
        T2[i, 1] ← 0
    end for
    for each observation j = 2, 3, ..., T do
        for each state i = 1, 2, ..., K do
            T1[i, j] ← max_k (T1[k, j−1] · A_ki · B_{iy_j})
            T2[i, j] ← argmax_k (T1[k, j−1] · A_ki · B_{iy_j})
        end for
    end for
    z_T ← argmax_k (T1[k, T])
    x_T ← s_{z_T}
    for j = T, T−1, ..., 2 do
        z_{j−1} ← T2[z_j, j]
        x_{j−1} ← s_{z_{j−1}}
    end for
    return X
end function
```

Source: https://en.wikipedia.org/wiki/Viterbi_algorithm

Map Matching with HMM

Map matching:

- **Input:** a sequence of GPS signals
- **Output:** a sequence of road segments

HMM Inference:

- **Input:** a sequence of **observations** and an HMM
- **Output:** the most likely **hidden state** sequence

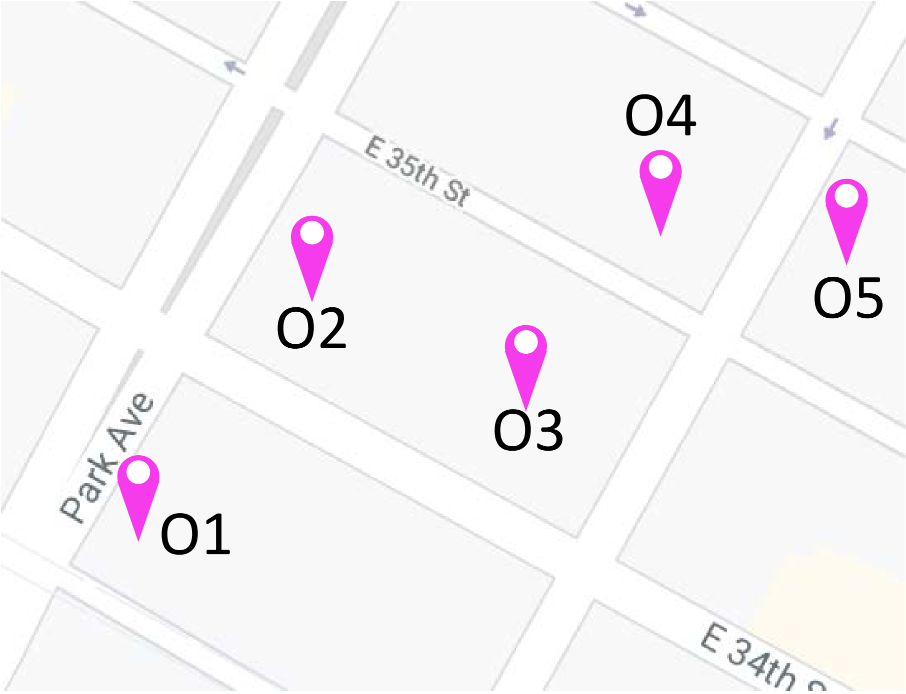
Map Matching with HMM

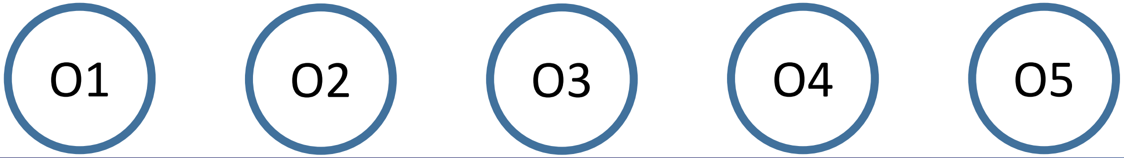
Map matching via HMM Inference:

1. Regard each GPS signal as an observation
2. For each GPS signal, find its 2 nearest road segments as its possible hidden states
3. Assign appropriate emission probabilities from hidden states to the observations
4. Assign appropriate transition probabilities among hidden states
5. Run the Viterbi algorithm and infer the most likely sequence of hidden states (road segments)

Map Matching with HMM

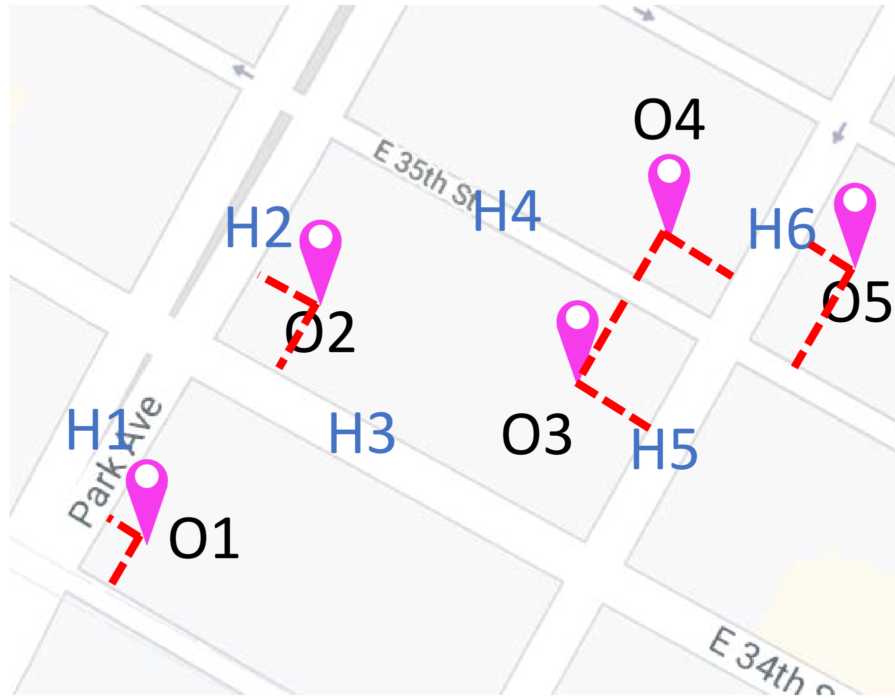
1. Regard each GPS signal as an observation



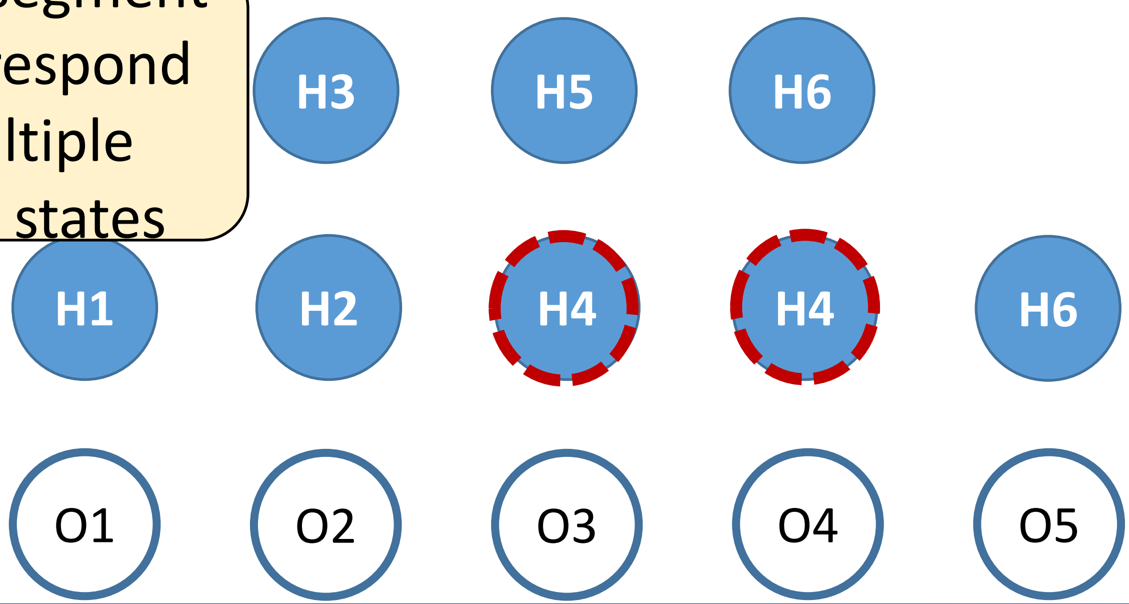


Map Matching with HMM

2. For each GPS signal, find its 2 nearest road segments as its possible hidden states

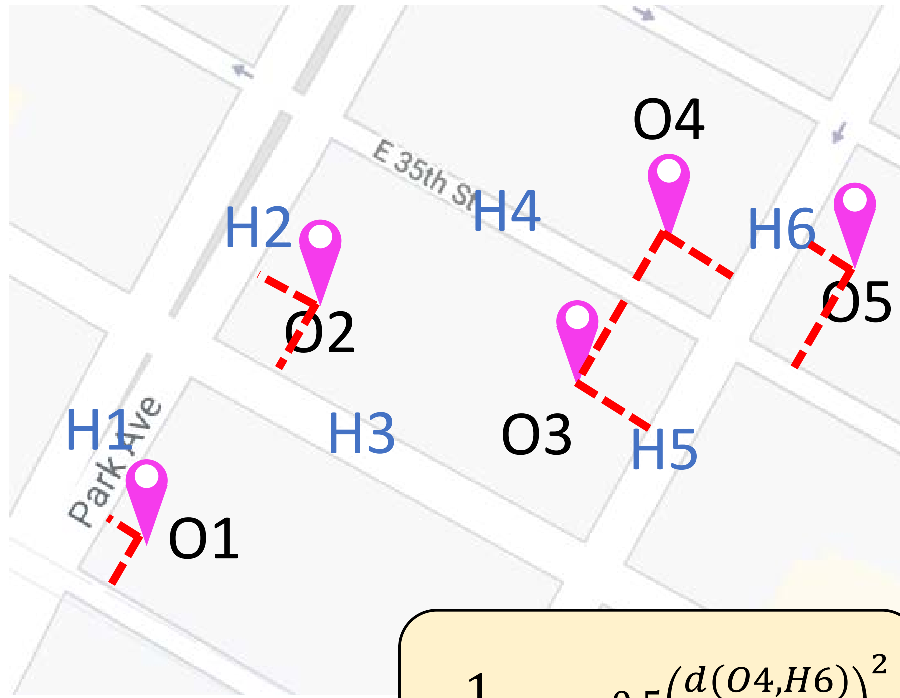


Note. A segment can correspond to multiple hidden states

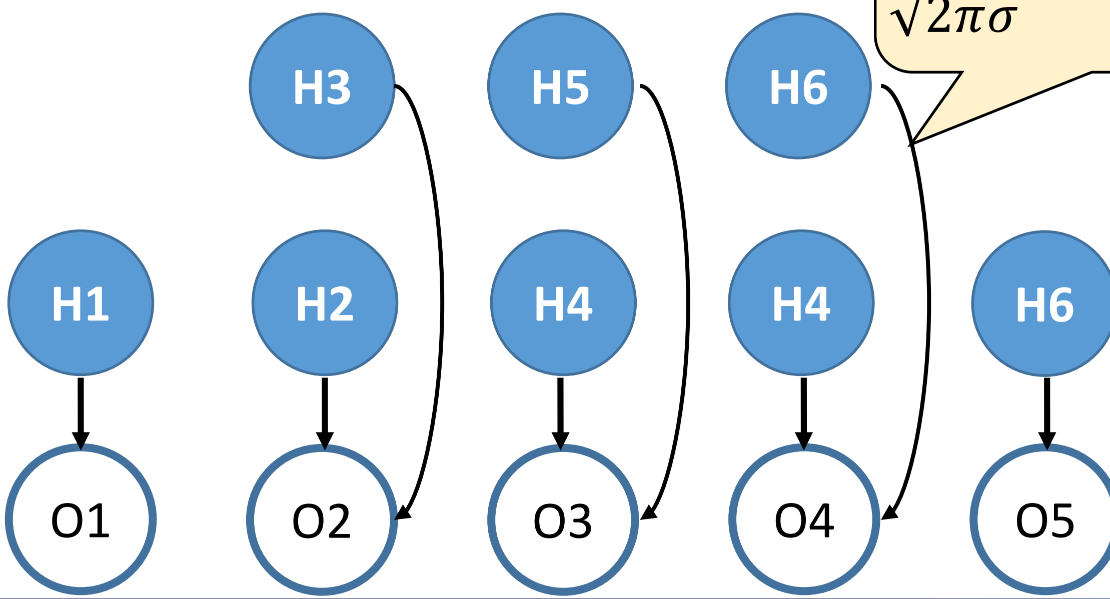


Map Matching with HMM

3. Assign appropriate emission probabilities from hidden states to the observations

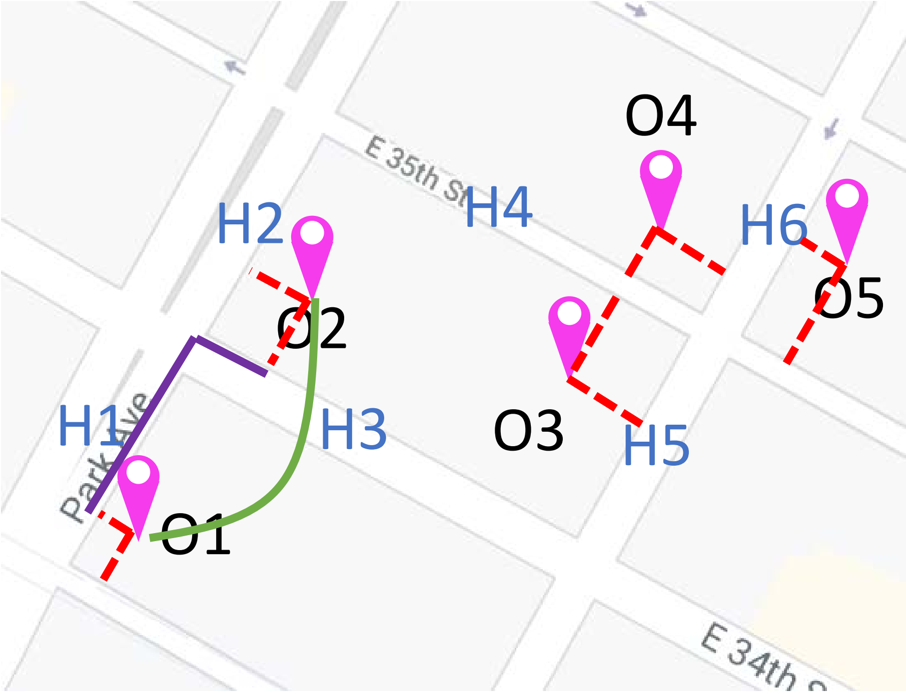


$$\frac{1}{\sqrt{2\pi}\sigma} e^{-0.5\left(\frac{d(O4,H6)}{\sigma}\right)^2}$$



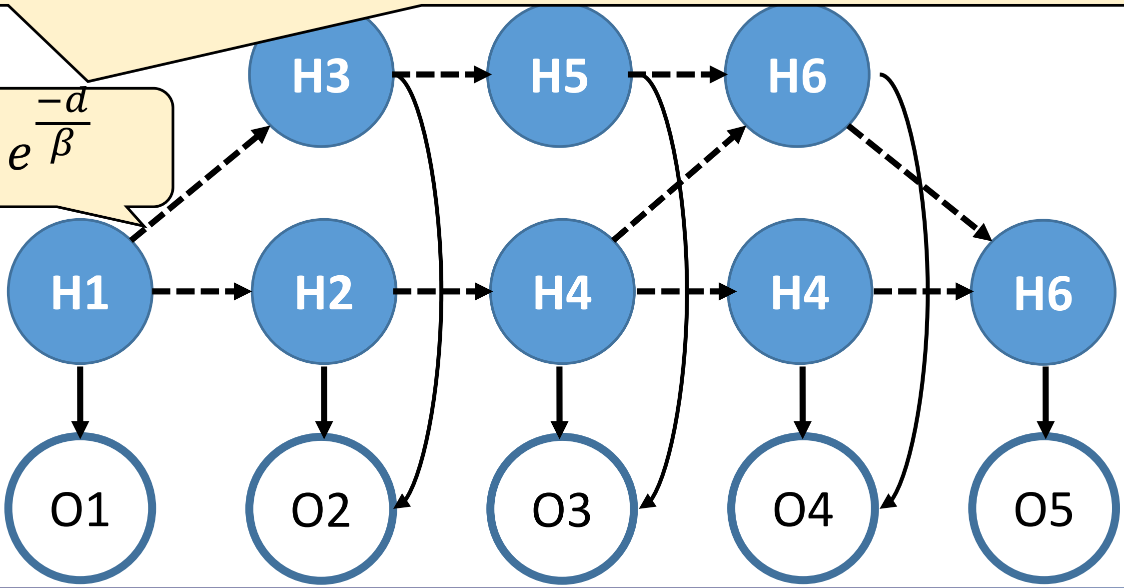
Map Matching with HMM

4. Assign appropriate transition probabilities among hidden states



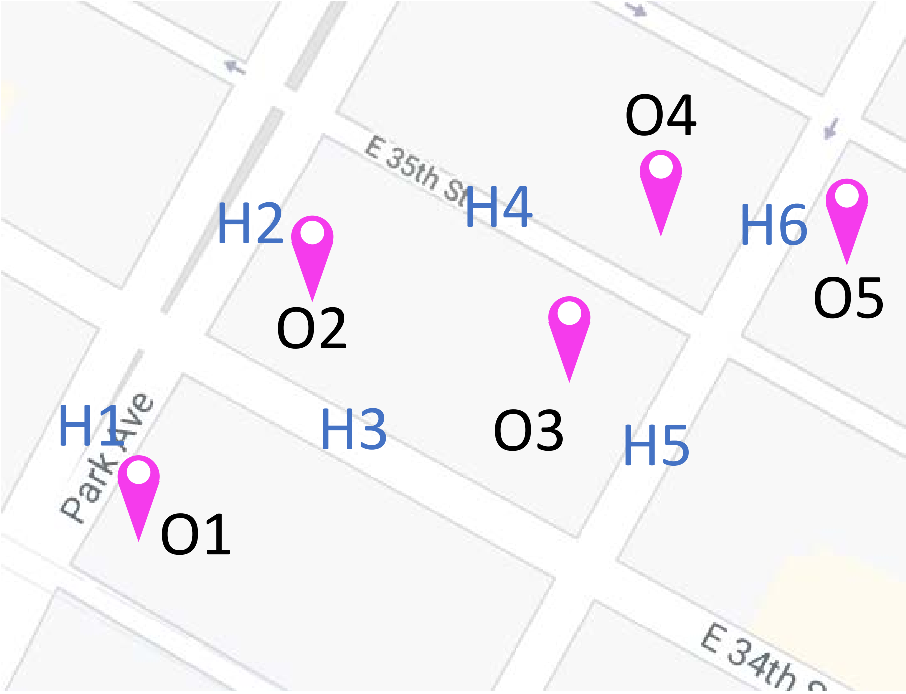
d = |geodesic distance (O1, O2) – network distance (H1, H3)|

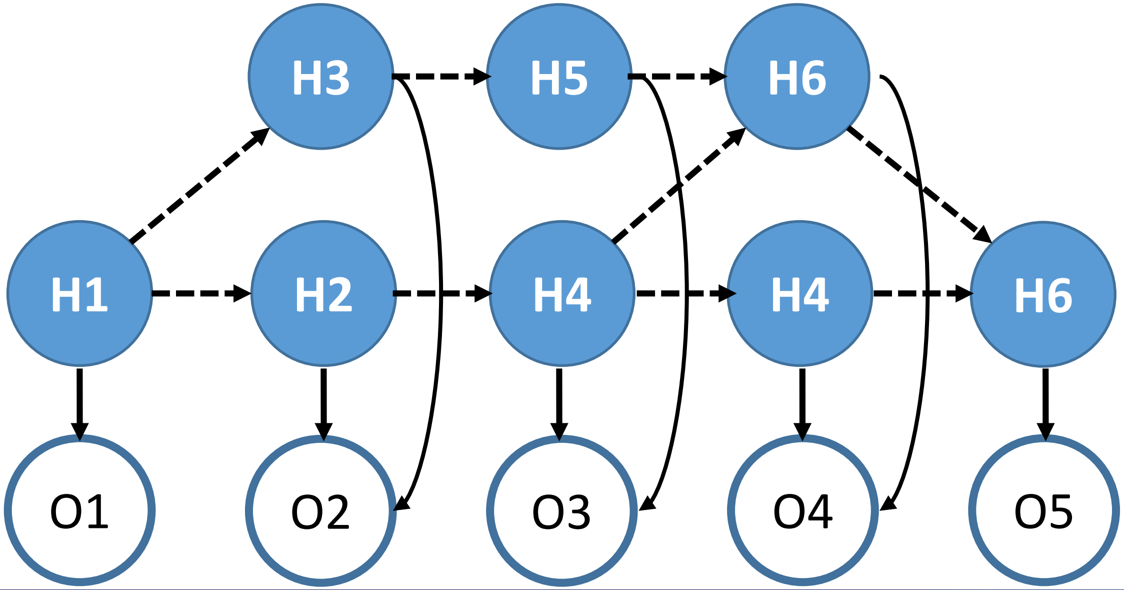
$$\frac{1}{\beta}e^{\frac{-d}{\beta}}$$



Map Matching with HMM

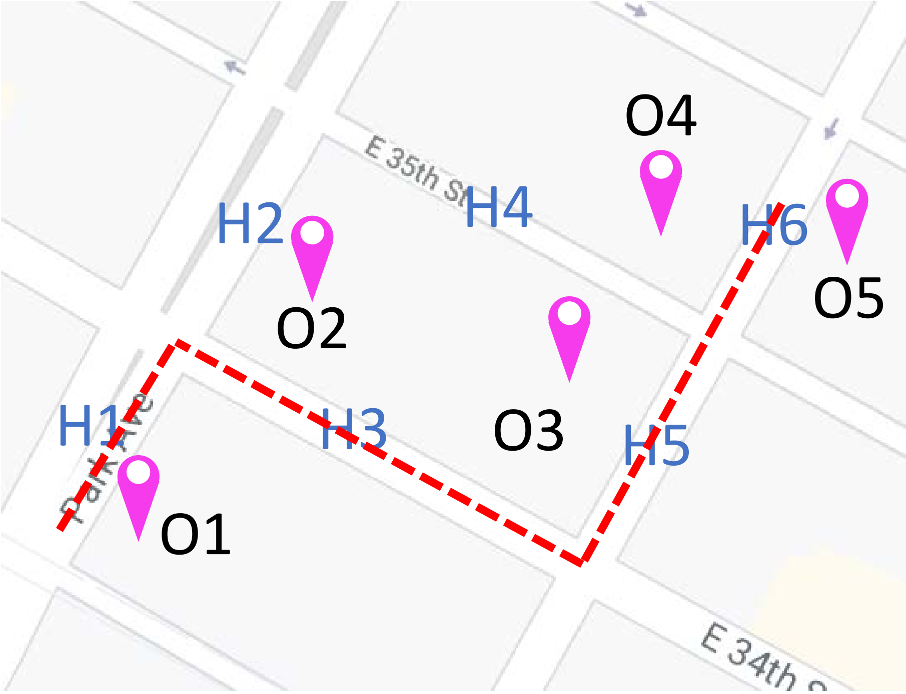
5. Run the Viterbi algorithm and infer the most likely sequence of hidden states (road segments)

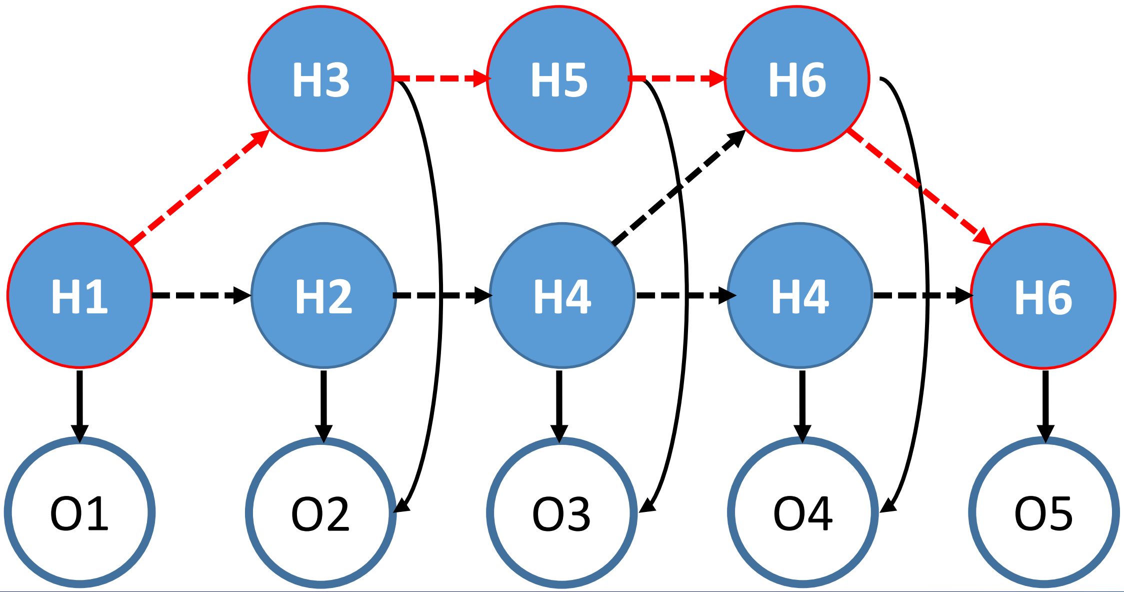




Map Matching with HMM

5. Run the Viterbi algorithm and infer the most likely sequence of hidden states (road segments)

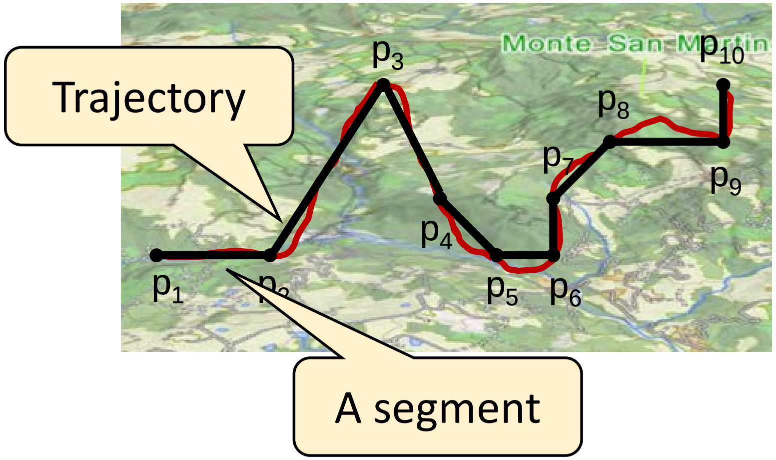




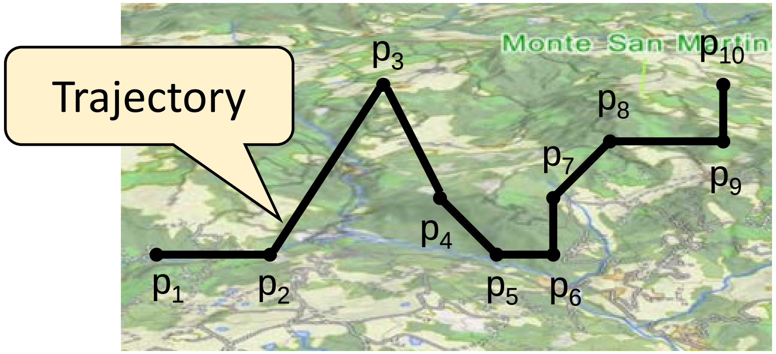
Trajectory Data Preprocessing

Map Matching	Trajectory Simplification	Others

Trajectory Simplification: Introduction



Trajectory Simplification: Introduction



10 sampled positions

9 segments

Trajectory Simplification: Motivation

Raw trajectory data is usually very large

Consider a scenario,

- 10,000 taxis
- Sampling rate: 5s
- $\approx \mathbf{1.7 \times 10^8}$ positions **per day**!!

Issue 1: Storing all sampled positions incurs a very high **space cost**

Issue 2: Query processing big trajectory data incurs high **time cost**

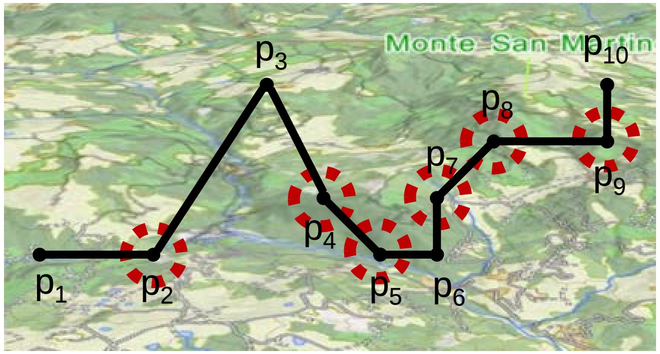
Trajectory Simplification

Drop some positions

As a result, only a portion of the positions is kept

Trajectory Simplification: Motivation

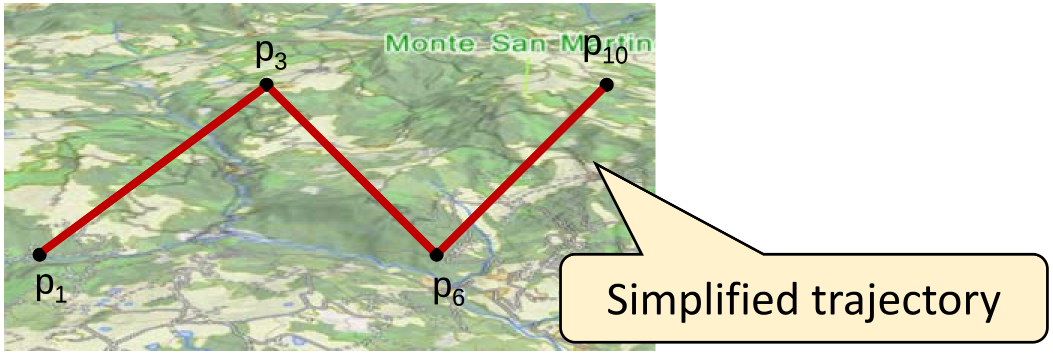
Trajectory Simplification:
Drop some positions



Suppose p_2, p_4, p_5, p_7, p_8, p_9 are dropped

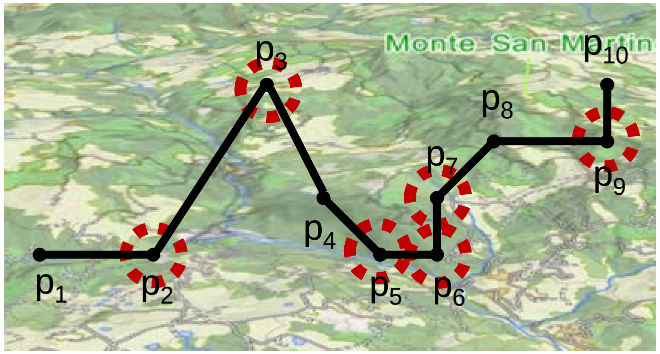
Trajectory Simplification: Motivation

Trajectory Simplification:
Drop some positions



Trajectory Simplification: Motivation

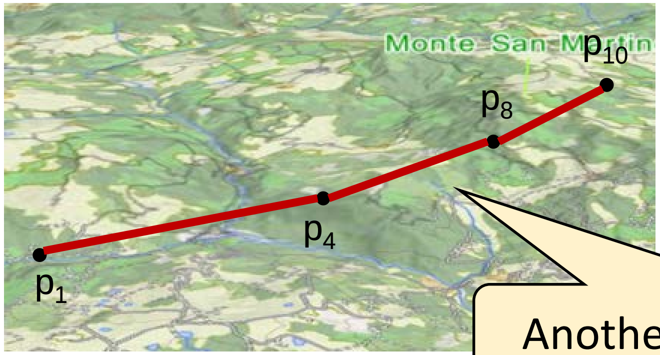
Trajectory Simplification:
Drop some positions



Suppose p_2, p_3, p_5, p_6, p_7, p_9 are dropped

Trajectory Simplification: Motivation

Trajectory Simplification:
Drop some positions



Another Simplified trajectory

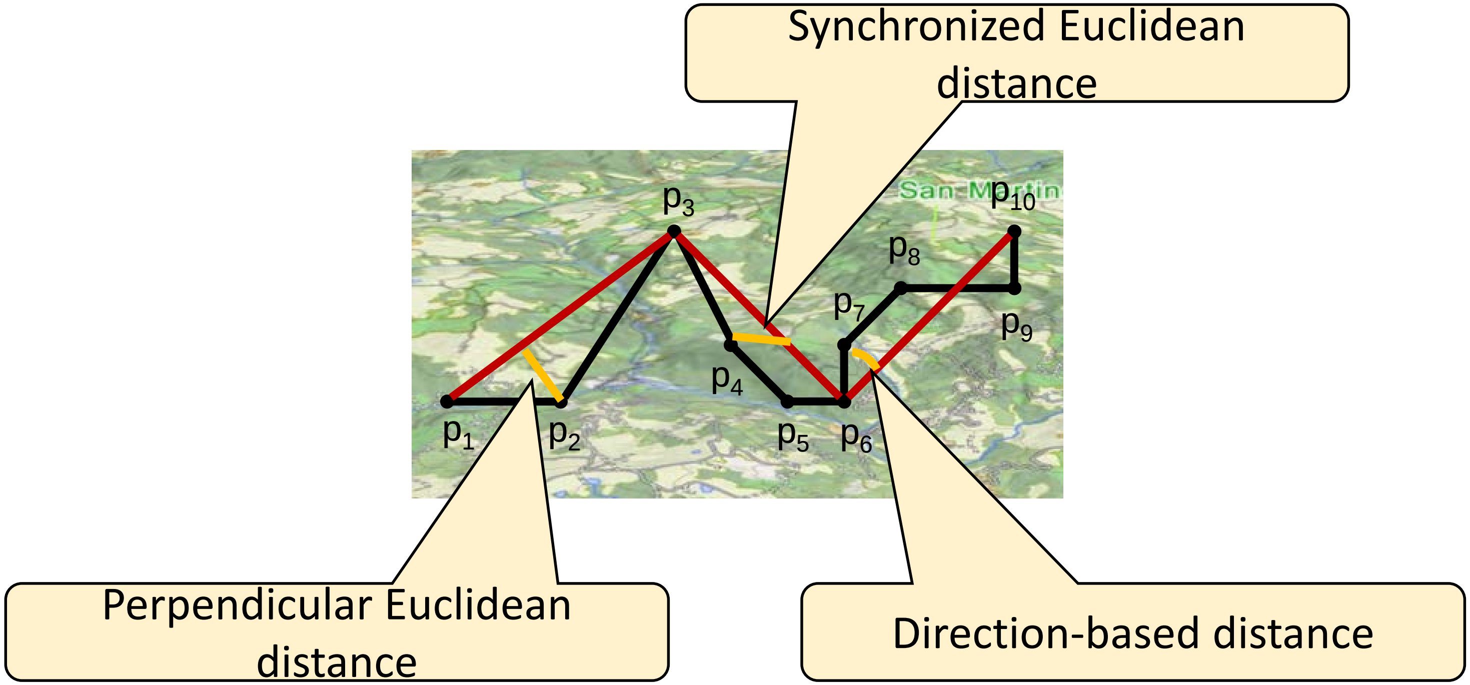
Trajectory Simplification: Problem

Trajectory Simplification:
Drop some positions

Which positions should be dropped?

Depending on which positions to be dropped, it returns different simplified trajectories

Trajectory Simplification: Problem



Trajectory Simplification: Problem

Problem (Min-Size):

Objective: drop as many positions as possible

Constraint: the error is at most a given error tolerance

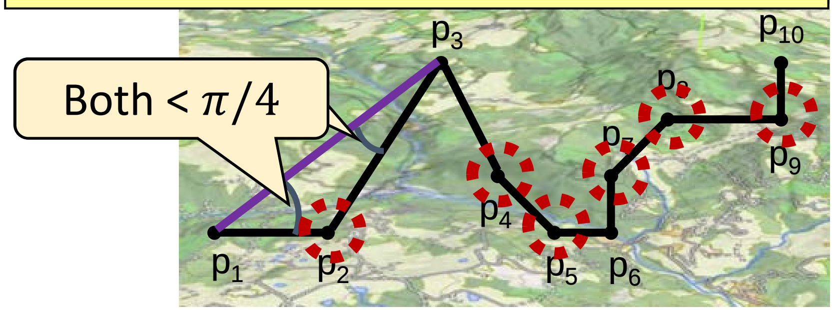
Trajectory Simplification: Algorithms

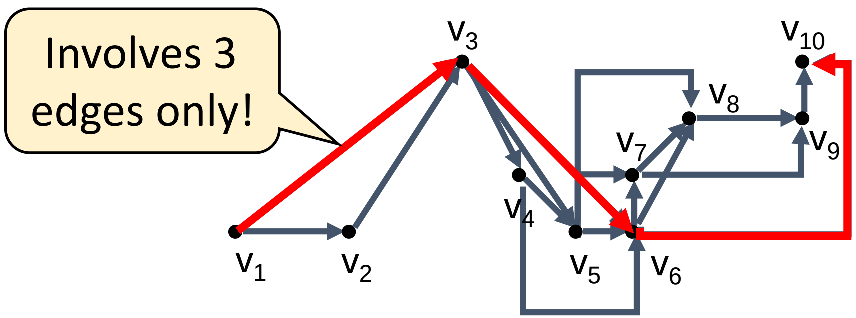
Non-learning based Algorithms

Learning based Algorithms

Trajectory Simplification: Non-Learning based Algorithms

Suppose the error tolerance is $\pi/4$





1: Construct a graph
- **Vertex set:** For each position p_i, create a corresponding vertex v_i
- **Edge set:** For each pair of p_i and p_j ($i<j$), create an edge (v_i, v_j) if
  -- the angle between segment p_i-p_j and segment p_k-p_{k+1} is at most the error tolerance for k = i, i+1, ..., j-1

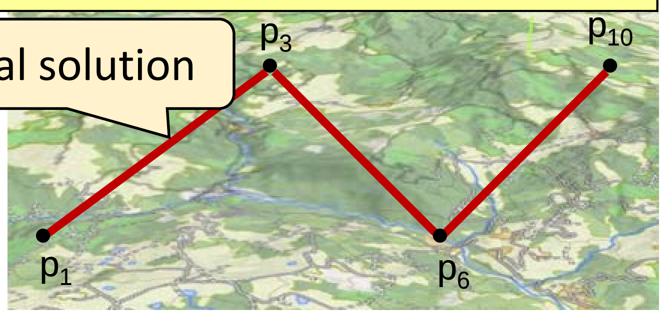
2: Find the path from v_1 to v_{10} with the fewest edges;

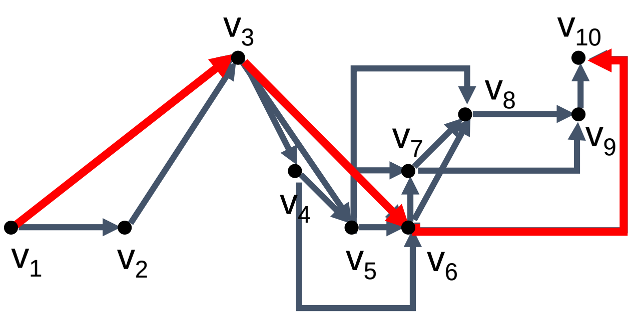
3: Drop the positions with corresponding vertices not involved in the path;

Trajectory Simplification: Non-Learning based Algorithms

Suppose the error tolerance is $\pi/4$

Optimal solution





1: Construct a graph

- **Vertex set:** For each position p_i, create a corresponding vertex v_i
- **Edge set:** For each pair of p_i and p_j ($i<j$), create an edge (v_i, v_j) if
  -- the angle between segment p_i-p_j and segment p_k-p_{k+1} is at most the error tolerance for $k = i, i+1, ..., j-1$

2: Find the path from v_1 to v_{10} with the fewest edges;

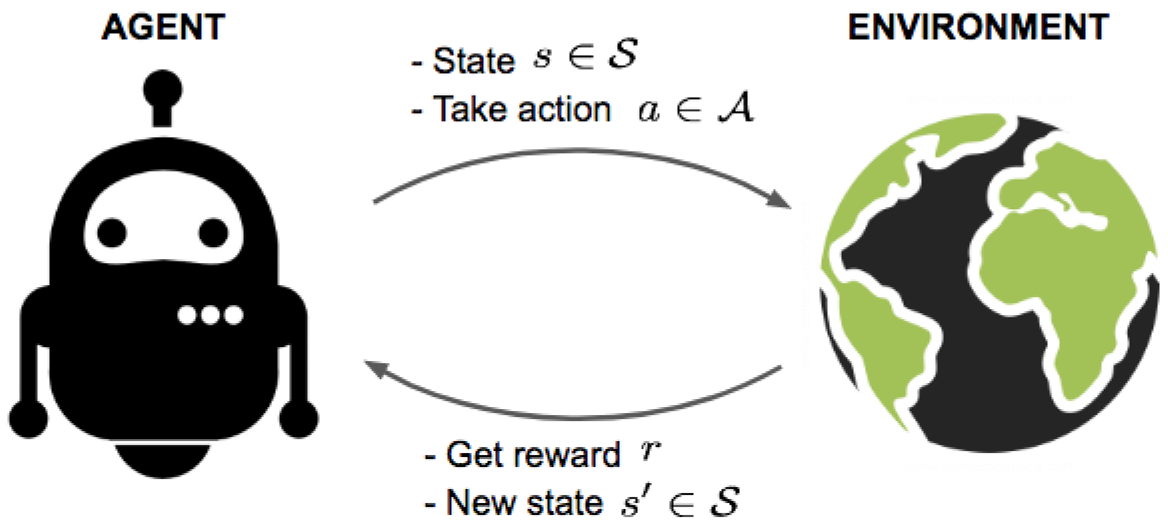
3: Drop the positions with corresponding vertices not involved in the path;

Trajectory Simplification: Algorithms

Non-learning based Algorithms

Learning based Algorithms

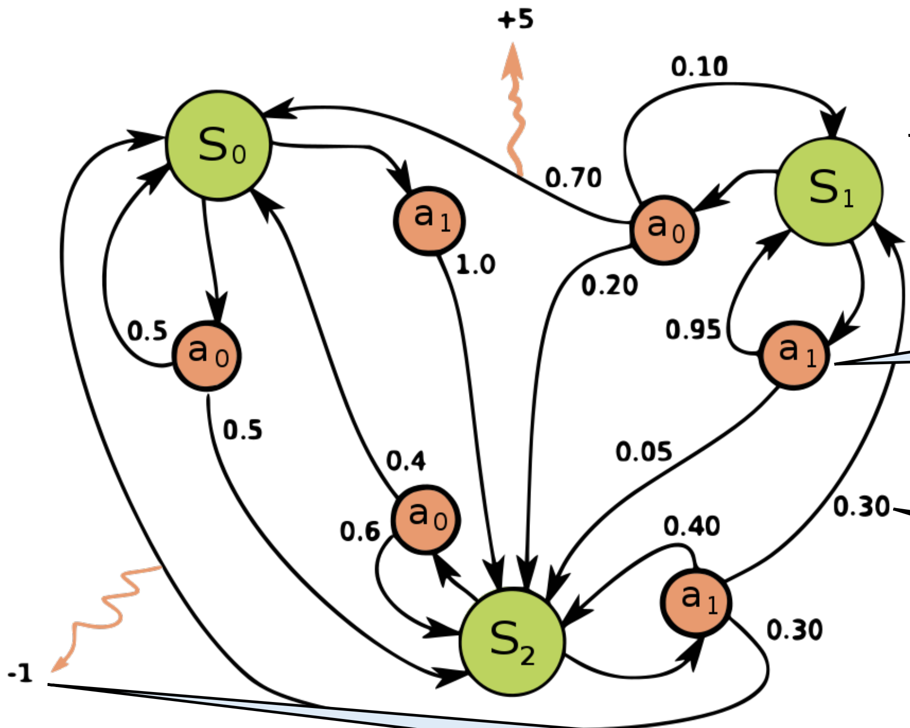
Reinforcement Learning Background



Markov Decision Process (MDP)

https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Reinforcement Learning Background



1. state

2. action

3. Transition probability

4. Transition reward

Photo source: https://en.wikipedia.org/wiki/Markov_decision_process

Reinforcement Learning Background

Aim of Reinforcement Learning (RL):

Find a **policy** $\boldsymbol{\pi}$ with the maximum cumulative random rewards

Reinforcement Learning Background

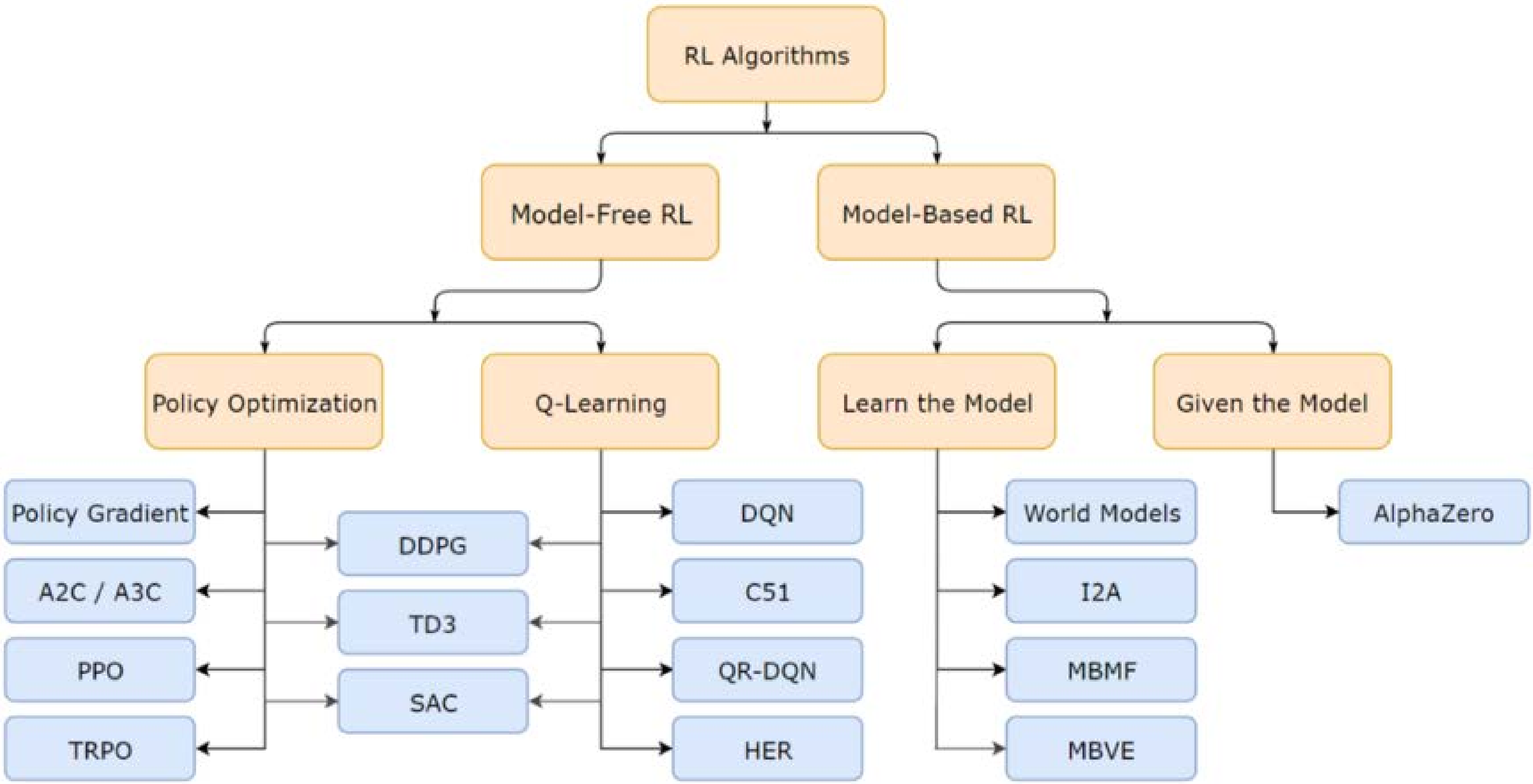
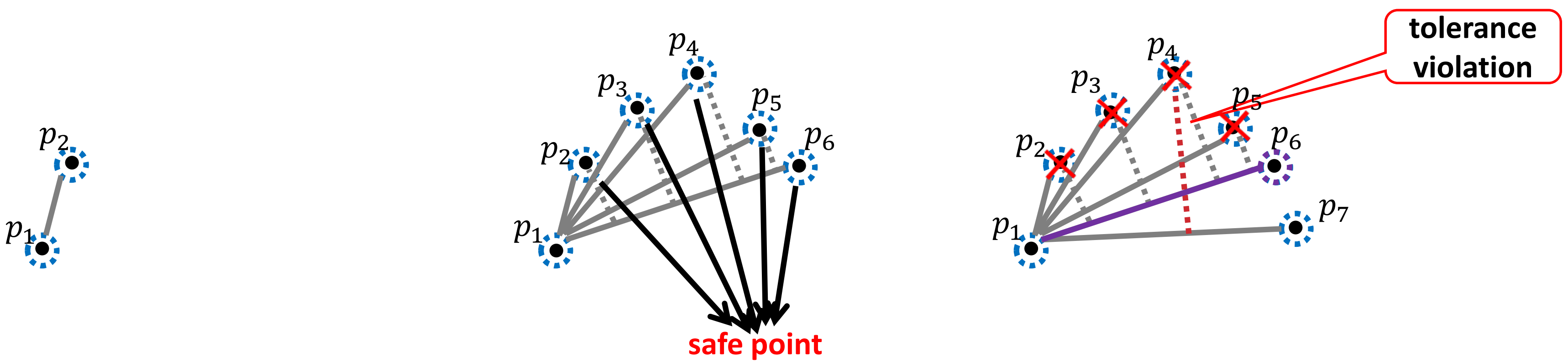


Photo source: https://smartlabai.medium.com/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc

Learning based Algorithms

Step 1	Step 2	Step 3
Open window	Expand window	Re-open window

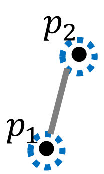


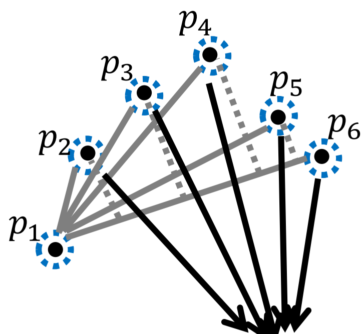
Learning based Algorithms

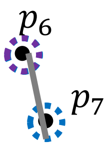
Step 1
Open window

Step 2
Expand window

Step 3
Re-open window







Learning based Algorithms

MDP for Expanding a Window (Agent-E):

- State s^e: {distance ratio, #points in the window}
- Action a^e: expand J points
- Transition: (1) mark safe and continue if bounded;
  (2) call Agent-R if unbounded
- Reward: Determined by Agent-R

Agent-E and Agent-R cooperate for the same objective

MDP for Re-Opening a Window (Agent-R):

- State s^r: define on the last K safe points, and each safe point has {error, #points}
- Action a^r: choose a safe point and drop the points before it
- Transitions: from s^r to $s^{r\prime}$
- Reward: #dropped points (⟺ objective of traj. simplification)

MDP for Expanding a Window (Agent-E):

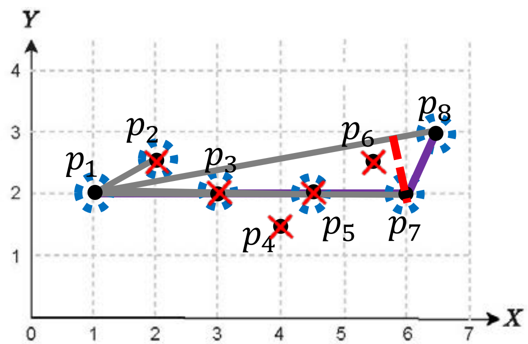
- State s^e: {distance ratio, #points in the window}
- Action a^e: expand J points
- Transition: (1) mark safe and continue if bounded;
  (2) call Agent-R if unbounded
- Reward: Determined by Agent-R

MDP for Re-Opening a Window (Agent-R):

- State s^r: define on the last K safe points, and each safe point has {error, #points}
- Action a^r: choose a safe point and drop the points before it
- Transitions: from s^r to $s^{r\prime}$
- Reward: #dropped points (⟺ objective of traj. simplification)

DQN Network

Initial: L=1, R=2, J=2, K=2, εt=1.0

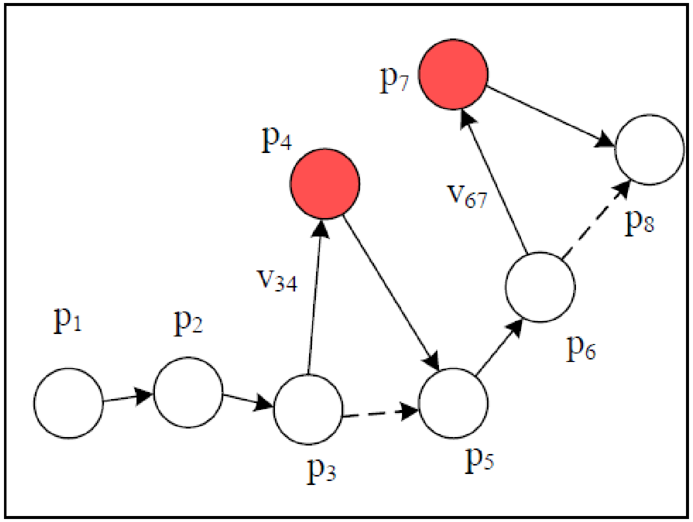


W[L, R]	Error	Safe points	Agent	State	Action
W[1, 2]	0.0 < εt	<P2>	E	s_1^e = {1.0, 2}	Expand to P3
W[1, 3]	0.5 < εt	<P2, P3>	E	s_2^e = {1.118, 3}	Expand to P5
W[1, 5]	0.5 < εt	<P2, P3, P5>	E	s_3^e = {1.160, 5}	Expand to P7
W[1, 7]	0.5 < εt	<P2, P3, P5, P7>	E	s_4^e = {1.177, 7}	Expand to P8
W[1, 8]	1.029 > εt	<P2, P3, P5, P7>	R	s_1^r = {(0.5, 5), (0.5, 7)}	Reopen at P7
W[7, 8]	0.0 < εt	<P8>	-	-	-
Output: T'=<P1, P7, P8>					

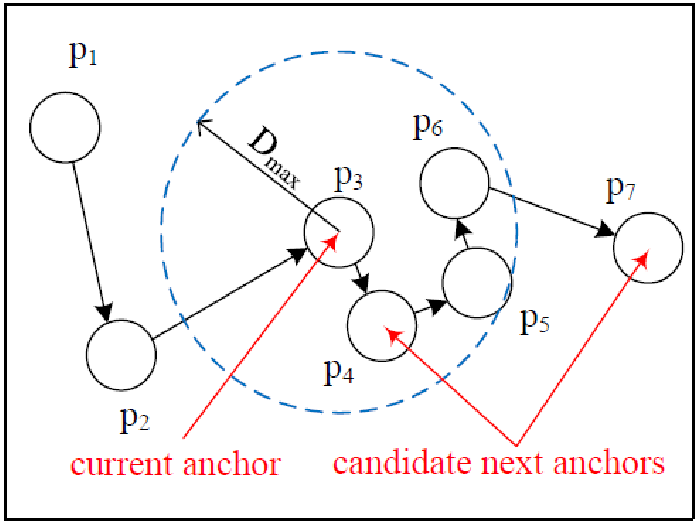
Trajectory Data Preprocessing

Map Matching	**Trajectory Simplification**	**Others**

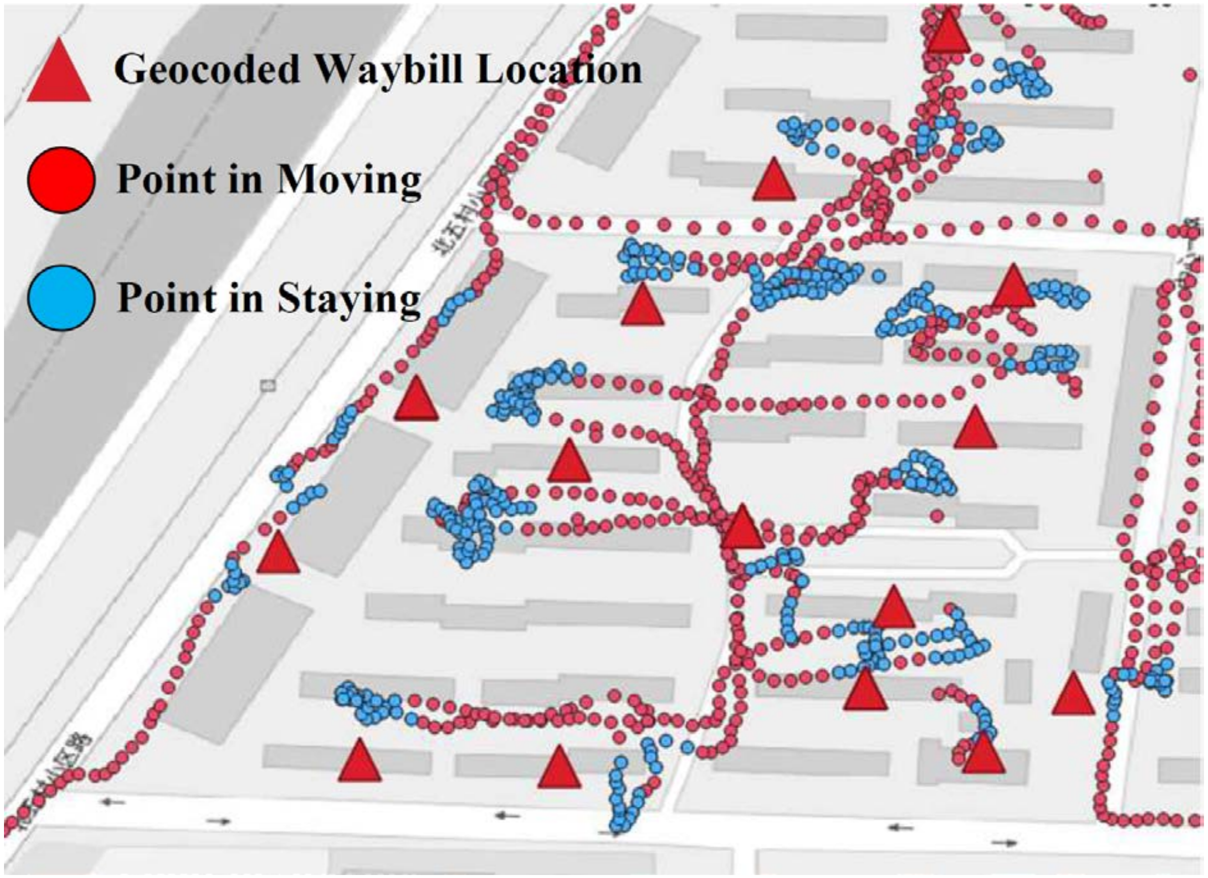
Noise Filtering



Stay Point Detection

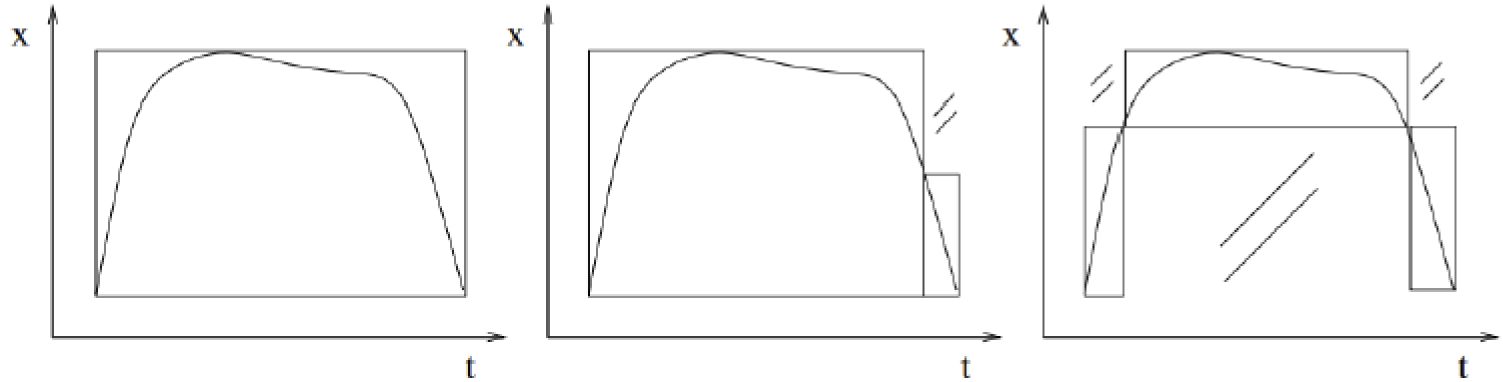


Stay Point Detection

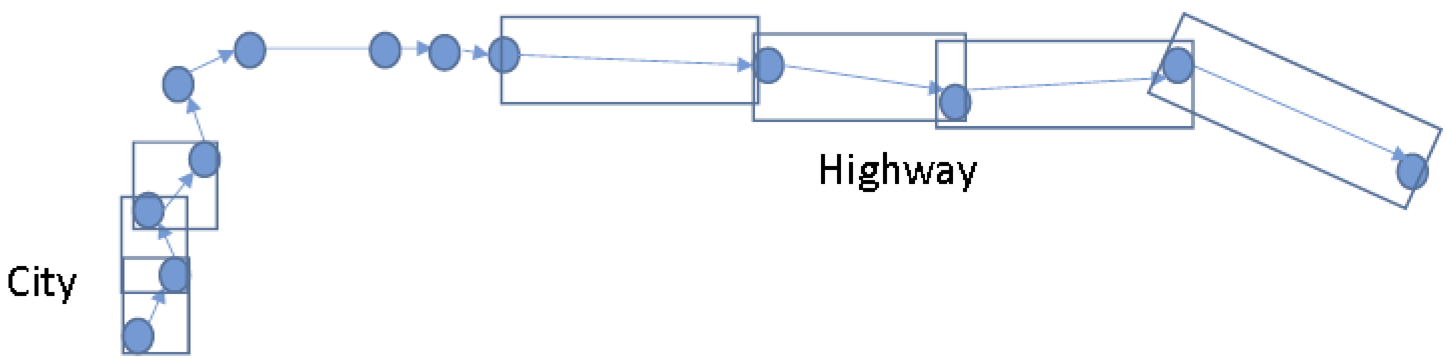


Trajectory Segmentation

- Too large MBR : false hit in Indexing



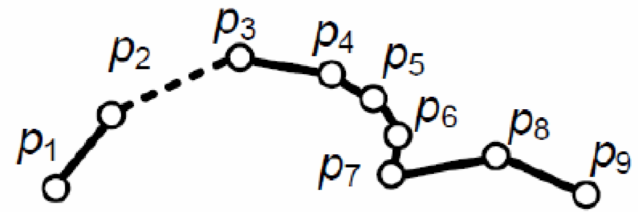
- Semantical segmentation for mining



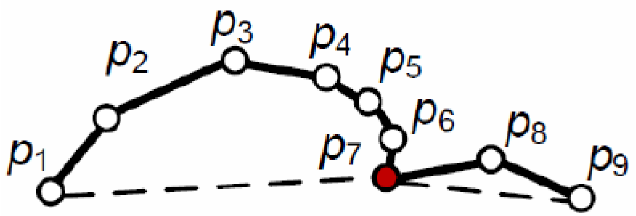
Source: Marios Hadjieleftherius, et. al, Efficient Indexing of Spatiotemporal Objects, EDBT 02

Trajectory Segmentation

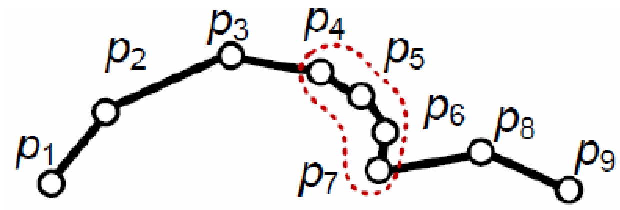
- Time interval segmentation



- Turning points segmentation



- Stay point segmentation



Source: Yu Zheng, Trajectory Data Mining : An Overview, ACM Trans. on Intelligent System and Technology, Sept. 2015

Urban Data Management

Preprocessing

Indexing

Query Processing

Spatial Data

Spatio-Temporal Data

Network Data

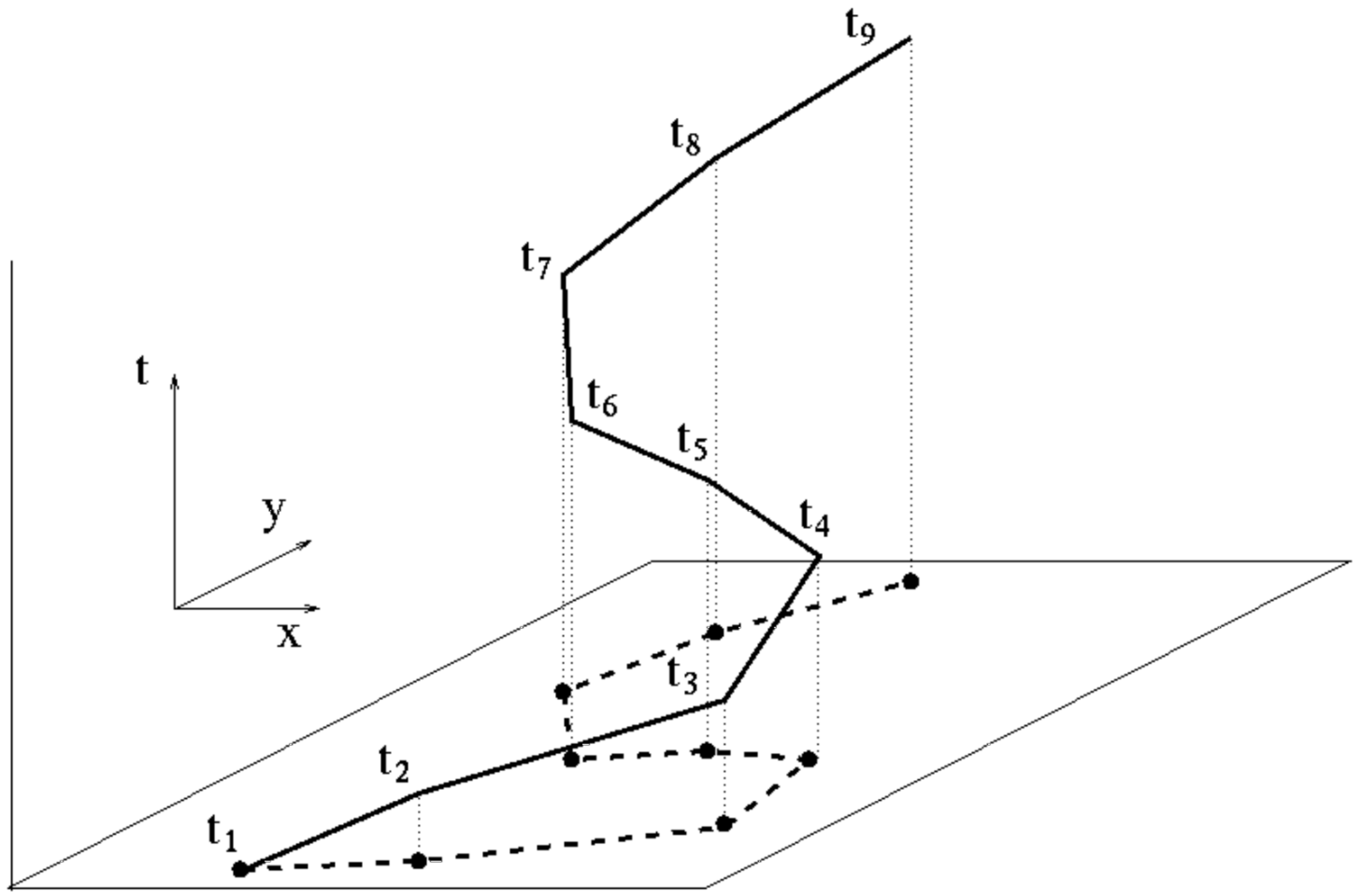
Trajectory Data: Indexing

3D R-Tree

Multi-version R-Tree

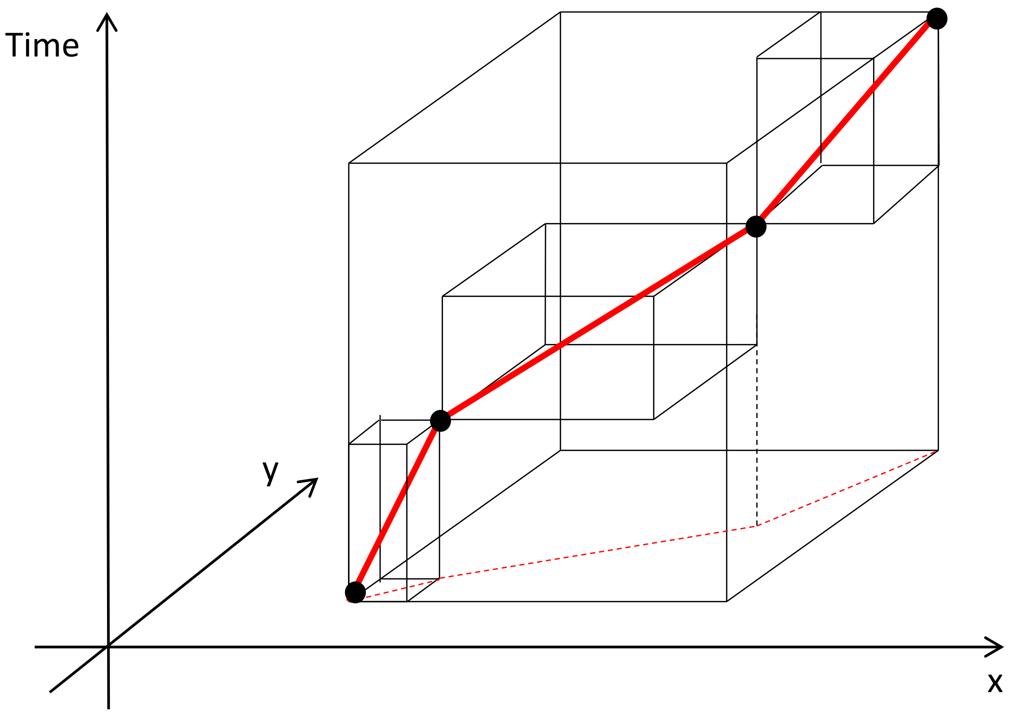
CSE-Tree

A Trajectory in 3D Space



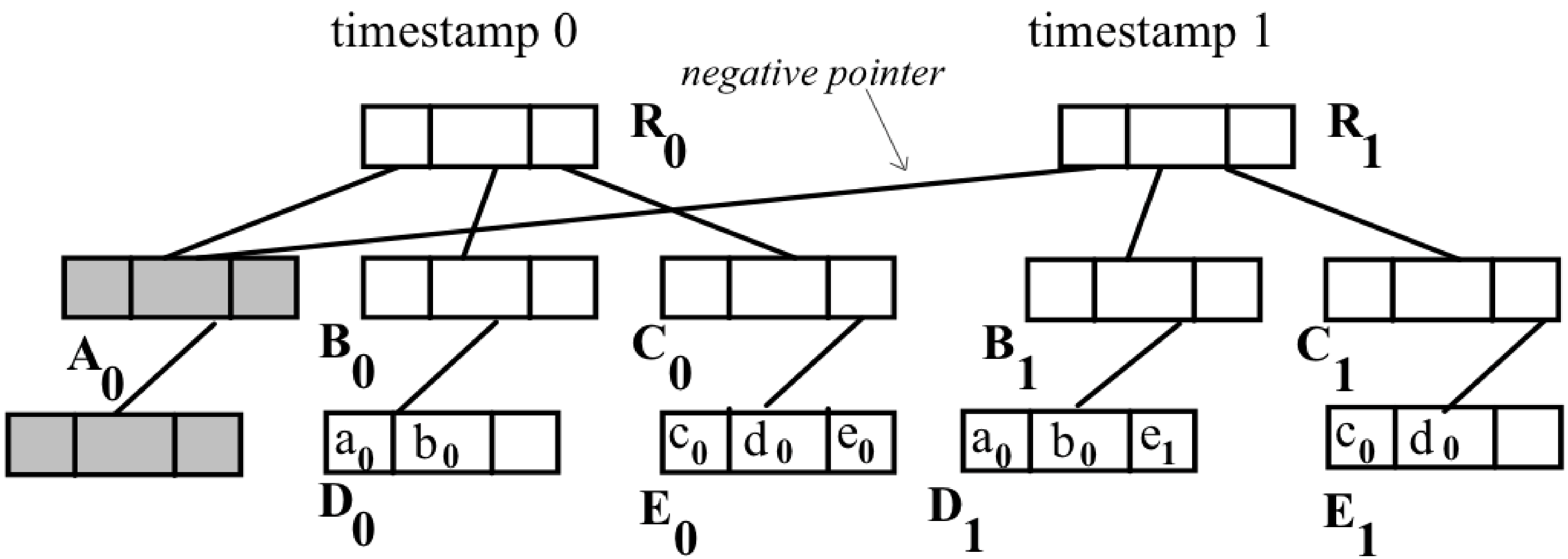
3D R-Tree

1. Treat the time information as a 3rd dimension
2. R-Tree in a 3-dimensional space



Multi-version R-Tree

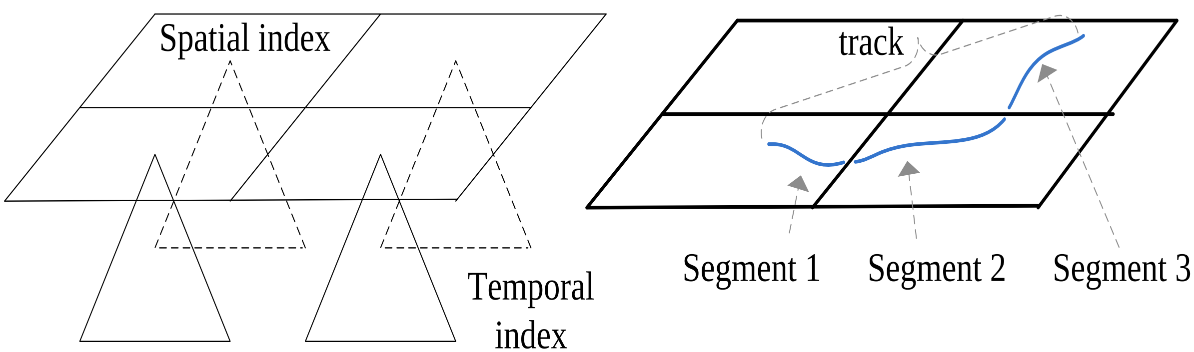
1. Maintain an R-tree for each time instance
2. R-tree nodes that are not changed across consecutive time instances are linked together



HR-tree [Tao2001]

CSE-Tree

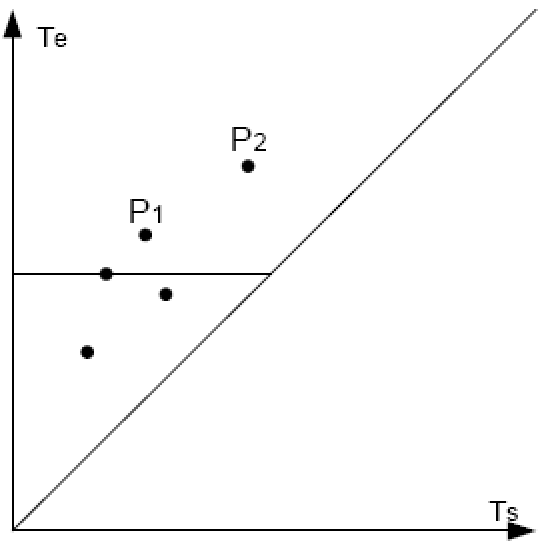
- **Architecture**
  - Partition space into disjoint grids
  - Maintain a temporal index for each grid
  - The temporal index (CSE-Tree) is special

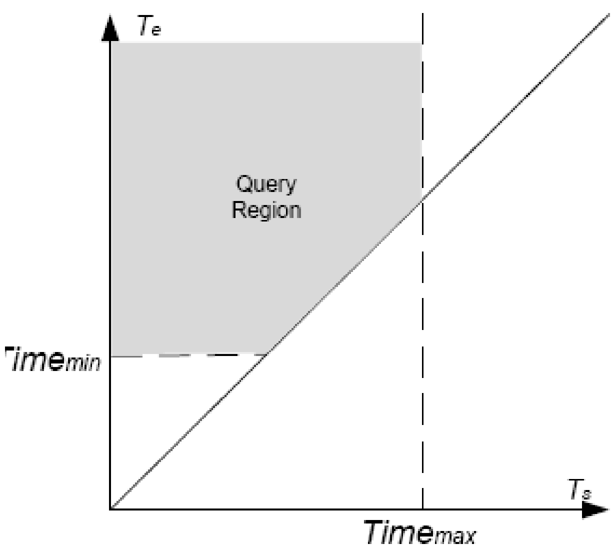


CSE-Tree

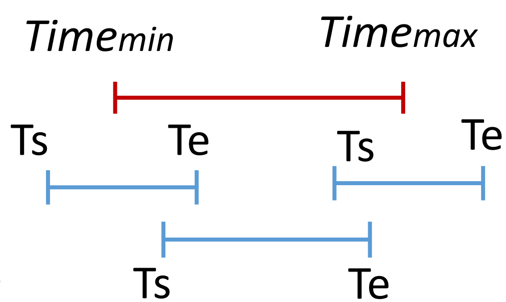
Temporal Index:

- A GPS segment can be represented by a pair (Ts, Te)
- A point on two dimensional plane
- A temporal query is a time span ($Time_{min}$, $Time_{max}$)





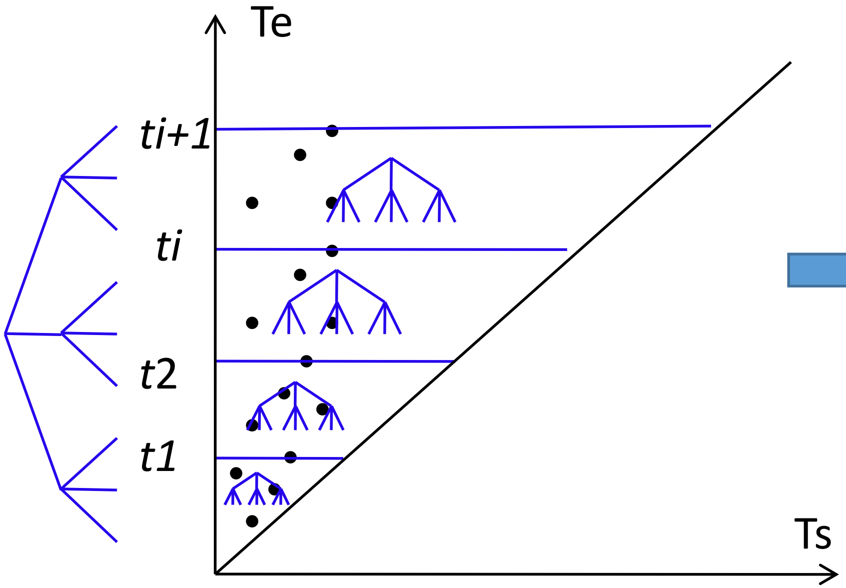
$$\begin{cases} T_e > Time_{min} \\ T_s < Time_{max} \end{cases}$$

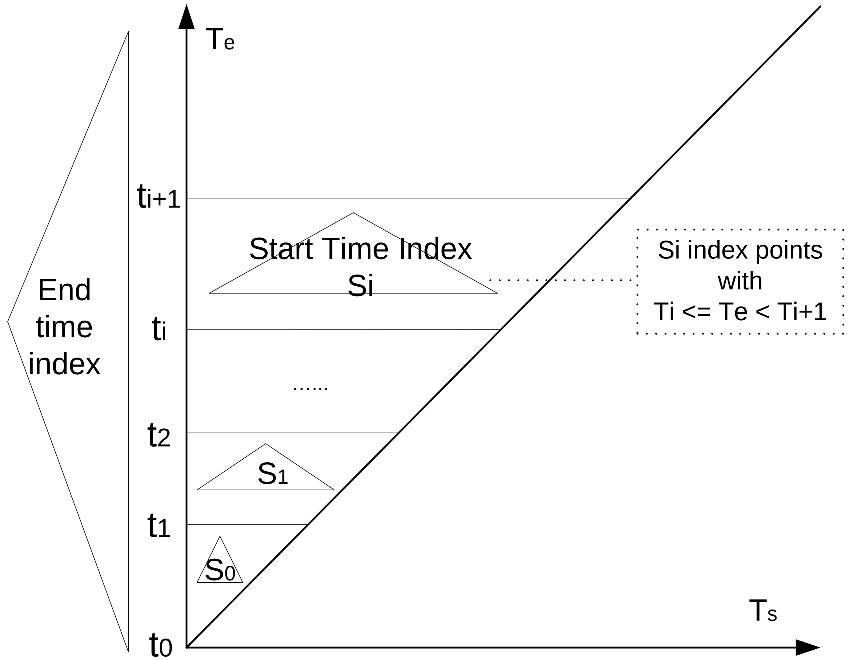


CSE-Tree

Structure

- Partition the points into groups by Te
- Build a *start time index* (B+ Tree) to index points of each group
- Build an end time index (B+ Tree) to index groups

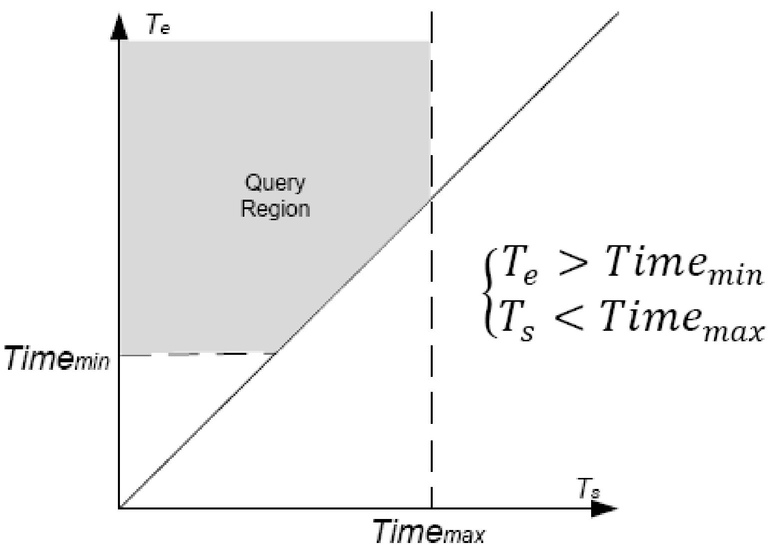


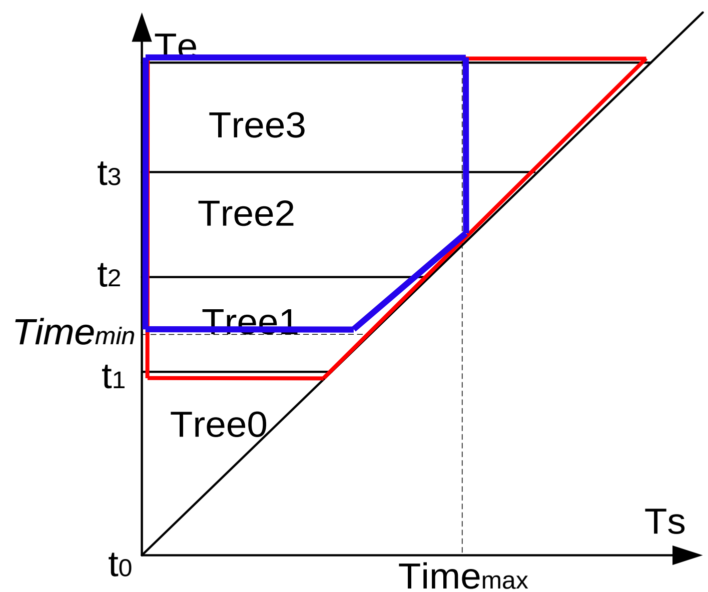


CSE-Tree

Search operation:

- *Te> Time_{min}*: Search End Time index to get the corresponding start time indexes
- *Ts< Time_{max}*: Look up each start time index candidate to find the correct points





Urban Data Management

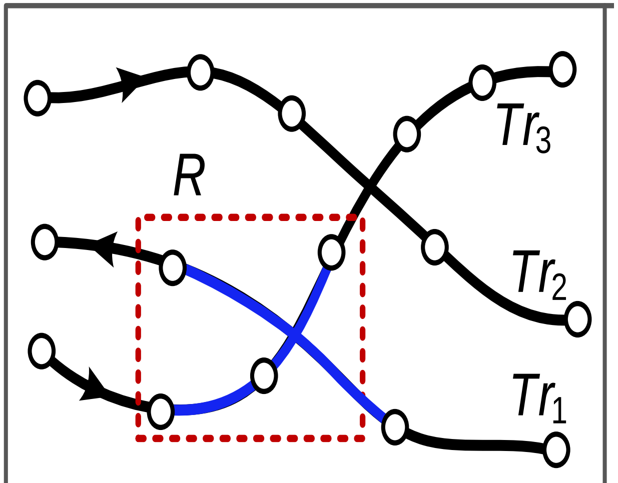
Preprocessing	**Indexing**	**Query Processing**
Spatial Data	**Spatio-Temporal Data**	**Network Data**

This part is for information only and will not be assessed

Range Query

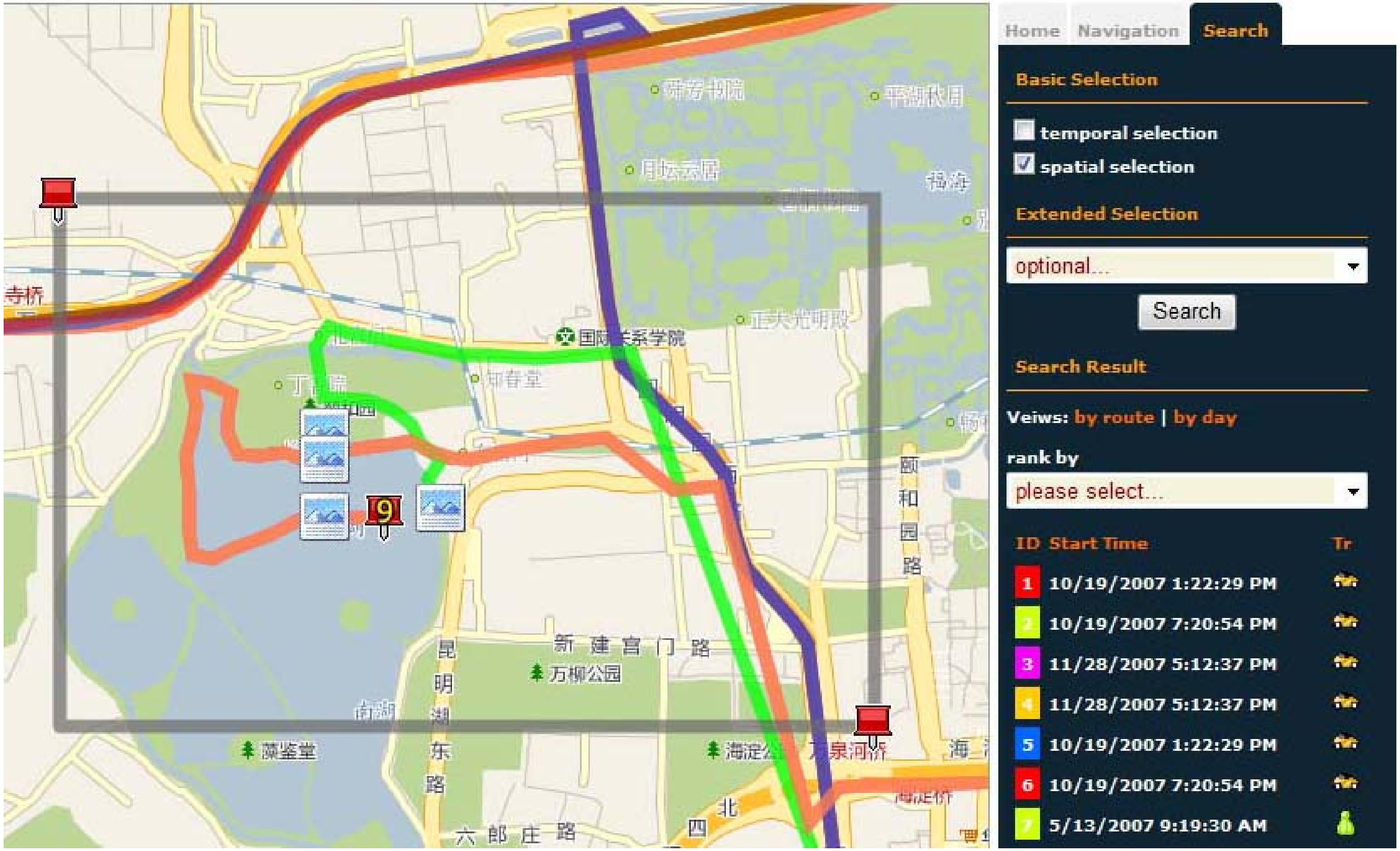
Definition:

Retrieve the trajectories of vehicles passing a given **rectangular region R** and/or between a **time interval**



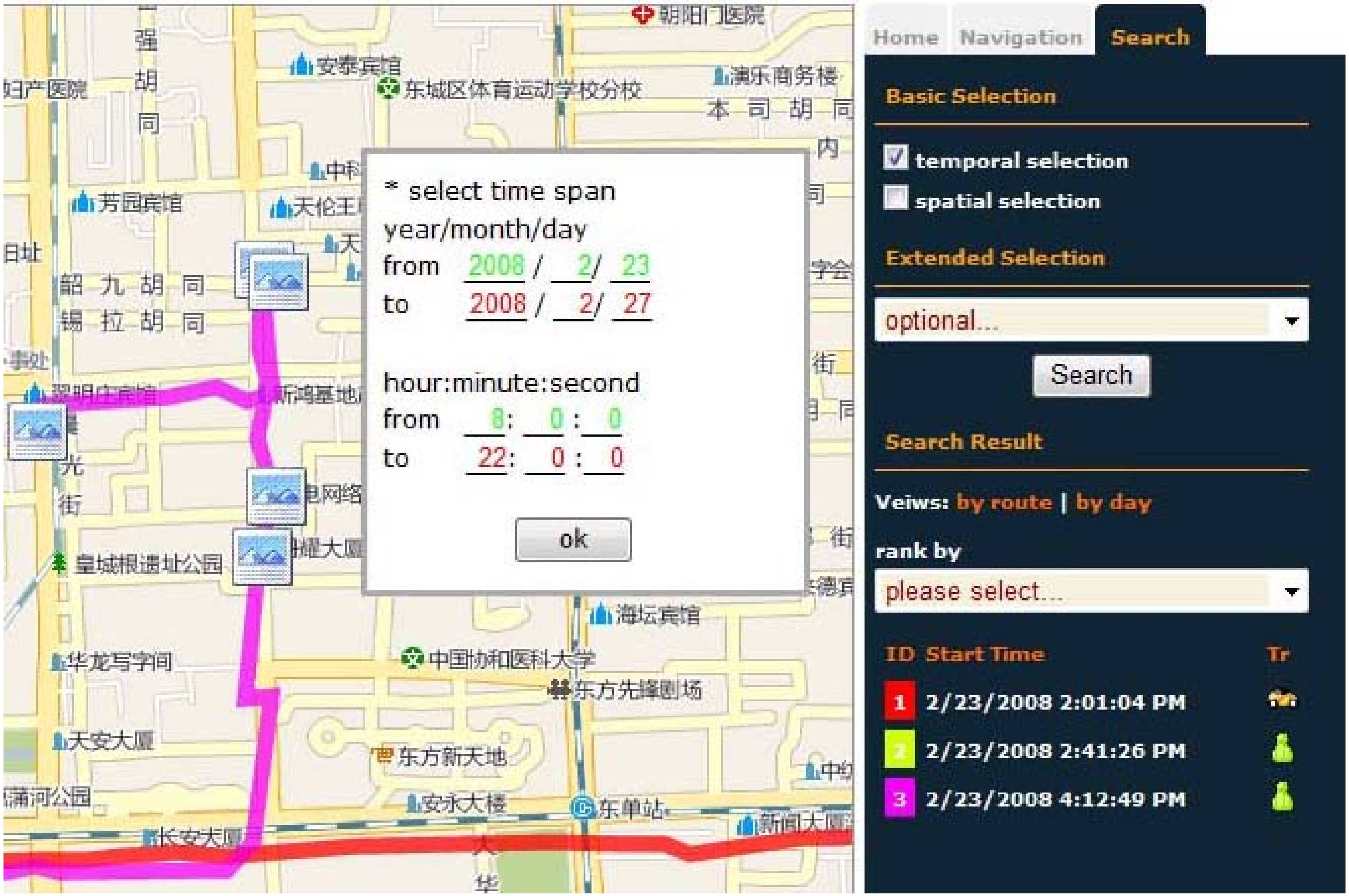
A) Range Query

Range Query - Spatial



Source: Yu Zheng, Trajectory Data Mining : An Overview, ACM Trans. on Intelligent System and Technology, Sept. 2015

Range Query - Temporal

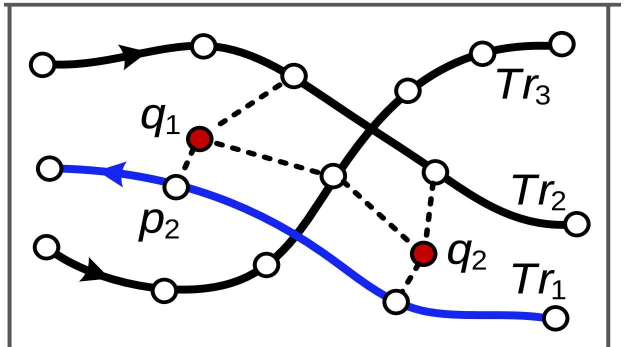


Source: Yu Zheng, Trajectory Data Mining : An Overview, ACM Trans. on Intelligent System and Technology, Sept. 2015

kNN Point Query

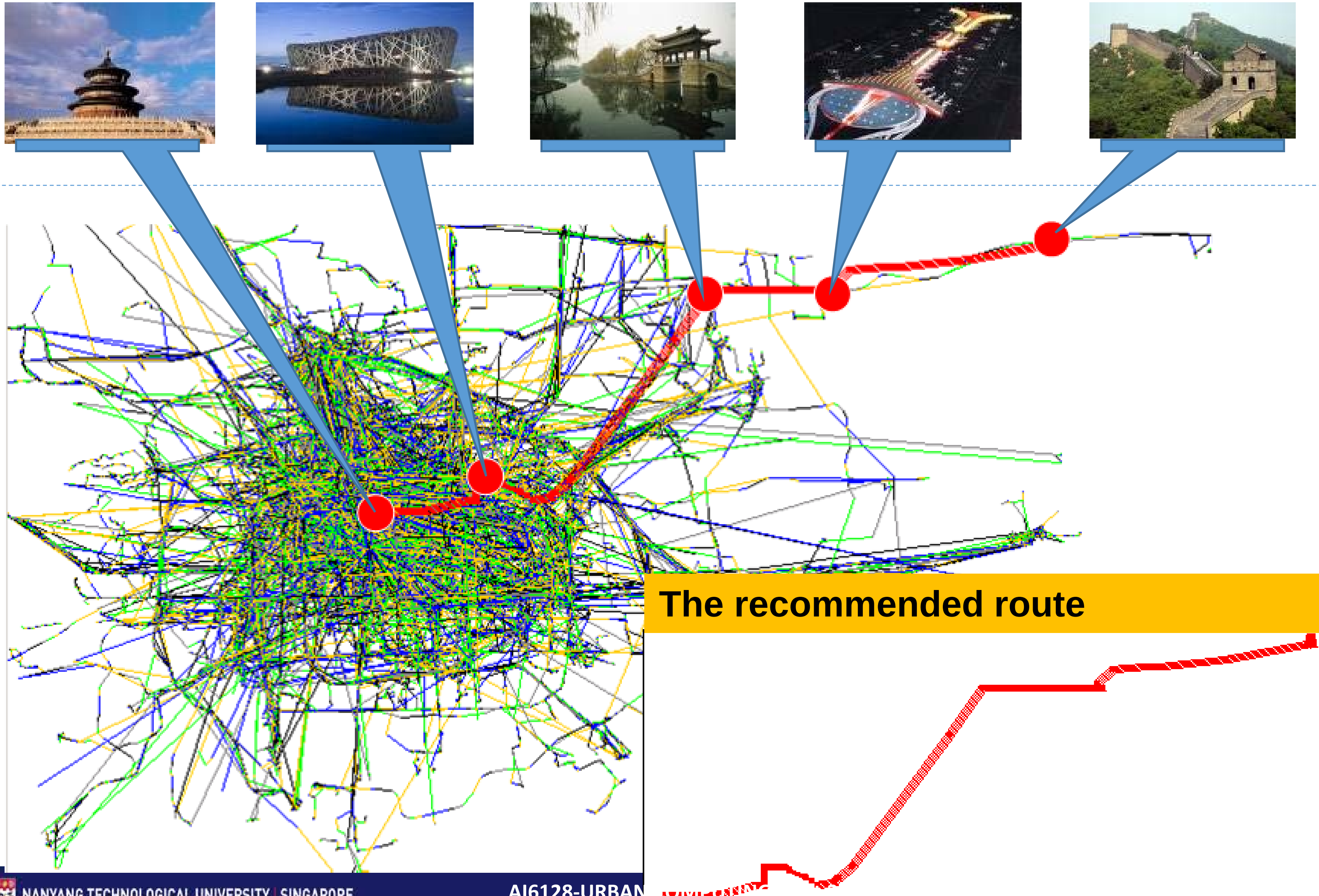
Definition:

Retrieve the trajectories of people with the minimum aggregated distance to a set of **query points**



B) KNN Point Query

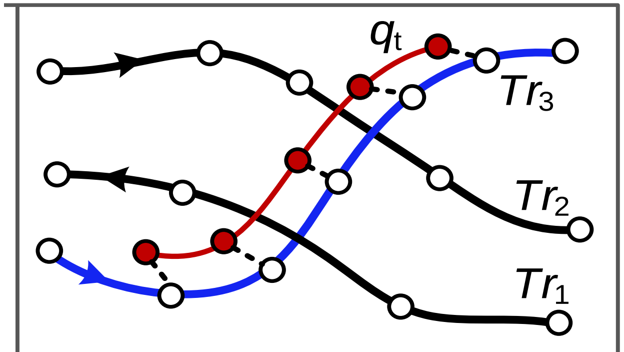
kNN Point Query



kNN Trajectory Query (Trajectory Similarity Search)

Definition:

Retrieve the trajectories of people with the minimum aggregated distance to a **query trajectory**

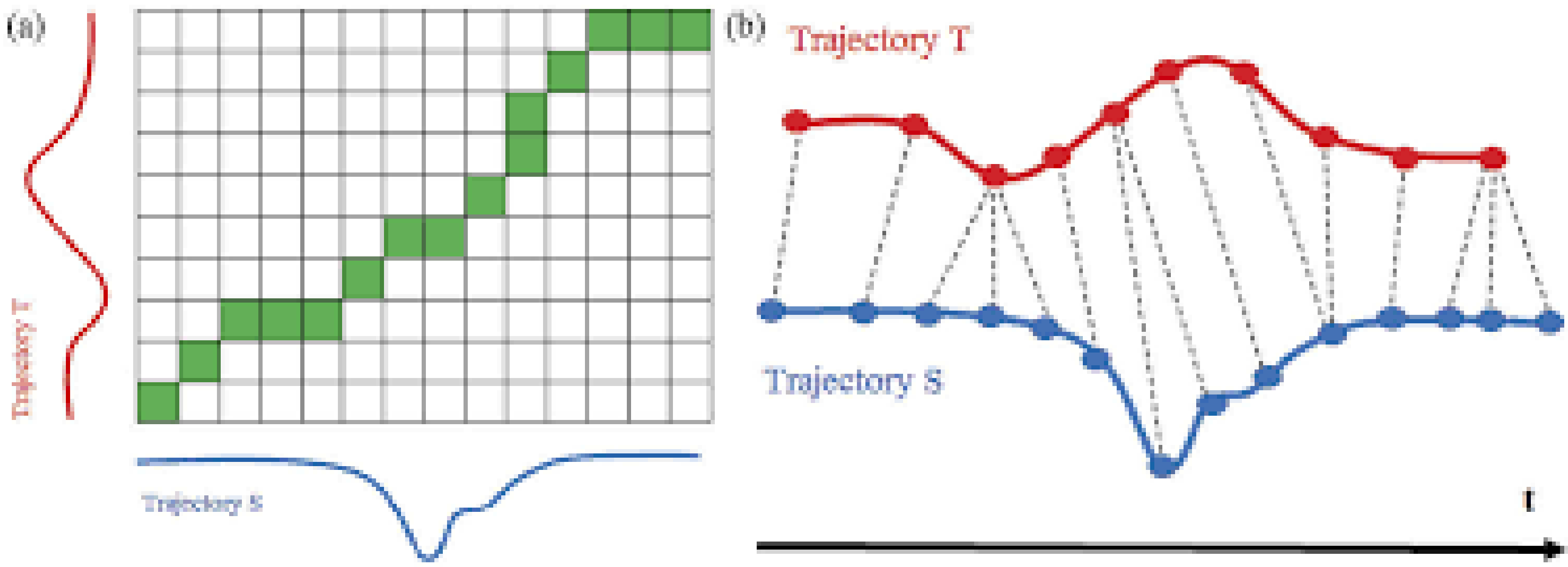


C) KNN Trajectory Query

Trajectory Similarity Measurements

Alignment-based Measurement:

- Pairwise matching
- Quadratic time complexity

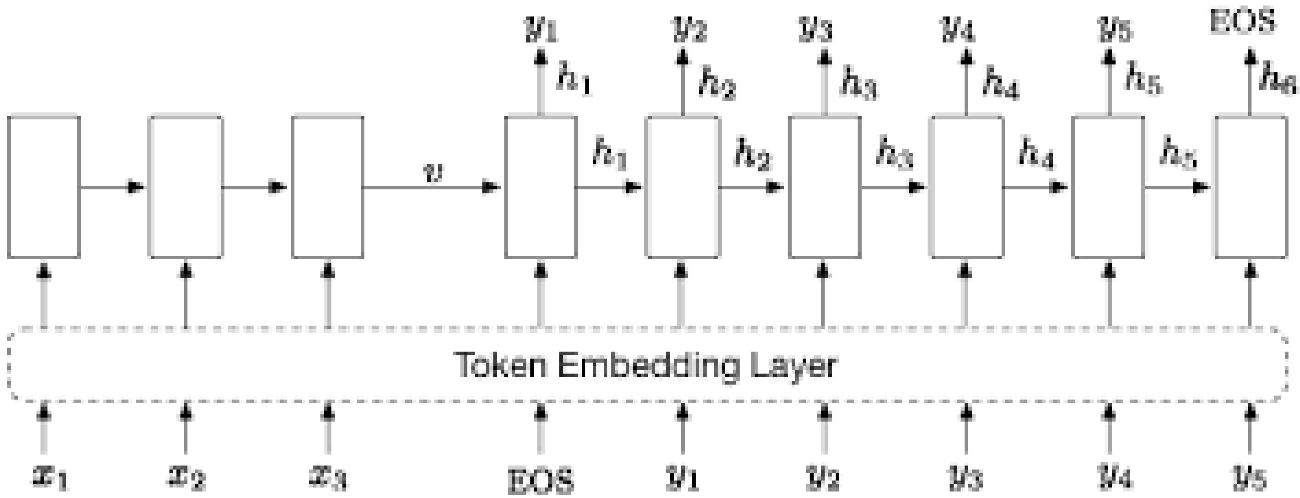


Context-awareness in similarity measures and pattern discoveries of trajectories: a context-based dynamic time warping method, by Mohammad Sharif &Ali Asghar Alesheikh

Trajectory Similarity Measurements

Learning-based Measurement:

- Data-driven (representation learning)
- Linear time complexity

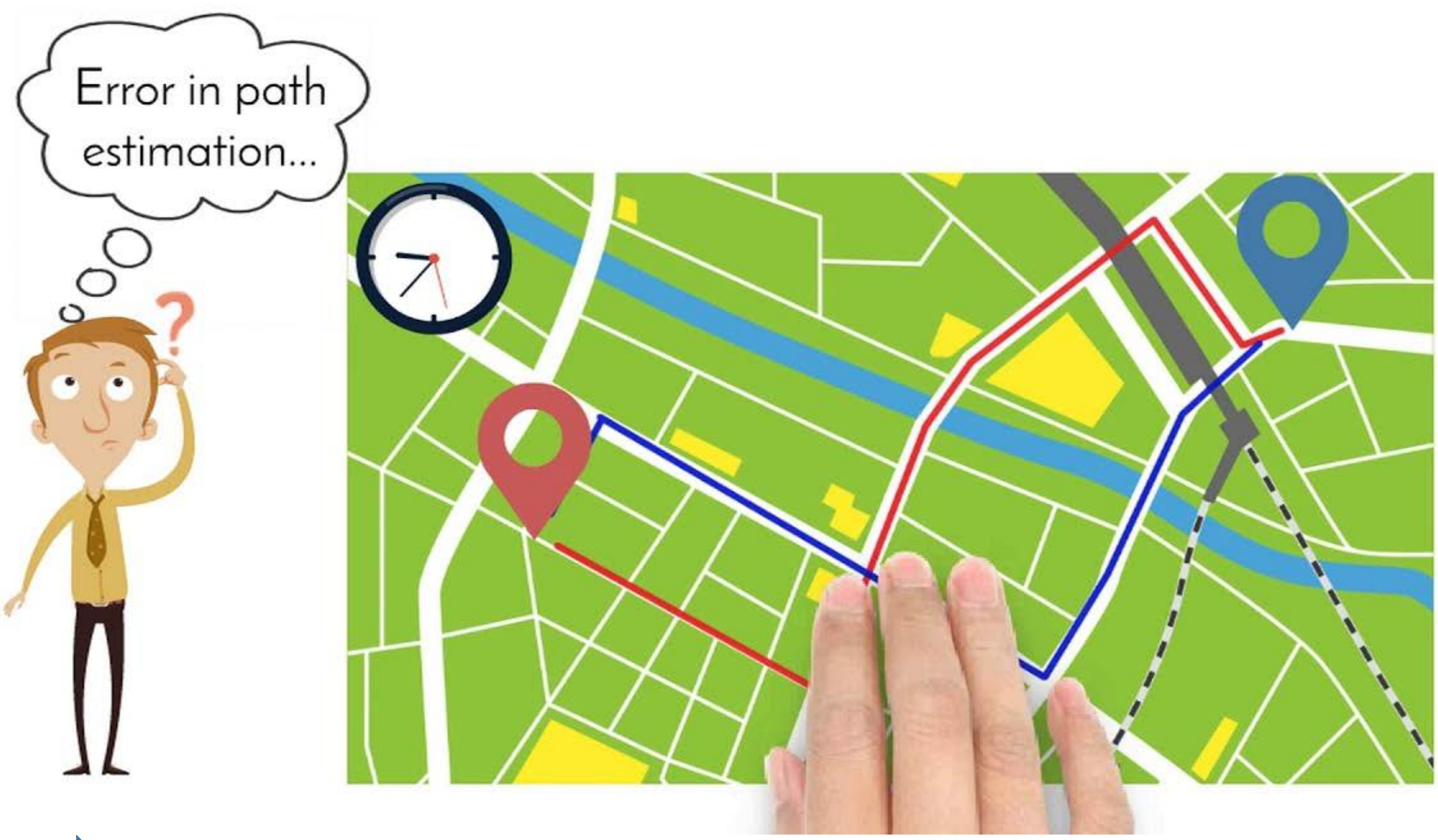


More details will be covered in the guest lectures

ence-to-Sequence Model

Li. et al. Deep Representation Learning for Trajectory Similarity Computation, ICDE'18

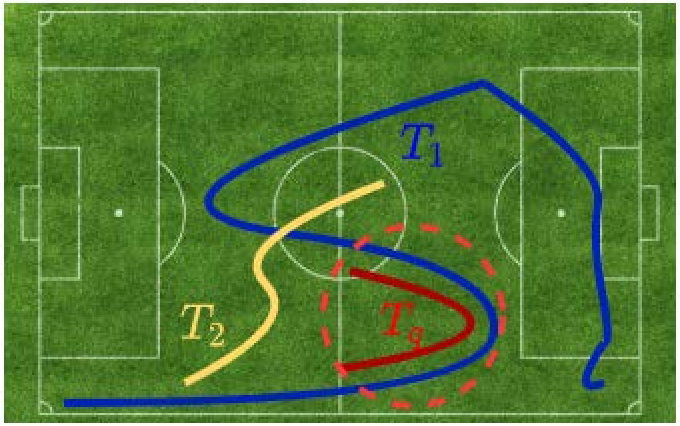
kNN Trajectory Query (Trajectory Similarity Search)



Make use of similar trajectories in the repository

Source: https://www.kdd.org/kdd2018/accepted-papers/view/multi-task-representation-learning-for-travel-time-estimation

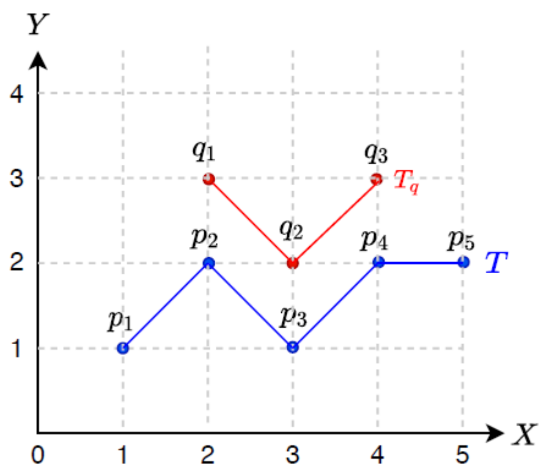
Sub-Trajectory Similarity Search



T_1 and T_2 are **dissimilar** to T_q

T_1 has a **portion** that is **similar** to T_q

Sub-Trajectory Similarity Search



SimSub problem

Sub-Trajectory Similarity Search:
Return a portion of a data trajectory (i.e., a sub-trajectory), which is the most similar to a query trajectory

Tq is a query trajectory and T is a data trajectory:

- Return T[2:4] = <p2, p3, p4>
- General framework using any similarity measurement

Urban Data Management

Preprocessing	**Indexing**	**Query Processing**
Spatial Data	**Spatio-Temporal Data**	**Network Data**

This part will not be tested.

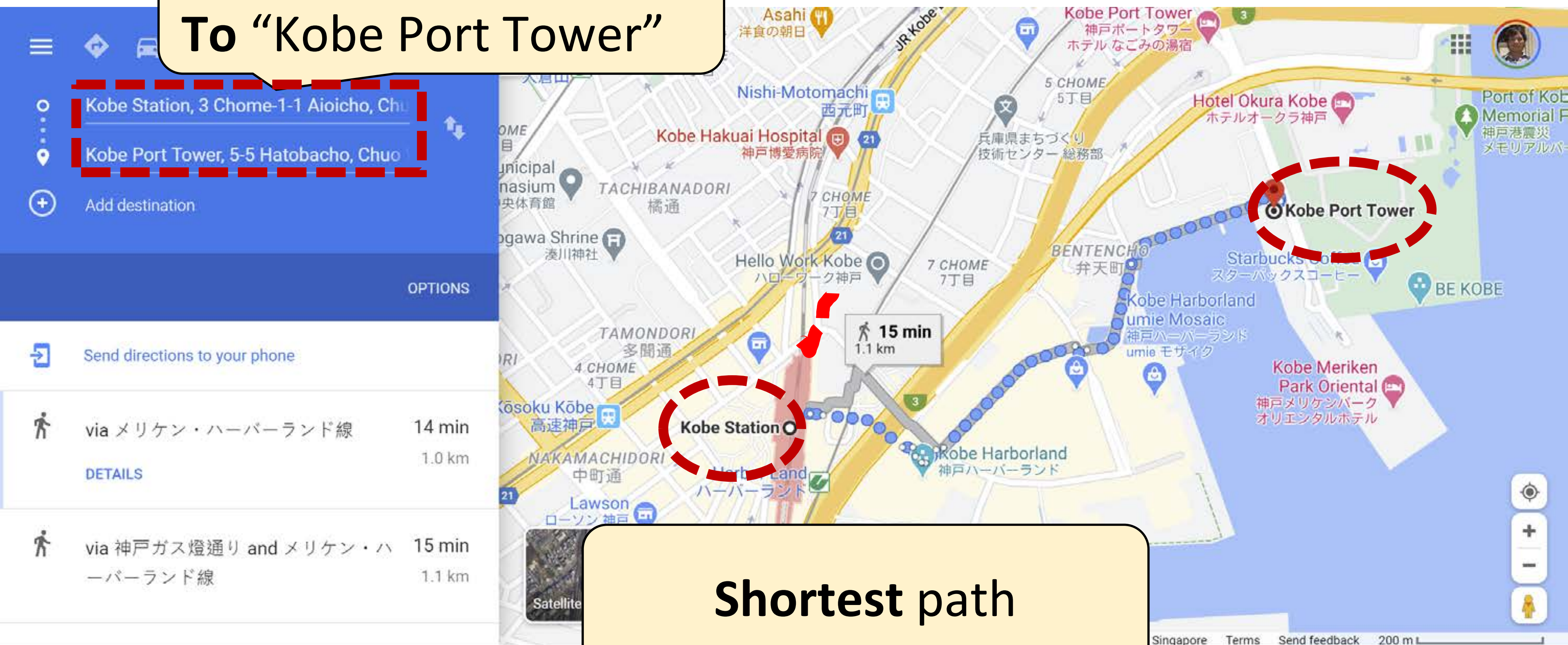
Possible Questions from Bob (1)

I want to go from my hotel to an attraction.

Which path should I follow?

Possible Questions from Bob (1)

From “Kobe Station”
To “Kobe Port Tower”



Shortest path

Possible Questions from Bob (1)

Question:

What's the **shortest / fastest** path from a source to a destination?

Popularity?
Scenic value?
Appeal of a route?
...

Possible Questions from Bob (2)

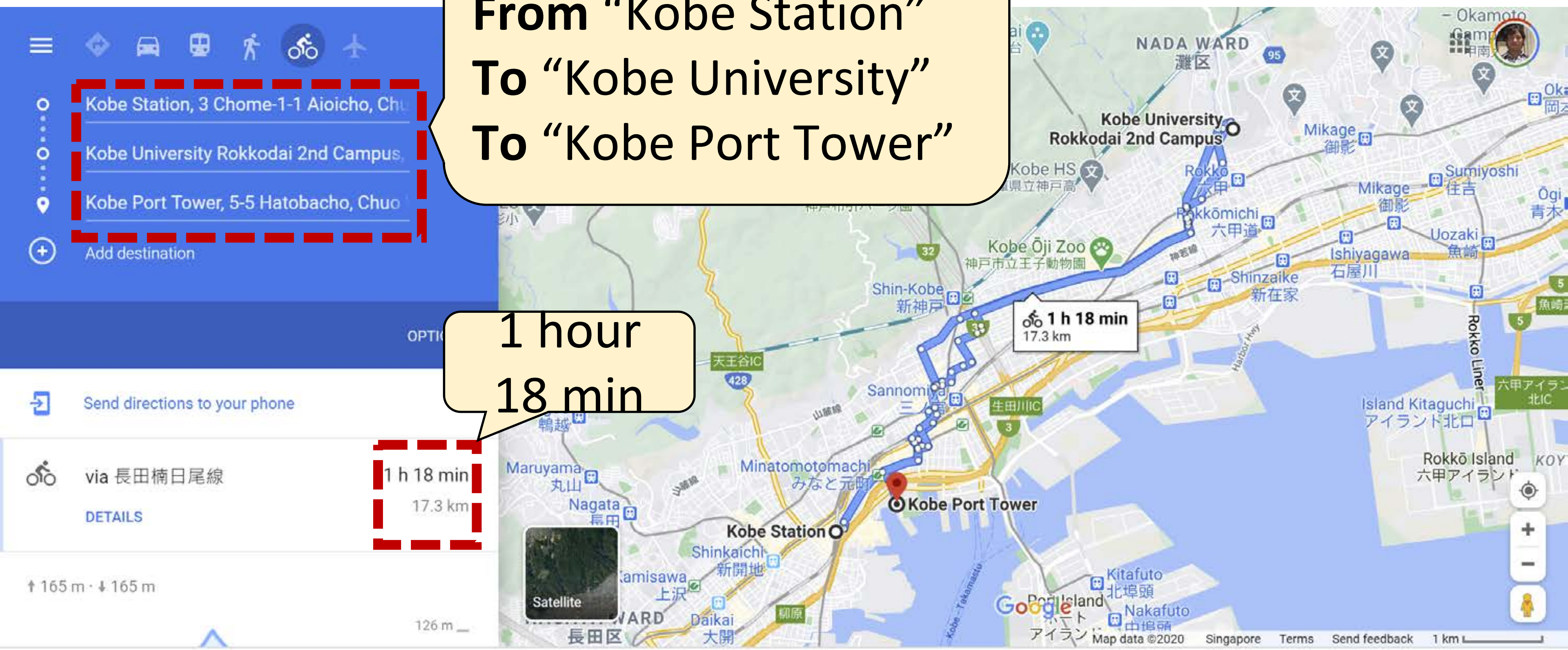
I want to visit attractions A, B, C, etc.

Can you plan a path to visit them all?

Possible Questions from Bob (2)

From “Kobe Station”
To “Kobe University”
To “Kobe Port Tower”

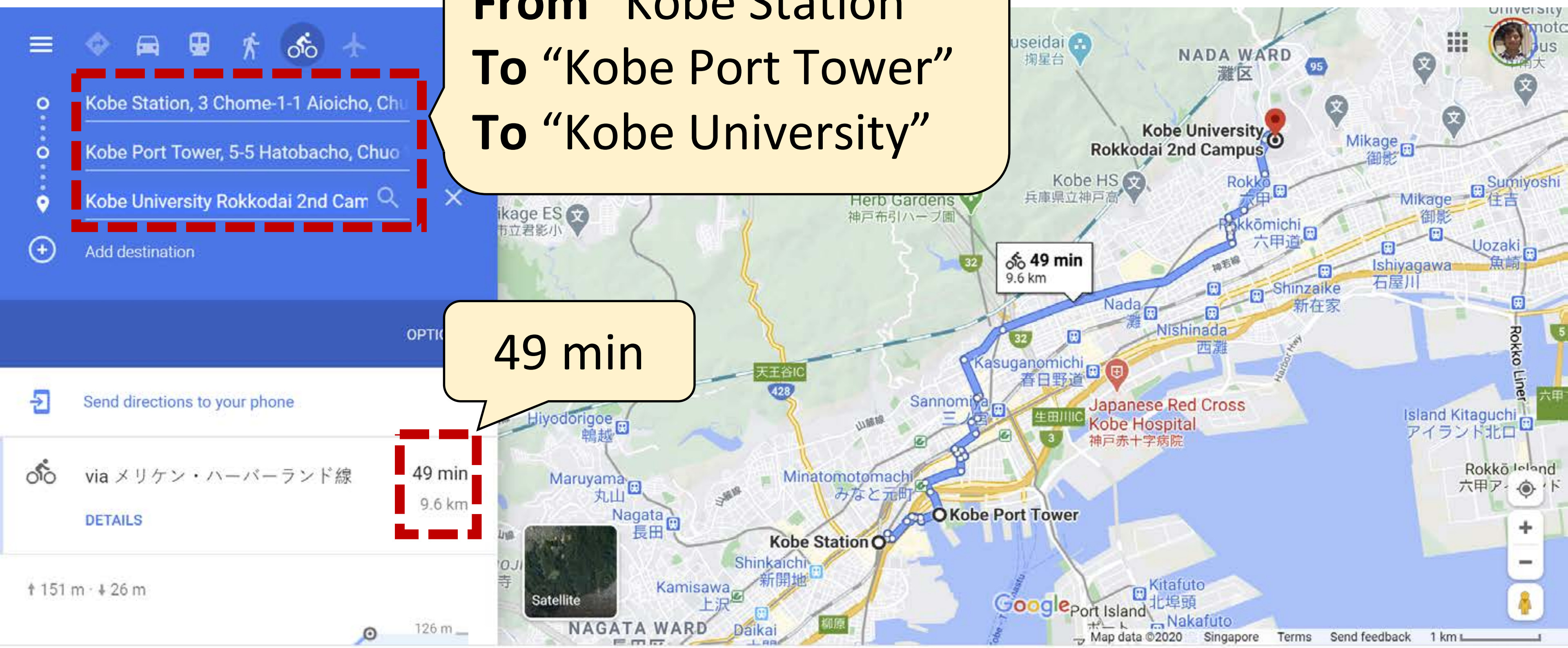
1 hour 18 min



Possible Questions from Bob (2)

From “Kobe Station”
To “Kobe Port Tower”
To “Kobe University”

49 min



Possible Questions from Bob (2)

Question:

Can you plan a route to visit some given tourist sites?

Query Processing on Network Data

Shortest-path problem

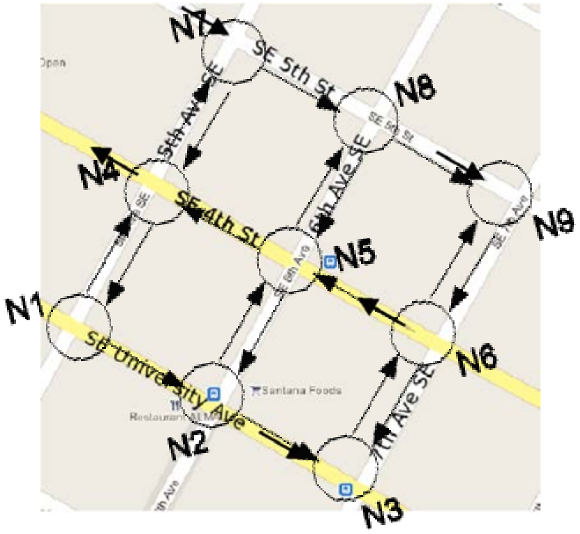
Travelling salesman problem

Road Network

An Example of road network

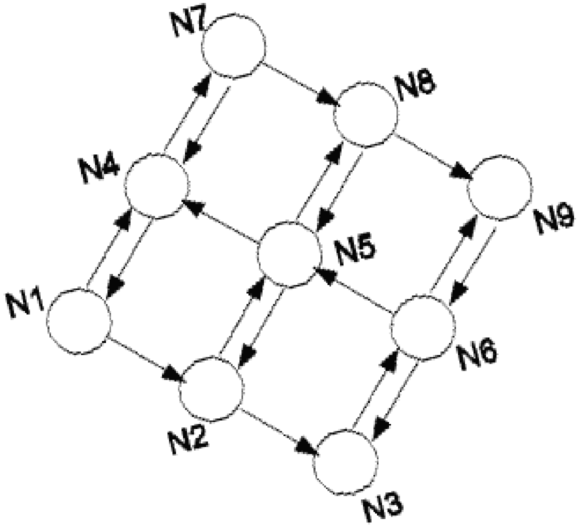
Road Network

Junction -> Vertex
Road segment -> Edge



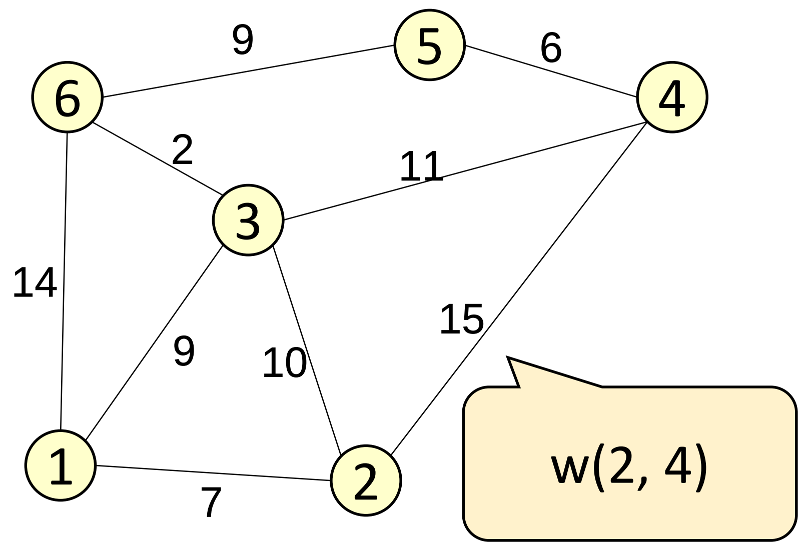
Road Network

Graph structure



Shortest Path Problem

What's the shortest path from 1 to 4?



Dijkstra Algorithm

Dijkstra Algorithm (s -> t):

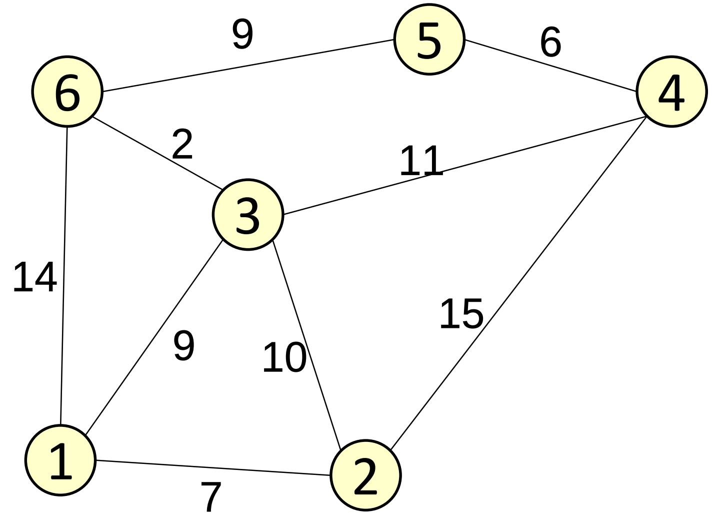
1. Initialize the **dist** of the source s to be 0 and those of others to be ∞; Intialize the **prev** of each vertex as undefined (UD);
2. **Repeat** until the destination t is picked
   - Pick the vertex v with the **smallest dist** & **unpicked**
   - For each **neighbor** u of v, update the dist and prev of u if **the distance from s to u via v** is shorter than the current dist of u;
3. Return the path as indicated by **prev**;

Dijkstra Algorithm

What's the shortest path from 1 to 4?

vertex	1	2	3	4	5	6
dist	0	∞	∞	∞	∞	∞
pre	ND	ND	ND	ND	ND	ND

1. Initialize the **dist** of the source s to be 0 and those of others to be ∞; Intialize the **prev** of each vertex as undefined (UD);

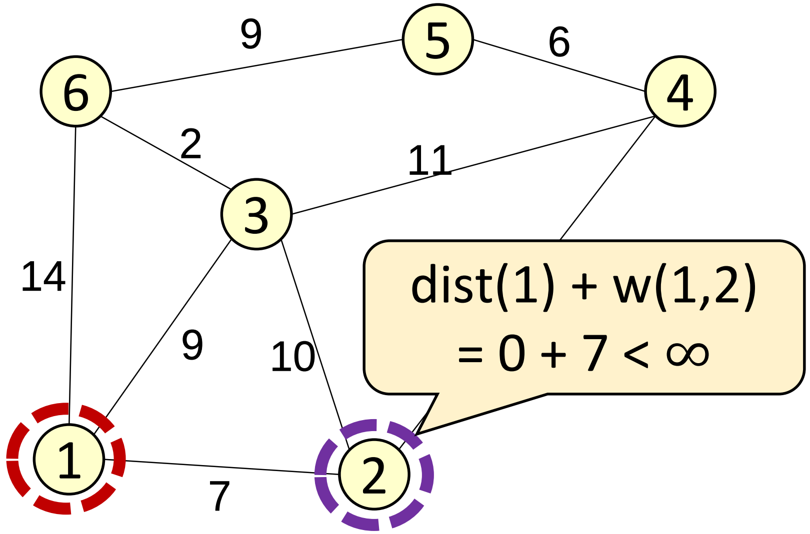


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	∞	∞	∞	∞	∞
pre	ND	ND	ND	ND	ND	ND

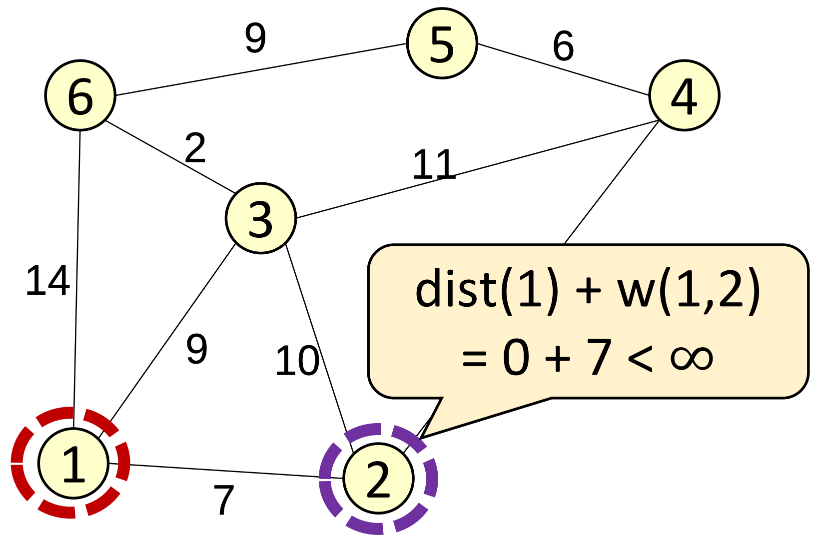


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	7	∞	∞	∞	∞
pre	ND	1	ND	ND	ND	ND

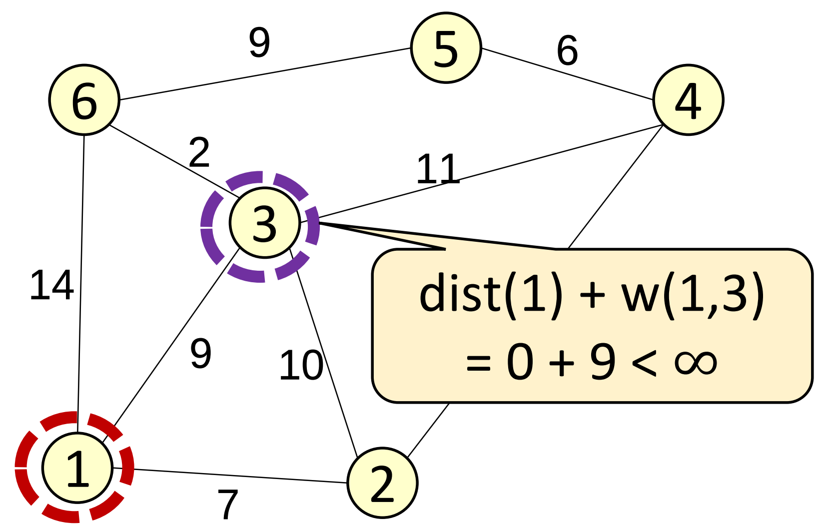


Dijkstra Algorithm

vertex	1	2	3	4	5	6
dist	0	7	∞	∞	∞	∞
pre	ND	1	ND	ND	ND	ND

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

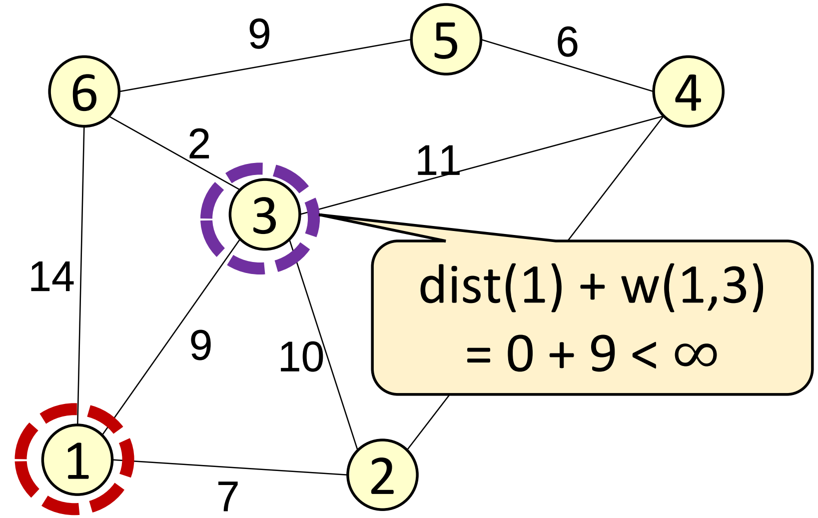


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	∞
pre	ND	1	1	ND	ND	ND



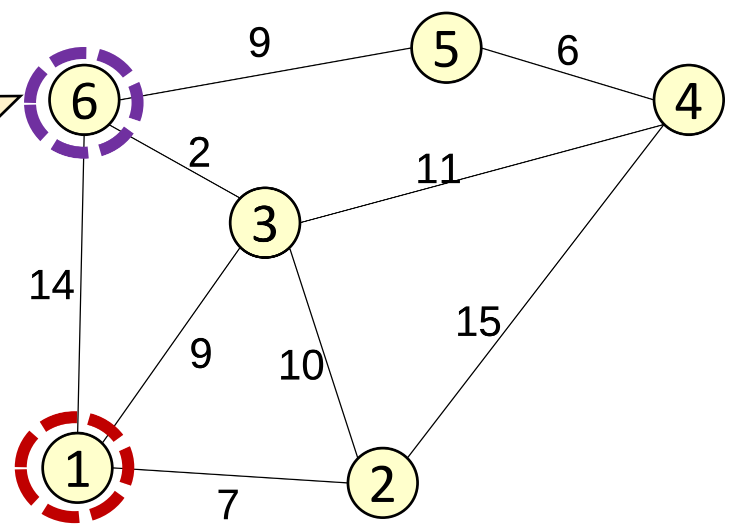
Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex
     the **smallest** dist &
     unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	∞
pre	ND	1	1	ND	ND	ND

dist(1) + w(1,6)
= 0 + 14 < ∞

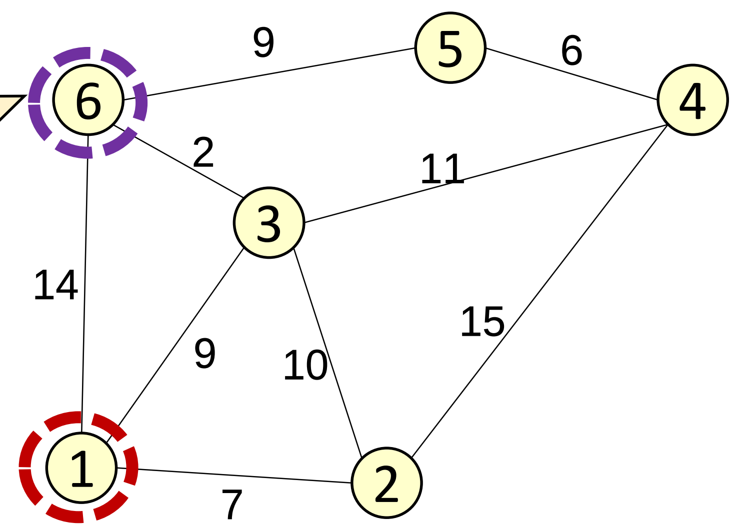


Dijkstra Algorithm

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	14
pre	ND	1	1	ND	ND	1

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;



Dijkstra Algorithm

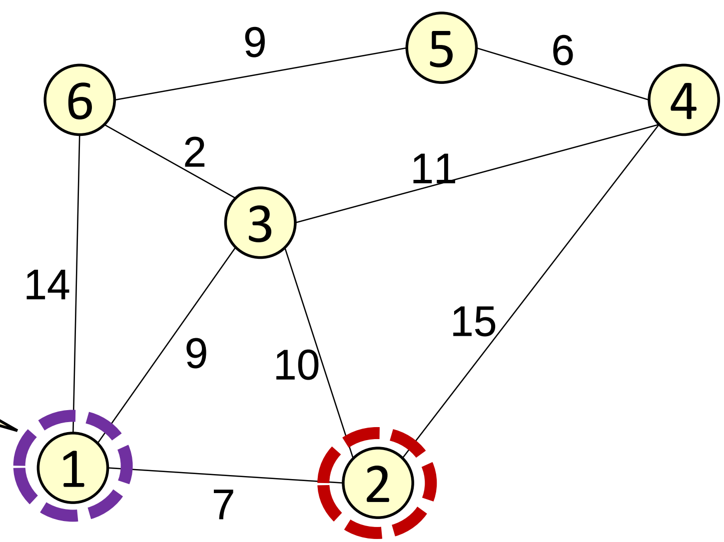
P

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	14
pre	ND	1	1	ND	ND	1

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neig
     update the **d**
     prev of u if the distance from s to u via v is shorter than the current dist of u;

We can skip 1 here. Think why?

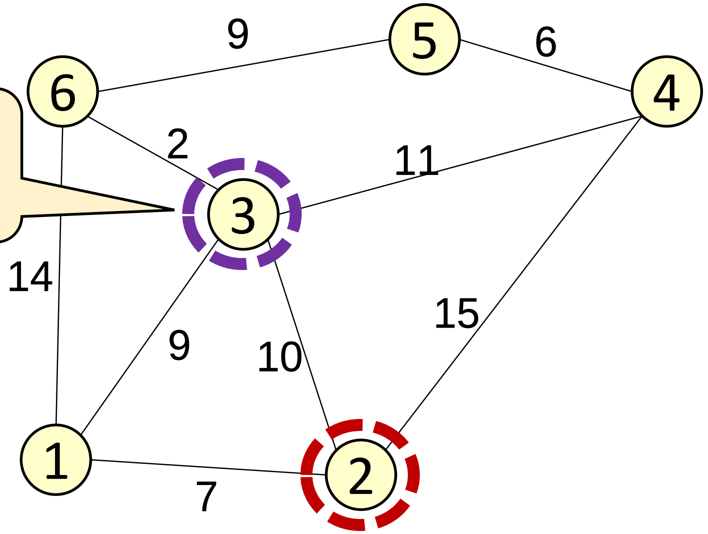


Dijkstra Algorithm

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	14
pre	ND	1	1	ND	ND	1

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist among the unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

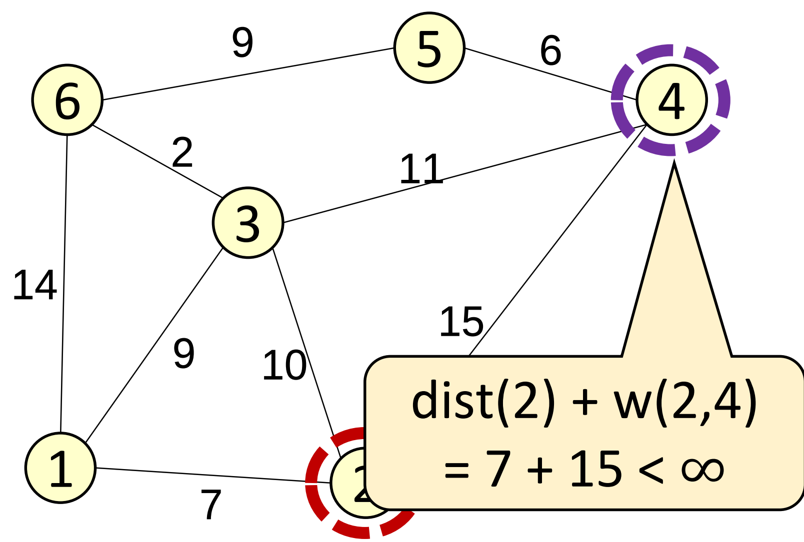


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	7	9	∞	∞	14
pre	ND	1	1	ND	ND	1

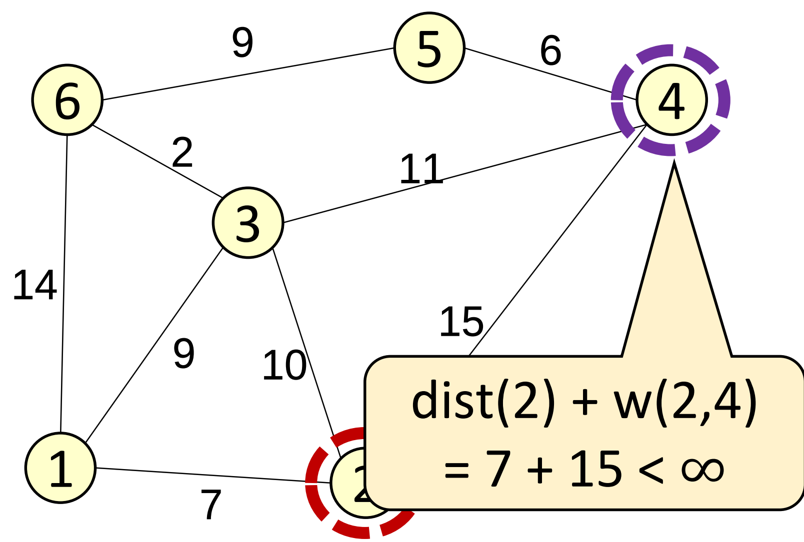


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

vertex	1	2	3	4	5	6
dist	0	7	9	22	∞	14
pre	ND	1	1	2	ND	1



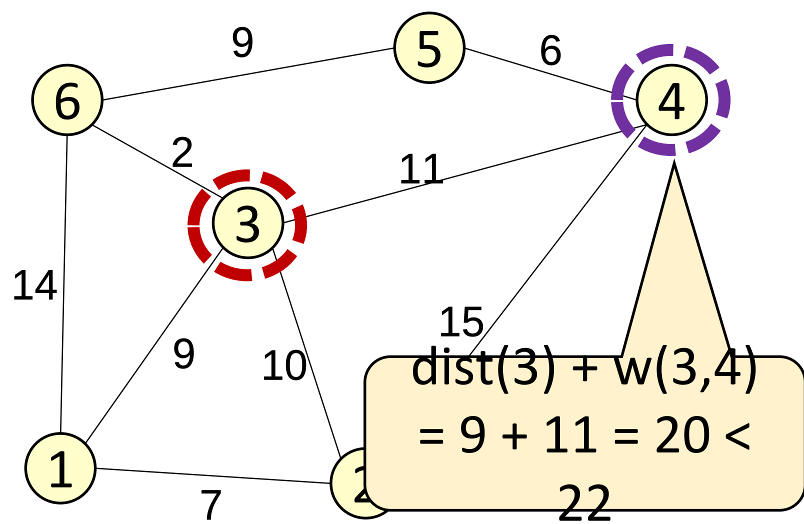
Dijkstra Algorithm

vertex	1	2	3	4	5	6
dist	0	7	9	22	∞	14
pre	ND	1	1	2	ND	1

P P

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

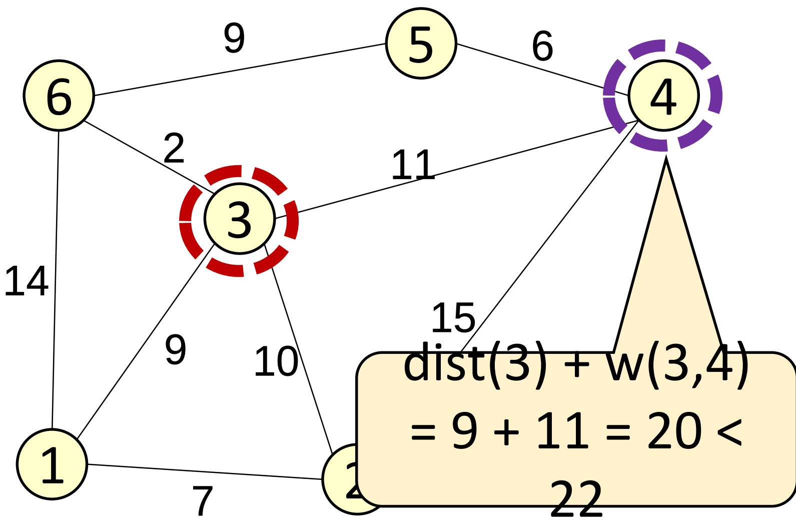


Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
    - Pick the vertex v with the **smallest** dist & unpicked
    - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

	P	P				
vertex	1	2	3	4	5	6
dist	0	7	9	20	∞	14
pre	ND	1	1	3	ND	1



dist(3) + w(3,4) = 9 + 11 = 20 < 22

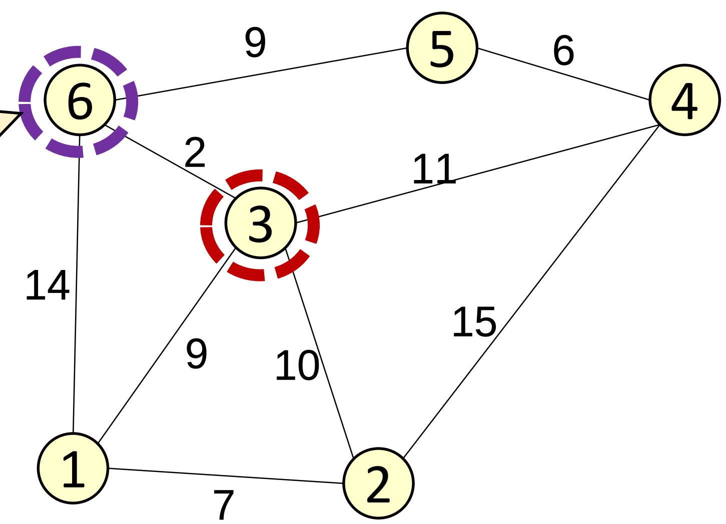
Dijkstra Algorithm

vertex	1	2	3	4	5	6
dist	0	7	9	20	∞	14
pre	ND	1	1	3	ND	1

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist among all unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

dist(3) + w(3,6) = 9 + 2 = 11 < 14



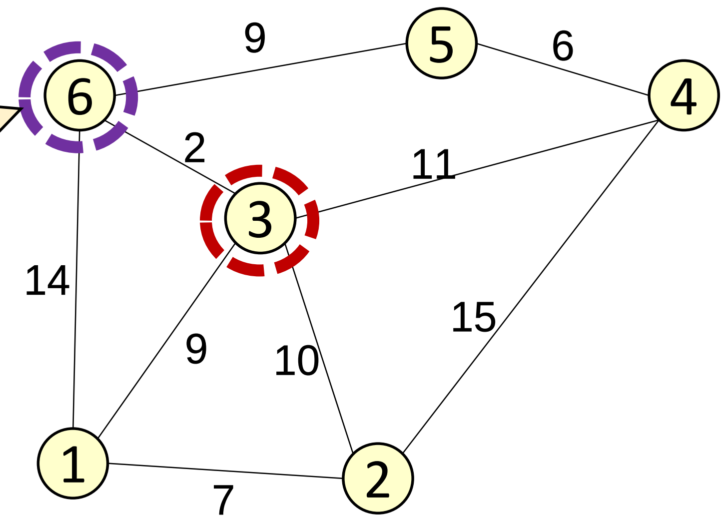
Dijkstra Algorithm

	P	P				
vertex	1	2	3	4	5	6
dist	0	7	9	20	∞	11
pre	ND	1	1	3	ND	3

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the verte the **smallest** unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

dist(3) + w(3,6)
= 9 + 2 = 11 < 14



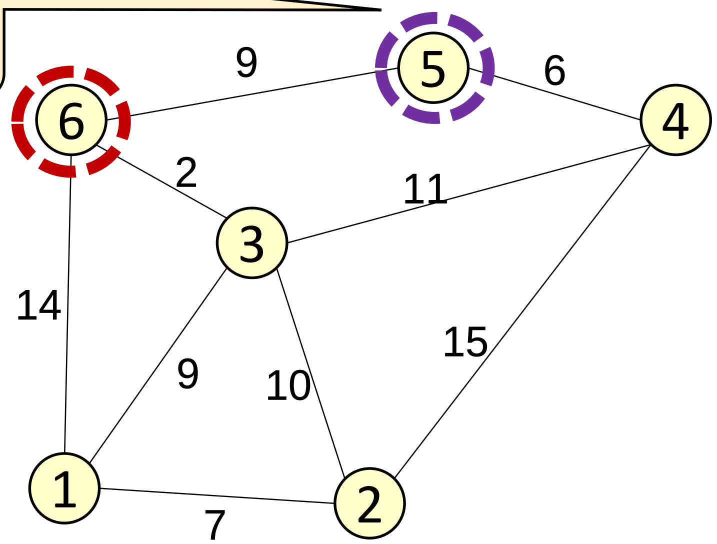
Dijkstra Algorithm

	P	P	P			↓
vertex	1	2	3	4	5	6
dist	0	7	9	20	∞	11
pre	ND	1	1	3	ND	3

What's the shortest path from 1 to 4?

2. Repeat until the destin[ation is] picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

dist(6) + w(6,5) = 11 + 9 = 20 < ∞



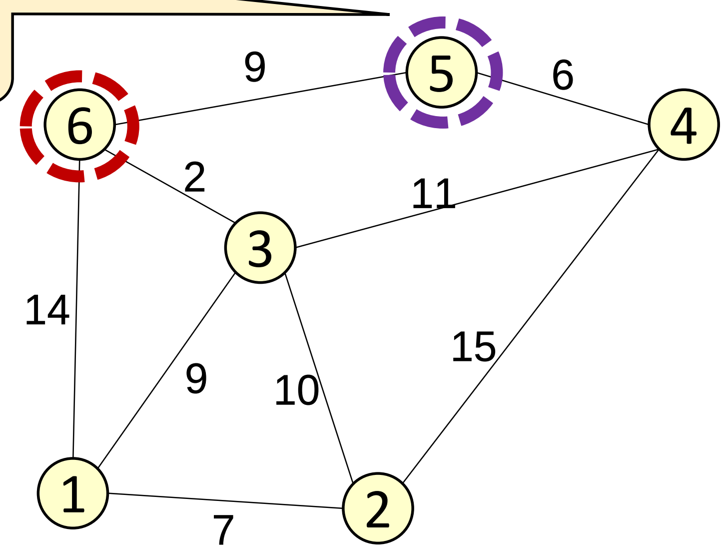
Dijkstra Algorithm

What's the shortest path from 1 to 4?

2. Repeat until the destin[ation is] picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

	P	P	P			
vertex	1	2	3	4	5	6
dist	0	7	9	20	20	11
pre	ND	1	1	3	6	3

dist(6) + w(6,5) = 14 + 9 = 23 < ∞

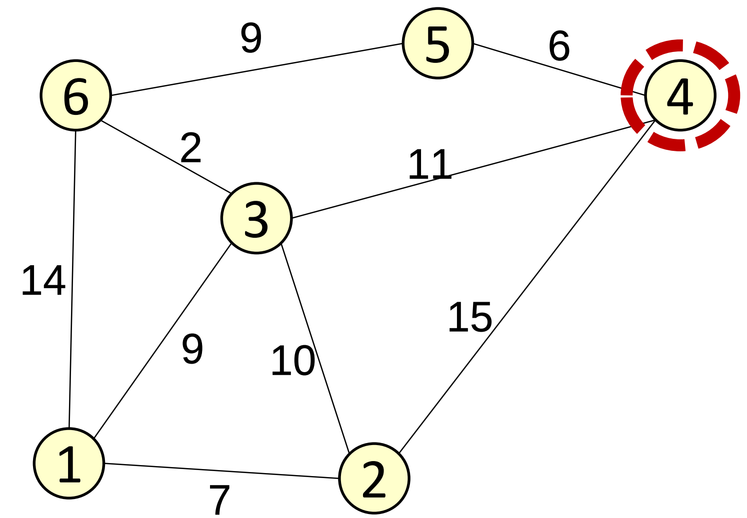


Dijkstra Algorithm

	P	P	P			P
vertex	1	2	3	4	5	6
dist	0	7	9	20	20	11
pre	ND	1	1	3	6	3

What's the shortest path from 1 to 4?

2. Repeat until the destination t is picked
   - Pick the vertex v with the **smallest** dist & unpicked
   - For each neighbor u of v, update the **dist** and **prev** of u if the distance from s to u via v is shorter than the current dist of u;

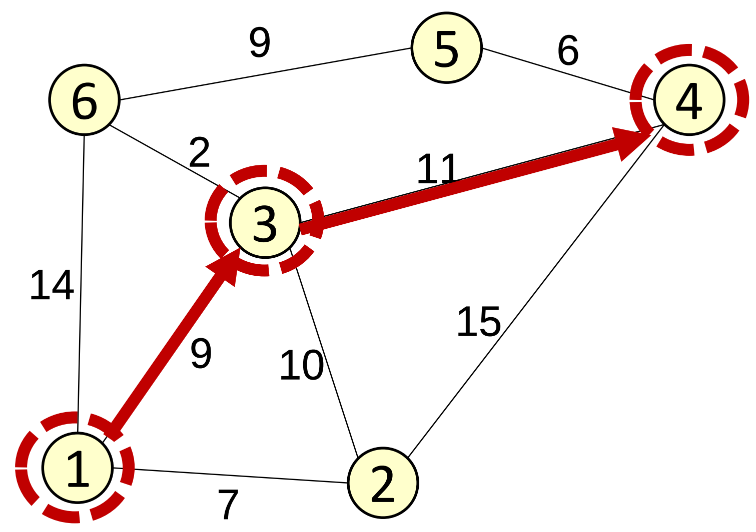


Dijkstra Algorithm

What's the shortest path from 1 to 4?

3. Return the path as indicated by **prev**;

	P	P	P			P
vertex	1	2	3	4	5	6
dist	0	7	9	20	20	11
pre	ND	1	1	3	6	3



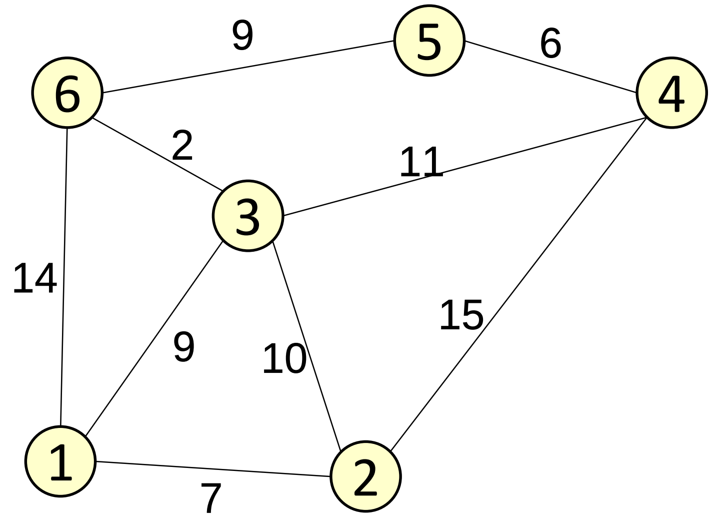
Query Processing on Network Data

Shortest-path problem

Travelling salesman problem

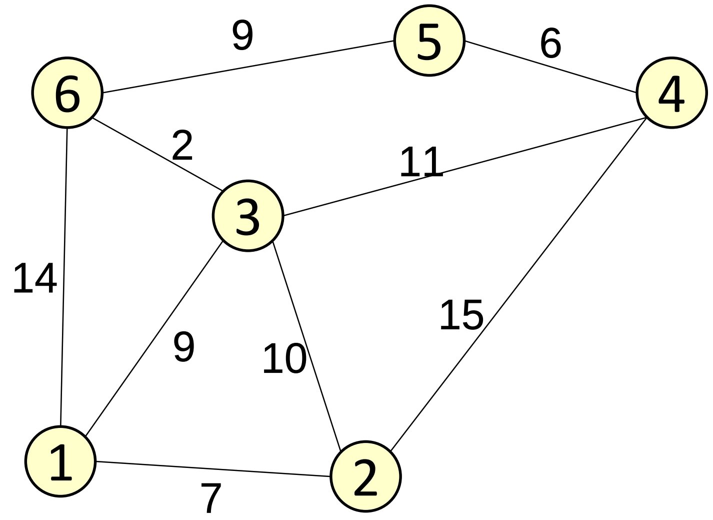
Traveling Salesman Problem

What's the shortest path to traverse each vertex exactly once?



Traveling Salesman Problem

NP-Hard Problem



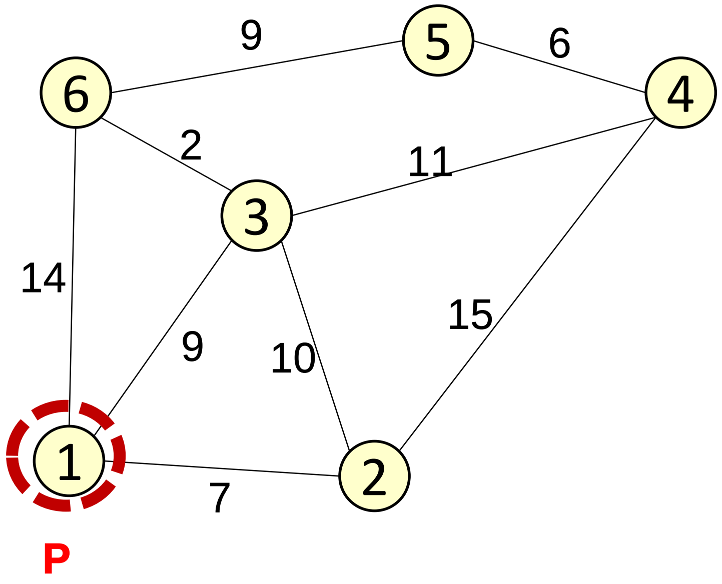
Nearest Neighbor Algorithm

Nearest Neighbor Algorithm:

1. Select a starting vertex u;
2. Repeat until each vertex is traversed;
    - Find the nearest neighbor of the starting point u, says v;
    - Traverse from u to v;
    - Start from v (i.e., set u to be v);

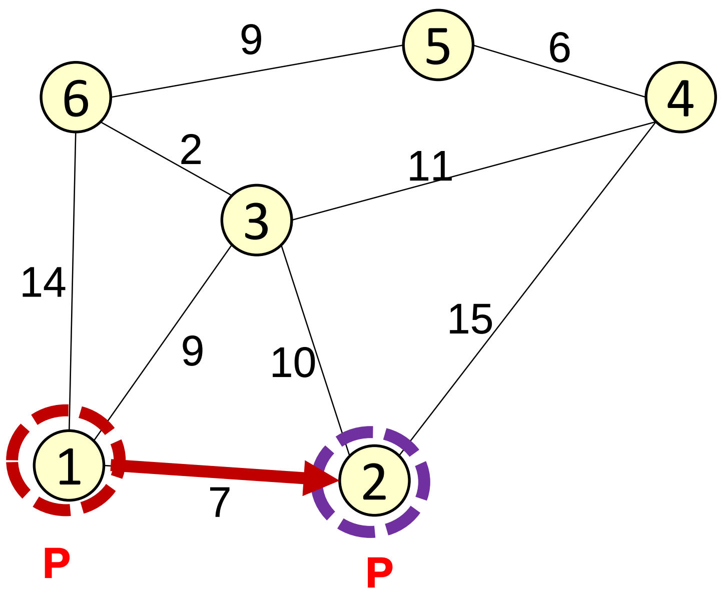
Traveling Salesman Problem

1. Select a starting vertex u;



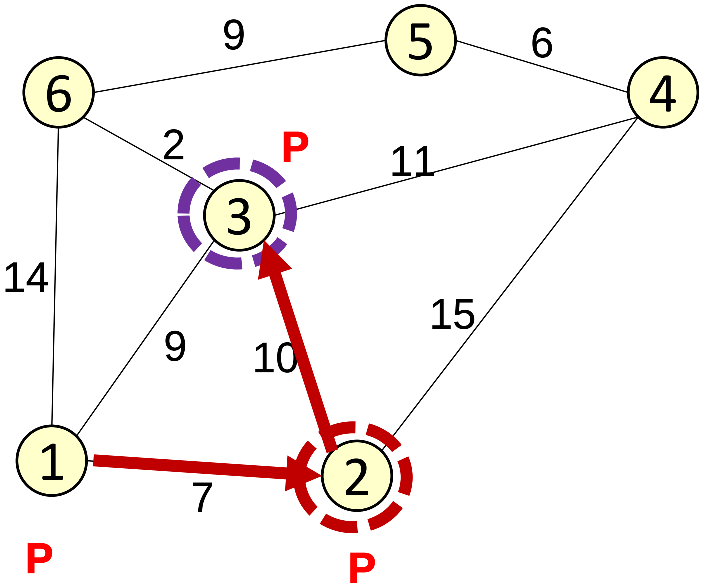
Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



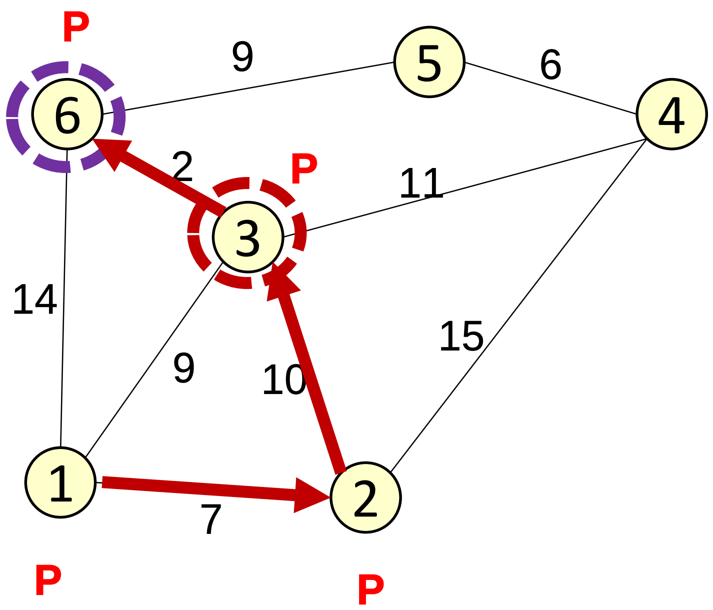
Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



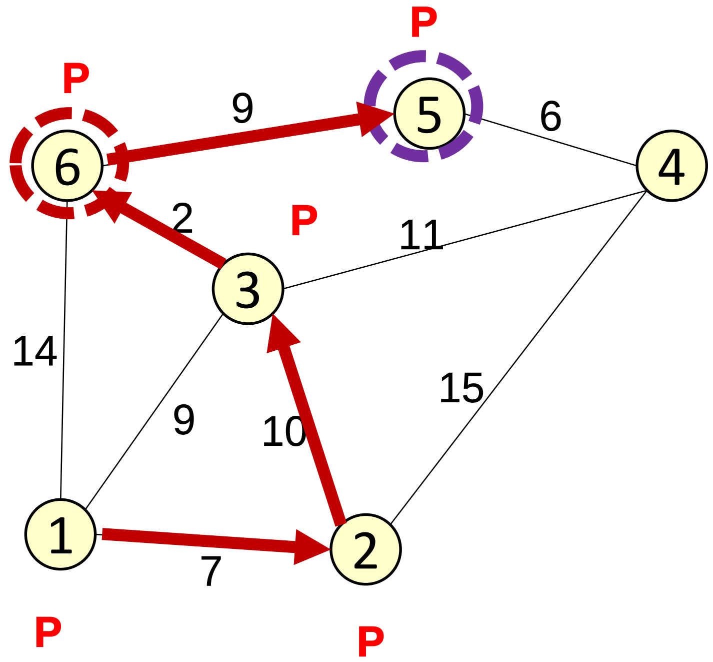
Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



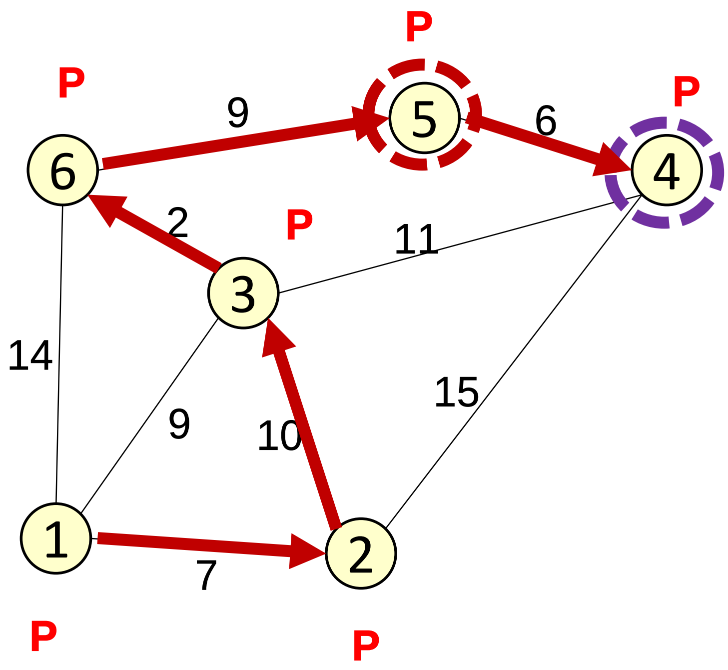
Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



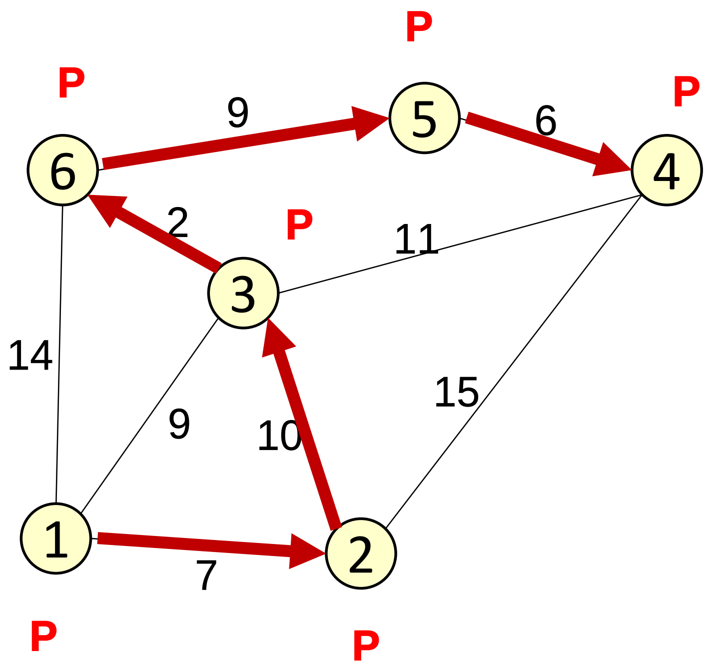
Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



Traveling Salesman Problem

2. Repeat until each vertex is traversed;
   - Find the nearest neighbor of the starting point u, says v;
   - Traverse from u to v;
   - Start from v (i.e., set u to be v);



Recap

- Trajectory Data
- Trajectory Data Preprocessing
- Trajectory Data Indexing
- Trajectory Query Processing
- Network Data Query Processing

Next Lecture

Part 2 – 03: Urban Data Mining (1)
(Spatial Data)