Mining Moving Object and Traffic Data

DASFAA 2010 Tutorial

Jiawei Han, Zhenhui Li, Lu An Tang
Department of Computer Science
University of Illinois at Urbana-Champaign

Tsukuba, Japan
April 2, 2010

©Jiawei Han, Zhenhui Li, Lu An Tang
Tutorial Outline

- Part I. Mining Moving Objects
- Part II. Mining Traffic Data
- Part III. Conclusions
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
Moving Object Data

- A sequence of the location and timestamp of a moving object

- Hurricanes

- Turtles

- Vessels

- Vehicles
Why Mining Moving Object Data?

- Satellite, sensor, RFID, and wireless technologies have been improved rapidly
  - Prevalence of mobile devices, e.g., cell phones, smart phones and PDAs
  - GPS embedded in cars
  - Telemetry attached on animals
- Tremendous amounts of trajectory data of moving objects
  - Sampling rate could be every minute, or even every second
  - Data has been fast accumulated
Complexity of the Moving Object Data

- Uncertainty
  - Sampling rate could be inconstant: From every few seconds transmitting a signal to every few days transmitting one
  - Data be sparse: A recorded location every 3 days
- Noise
  - Erroneous points (e.g., a point in the ocean)
- Background
  - Cars follow underlying road network
  - Animals movements relate to mountains, lakes, ...
- Movement interactions
  - Affected by nearby moving objects
Research Impacts

- Moving object and trajectory data mining has many important, real-world applications driven by the real need
  - Homeland security (e.g., border monitoring)
  - Law enforcement (e.g., video surveillance)
  - Ecological analysis (e.g., animal scientists)
  - Weather forecast
  - Traffic control
  - Location-based service
  - …
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
A moving cluster is a set of objects that move close to each other for a long time interval

- **Note**: Moving clusters and flock patterns are essentially the same

**Formal Definition** [Kalnis et al., SSTD’05]:

- A moving cluster is a sequence of (snapshot) clusters $c_1, c_2, \ldots, c_k$ such that for each timestamp $i$ ($1 \leq i < k$), 
  \[ \frac{|c_i \cap c_{i+1}|}{|c_i \cup c_{i+1}|} \geq \theta \] 
  ($0 < \theta \leq 1$)
Retrieval of Moving Clusters
(Kalnis et al. SSTD’05)

- Basic algorithm (MC1)
  1. Perform DBSCAN for each time slice
  2. For each pair of a cluster $c$ and a moving cluster $g$, check if $g$ can be extended by $c$
     - If yes, $g$ is used at the next iteration
     - If no, $g$ is returned as a result

- Improvements
  - MC2: Avoid redundant checks (Improve Step 2)
  - MC3: Reduce the number of executing DBSCAN (Improve Step 1)
Relative Motion Patterns
(Laube et al. 04, Gudmundsson et al. 07)

- **Flock** \((m > 1, \ r > 0)\): At least \(m\) entities are within a circular region of **radius** \(r\) and they move in the same direction.

- **Leadership** \((m > 1, \ r > 0, \ s > 0)\) At least \(m\) entities are within a circular region of radius \(r\), they move in the same direction, and at least one of the entities was already heading in this direction for at least \(s\) time steps.

- **Convergence** \((m > 1, \ r > 0)\) At least \(m\) entities will **pass through** the same circular region of radius \(r\) (assuming they keep their direction).

- **Encounter** \((m > 1, \ r > 0)\) At least \(m\) entities will be **simultaneously inside** the same circular region of radius \(r\) (assuming they keep their speed and direction).
Examples

An example of a **flock** pattern for $p_1$, $p_2$, and $p_3$ at 8th time step; also a **leadership** pattern with $p_2$ as the leader.

A **convergence** pattern if $m = 4$ for $p_2$, $p_3$, $p_4$, and $p_5$. 
Complexity of Moving Relationship Pattern Mining

- Algorithms: Exact and approximate algorithms are developed
  \( t \) is multiplicative factor in all time bounds

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Exact (from [15])</th>
<th>Exact (new)</th>
<th>Approximate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flock</td>
<td>( O(nm^2 + n \log n) )</td>
<td>–</td>
<td>( O(n \frac{r}{\varepsilon^2} \log \frac{1}{\varepsilon} + n \log n) ) (radius)</td>
</tr>
<tr>
<td>Leadership</td>
<td>( O(ns + nm^2 + n \log n) )</td>
<td>–</td>
<td>( O(ns + \frac{1}{\varepsilon^2} n \log \frac{1}{\varepsilon} + n \log n) ) (radius)</td>
</tr>
<tr>
<td>Convergence</td>
<td>( O(n^2) )</td>
<td>–</td>
<td>( O(n^2 + \frac{\delta}{\varepsilon m}) ) (subset)</td>
</tr>
<tr>
<td>Encounter</td>
<td>( O(n^4) )</td>
<td>( O(n^3) ) (all)</td>
<td>( O(\frac{1}{\varepsilon^2} n^2 \log n) ) (radius)</td>
</tr>
</tbody>
</table>
  \( O(((m + \log n)n^2) \) (detect) |
  \( O((M + \log n)n^2 \log M) \) (largest) |

- Flock: Use the higher-order Voronoi diagram
- Leadership: Check the leader condition additionally
- …
An Extension of Flock Patterns
(Gudmundsson et al. GI S’06, Benkert et al. SAC’07)

- A new definition considers *multiple* time steps, whereas the previous definition *only one* time step
- **Flock**: A *flock* in a time interval \( I \), where the duration of \( I \) is at least \( k \), consists of at least \( m \) entities such that for every point in time within \( I \), there is a disk of radius \( r \) that contains all the \( m \) entities
  - e.g.,

A flock through 3 time steps
Computing Flock Patterns

- Approximate flocks
  - Convert overlapping segments of length $k$ to points in a $2k$-dimensional space
  - Find $2k$-d pipes that contain at least $m$ points

- Longest duration flocks
  - For every entity $v$, compute a cylindrical region and the intervals from the intersection of the cylinder
  - Pick the longest one
Convoy: An Extension of Flock Pattern
(Jeung et al. ICDE’08 & VLDB’08)

- Flock pattern has rigid definition with a circle
- Convoy use *density-based clustering* at each timestamp

**Figure 1: Lossy-flock Problem**

**Figure 4: An Example of a Convoy**
Efficient Discovery of Convoys

- Base-line algorithm:
  - Calculate density-based clusters for each timestamp
  - Overlap clusters for every k consecutive timestamps
- Speedup algorithm using trajectory simplification
  - Trajectory simplification

Figure 6: Trajectory Simplification
A Filter-and-Refine Framework for Convoy Mining

- Filter-and-refine framework
  - Filter: partition time into $\lambda$-size time slot; a trajectory is transformed into a set of segments; density-based clustering on segments.
  - Refine: Look into every $\lambda$-size time slot, refine the clusters based on points.

Figure 9: Measure of $\omega(o'_q, o'_i)$ and Time Partitioning
**Leadership**: if there is an entity that is a leader of at least $m$ entities for at least $k$ time units.

An entity $e_j$ is said to be a leader at time $[t_x, t_y]$ for time-points $t_x$, $t_y$, if and only if $e_j$ does not follow anyone at time $[t_x, t_y]$, and $e_j$ is followed by sufficiently many entities at time $[t_x, t_y]$.

$e_i$ follows $e_j$

$||d_i - d_j|| \leq \beta$
Algorithm: Build and use the follow-arrays

e.g., Store nonnegative integers specifying for how many past consecutive unit-time-intervals \( e_j \) is following \( e_i \) (\( e_j \neq e_i \))
Swarms: A Relaxed but Real, Relative Movement Pattern

- Flock and convoy all require \( k \) consecutive time stamps (still very rigid definition)
- Moving objects may not be close to each other for consecutive time stamps (need to relax time constraint)
Discovery of Swarm Patterns

- A system that mines moving object patterns: Z. Li, et al., "MoveMine: Mining Moving Object Databases", SIGMOD’10 (system demo)
- Z. Li, B. Ding, J. Han, and R. Kays, “Swarm: Mining Relaxed Temporal Moving Object Clusters”, in submission
A trajectory pattern should describe the movements of objects both in space and in time.
Trajectory (T-) Patterns: Definition

- A Trajectory Pattern (T-pattern) is a couple \((s, \alpha)\):
  - \(s = (x_0, y_0), \ldots, (x_k, y_k)\) is a sequence of \(k+1\) locations
  - \(\alpha = \alpha_1, \ldots, \alpha_k\) are the transition times (annotations)
    also written as:

\[
(x_0, y_0) \rightarrow (x_1, y_1) \rightarrow \ldots \rightarrow (x_k, y_k)
\]

- A T-pattern \(T_p\) occurs in a trajectory if the trajectory contains a subsequence \(S\) such that:
  - Each \((x_i, y_i)\) in \(T_p\) matches a point \((x_i', y_i')\) in \(S\), and the transition times in \(T_p\) are similar to those in \(S\)
Characteristics of Trajectory-Patterns

- Routes between two consecutive regions are not relevant

  These two movements are not discriminated

  \[ \text{A} \rightarrow \text{B} \]
  \[ \text{B} \rightarrow \text{A} \]
  1 hour

- Absolute times are not relevant

  These two movements are not discriminated

  \[ \text{A} \rightarrow \text{B} \]
  \[ \text{B} \rightarrow \text{A} \]
  1 hour at 5 p.m.
  1 hour at 9 a.m.
Trajectory-Pattern Mining

- Convert each trajectory to a sequence, i.e., by converting a location \((x, y)\) into a region.
  \[
  S = \langle (x_1,y_1,t_1), \ldots, (x_5,y_5,t_5) \rangle
  \]
  \[
  \langle (R_4,t_1), (R_3,t_3), (R_3,t_4), (R_1,t_5) \rangle
  \]

- Execute the TAS (temporally annotated sequence) algorithm, over the set of converted trajectories.
  - A TAS is a sequential pattern annotated with typical transition times between its elements.
  - The algorithm of TAS mining is an extension of PrefixSpan so as to accommodate transition times.
Sample Trajectory-Patterns

Data Source: Trucks in Athens – 273 trajectories

$t_1$ in $[400, 513]$
$t_2$ in $[41, 61]$
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
In many applications, objects follow the same routes (approximately) over regular time intervals.

e.g., Bob wakes up at the same time and then follows, more or less, the same route to his work everyday.
Let \( S \) be a sequence of \( n \) spatial locations, \( \{l_0, l_1, \ldots, l_{n-1}\} \), representing the movement of an object over a long history.

Let \( T << n \) be an integer called *period*, and \( T \) is given.

A *periodic pattern* \( P \) is defined by a sequence \( r_0 r_1 \ldots r_{T-1} \) of length \( T \) that appears in \( S \) by more than \( min\_sup \) times.

For every \( r_i \) in \( P \), \( r_i = * \) or \( l_{j*T+i} \) is inside \( r_i \).
Periodic Pattern Mining (I)

1. Obtain frequent 1-patterns

- Divide the sequence $S$ of locations into $T$ spatial datasets, one for each offset of the period $T$, i.e., locations \( \{l_i, l_{i+T}, \ldots, l_{i+(m-1)T}\} \) go to a set $R_i$

- Perform DBSCAN on each dataset

  - e.g.,

Five clusters discovered in datasets $R_1, R_2, R_3, R_4,$ and $R_6$
2. Find longer patterns: Two methods

- **Bottom-up level-wise technique**
  - Generate $k$-patterns using a pair of $(k-1)$-patterns with their first $k-2$ non-* regions in the same position
  - Use a variant of the Apriori-TID algorithm
Periodic Pattern Mining (III)

- Faster top-down approach
  - Replace each location in $S$ with the cluster-id which it belongs to or with * if the location belongs to no cluster
  - Use the sequence mining algorithm to discover fast all frequent patterns of the form $r_0r_1…r_{T-1}$, where each $r_i$ is a cluster in a set $R_i$ or *
  - Create a max-subpattern tree and traverse the tree in a top-down, breadth-first order
Periodic Patterns of Moving objects

- Periodic behavior is the intrinsic behavior for most moving objects
  - Yearly migration of birds
    - Fly to south for winter, fly back to north for summer
  - People’s daily routines
    - Go to office at 9:00am, back home around 6:00pm
- Detecting periodic behavior is useful for:
  - Summarizing over long historical movement
    - People’s behavior could be summarized as some daily behavior and weekly behavior
  - Predicting future movement
    - E.g., predict the location at the future time (next day, next week, or next year)
  - Help detect abnormal events
    - A bird does not follow its usual migration path ⇒ a signal of environment change
Challenges of Periodic Pattern Mining

Raw data of David’s movement

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Location</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009–02–05</td>
<td>07:01</td>
<td></td>
<td>601, 254</td>
</tr>
<tr>
<td>2009–02–05</td>
<td>09:14</td>
<td></td>
<td>811, 60</td>
</tr>
<tr>
<td>2009–02–05</td>
<td>10:58</td>
<td></td>
<td>810, 55</td>
</tr>
<tr>
<td>2009–02–05</td>
<td>14:29</td>
<td></td>
<td>820, 100</td>
</tr>
</tbody>
</table>

Hidden periodic behaviors

- **Periodic Behavior #1**
  (Period: day; Time span: Sept. – May)
  9:00–18:00 in the office
  20:00–8:00 in the dorm

- **Periodic Behavior #2**
  (Period: day; Time span: June – Aug.)
  8:00–18:00 in the company
  20:00–7:30 in the apartment

- **Periodic Behavior #3**
  (Period: week; Time span: Sept. – May)
  13:00–15:00 Mon. and Wed. in the classroom
  14:00–16:00 Tues. and Thurs. in the gym

**Interleaved periods**

**Multiple periods**

**Different locations**
Detection of Periods: A Naïve Method

- Transform the movement points into complex plane
  - \((x, y) \rightarrow x-yi\)
  - \((x, y) \rightarrow y-xi\)
- Apply Fourier Transform
- Weakness:
  - Affected by noise
  - \((x, y) \rightarrow x-yi\) and \(y-xi\), each produce different result
  - It cannot detect partial period
    - Short FFT solves partial period problem, but it is not easy to generalize the result

36
A Motivating Example: Trajectories of Bees

Bee and Flower:
8 hours stays in the nest
16 hours fly nearby
FFT Transformation Does Not Work

Transform \((x,y)\) into complex plane (two ways to transform)

\((x,y) \rightarrow x - yi\)

\((x,y) \rightarrow y - xi\)

FFT should have strongest power at 42.7 (\(T = 24\), \(NFFT/T = 1024/24 = 42.7\))

Failed!
Observation/Reference Spot: The Nest

Period is more obvious in this binary sequence!
Algorithm General Framework

- **Detecting periods**: Use observation spots to find multiple interleaved periods
  - Observation spots are detected using **density-based method**
  - Periods are detected for each obs. spot using **Fourier Transform and auto-correlation**

- **Summarizing periodic behaviors**: via clustering
  - Give the statistical explanation of the behavior
  - E.g., “David has 80% probability to be at the office.”
Running Example: Bald Eager Migration

Real data of a bald eagle over 3 years
Method: Finding Observation Spots

- Observation spots:
  - Frequently visited regions/locations
  - They should have higher density than a random location

- Partition the map into grids and use kernel-based method to find high density regions:

  For each grid cell \( c \), the density is estimated using the bivariate normal density kernel,

  \[
  f(c) = \frac{1}{n\gamma^2} \sum_{i=1}^{n} \frac{1}{2\pi} \exp\left(-\frac{|c - loc_i|^2}{2\gamma^2}\right),
  \]

- Find the observation spots using the contour of high density places
Example: Finding Observation Spots

Density

Observation spots
Period Detection for Each Observation Spot

- For each observation spot, the movement is transformed into a binary sequence.
  - 0: not in the obs. spot
  - 1: in the obs. spot
- Use Fourier Transform combined with auto-correlation to find the periods
Example: Detect Periods for Each Obs. Spot

Period candidates first detected using Fourier Transform

The exact period is further refined using circular autocorrelation
Summarizing Periodic Behaviors

- For each period, the movement is divided into segments
  - if the period is “day”, each segment is a day
- Some segments during a time period form a periodic behavior
  - Daily behavior in summer
  - Daily behavior in winter
- To distinguish interleaved behavior, apply clustering on the segments
- A representative behavior is summarized over all the segments in a cluster
Example: Summarizing Periodic Behaviors
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
Clustering: Distance-Based vs. Shape-Based

- Distance-based clustering: Find a group of objects moving together
  - For whole time span
    - high-dimensional clustering
    - probabilistic clustering
  - For partial continuous time span
    - density-based clustering
    - moving cluster, flock, convoy (borderline case between clustering and patterns)
  - For partial discrete time span
    - swarm (borderline case between clustering and patterns)
- Shape-based clustering: Find similar shape trajectories
  - Variants of shape: translation, rotation, scaling, and transformation
  - Sub-trajectory clustering
High-Dimensional Clustering & Distance Measures

- Treat each timestamp as one dimension
- Many high-dimensional clustering methods can be applied to cluster moving objects
- Most popular high-dimensional distance measure
  - Euclidean distance
  - Dynamic Time Warping
  - Longest Common Subsequence
  - Edit Distance with Real Penalty
  - Edit Distance on Real Sequence
# High-Dimensional Distance Measures

<table>
<thead>
<tr>
<th>Distance Measure</th>
<th>Local Time Shifting</th>
<th>Noise</th>
<th>Metric</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>O(n)</td>
</tr>
<tr>
<td>DTW (Yi et al., ICDE’98)</td>
<td>✓</td>
<td></td>
<td></td>
<td>O(n^2)</td>
</tr>
<tr>
<td>LCSS (Vlachos et al., KDD’03)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>O(n^2)</td>
</tr>
<tr>
<td>ERP (Chen et al., VLDB’04)</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>EDR (Chen et al., SIGMOD’05)</td>
<td>✓</td>
<td>✓</td>
<td>(consider gap)</td>
<td>O(n^2)</td>
</tr>
</tbody>
</table>
Probabilistic Trajectory Clustering
(Gaffney et al., KDD’00; Chudova et al., KDD’03)

- Basic assumption: Data produced in the following *generative* manner
  - An individual is drawn randomly from the population of interest
  - The individual has been assigned to a cluster $k$ with probability $w_k$, $\sum_{k=1}^{K} w_k = 1$, these are the *prior* weights on the $K$ clusters
  - Given that an individual belongs to a cluster $k$, there is a density function $f_k(y_j \mid \theta_k)$ which generates an observed data item $y_j$ for the individual $j$
  - The probability density function of observed trajectories is a mixture density

$$P(y_j \mid x_j, \theta) = \sum_{k}^{K} f_k(y_j \mid x_j, \theta_k) w_k$$

- $f_k(y_j \mid x_j, \theta_k)$ is the density component
- $w_k$ is the weight, and $\theta_k$ is the set of parameters for the $k$-th component
- $\theta_k$ and $w_k$ can be estimated from the trajectory data using the *Expectation-Maximization* (EM) algorithm
Clustering Results For Hurricanes
(Camargo et al. 06)

Density-Based Trajectory Clustering
(M. Nanni & D. Pedreschi, JIIIS’06)

- Define the distance between whole trajectories
  - A trajectory is represented as a sequence of location and timestamp
  - The distance between trajectories is the average distance between objects for every timestamp
- Use the OPTICS algorithm for trajectories
  - e.g.,

![Diagram of trajectory clustering with reachability plot](attachment://reachability_plot.png)

Reachability Plot
Four clusters
Temporal Focusing: TF-OPTICS
(M. Nanni & D. Pedreschi, JIIS'06)

- In a real environment, not all time intervals have the same importance
  - e.g., *in rush hours*, many people move from home to work or vice versa
- Clustering trajectories only in meaningful time intervals can produce more interesting results
- TF-OPTICS aims at **searching the most meaningful time intervals**, which allows us to isolate the clusters of higher quality

Method:
- Define the quality of a clustering
  - Take account of both high-density clusters and low-density noise
  - Can be computed directly from the reachability plot
- Find the time interval that maximizes the quality
  1. Choose an initial random time interval
  2. Calculate the quality of neighborhood intervals generated by increasing or decreasing the starting or ending times
  3. Repeat Step 2 as long as the quality increases
Invariant Distance Measures for Trajectories
(Vlachos et al., KDD’04)

- Invariants: Translation, Rotation, Scaling, Transformation

Similar trajectories in different invariants

- Map each trajectory to a trajectory in a rotation invariant space.
  - Movement vector: \( V_t = P_t - P_{t-1}, \quad t = 1, \ldots, n - 1 \)
  - angles of each movement vector is relative to a reference vector
  - Angle/Arc-Length pairs (AAL)
    \[
    P_{AAL} = [(\hat{V}_1, \frac{\|V_1\|}{\sum_i \|V_i\|}), \ldots, (\hat{V}_{n-1}, \frac{\|V_{n-1}\|}{\sum_i \|V_i\|})]
    \]

- Use Dynamic Time Warping (DTW) to measure the distance in invariant space

Figure 7: Elastic matching achieved by DTW.
Trajectory Clustering: A Partition-and-Group Framework (Lee et al., SIGMOD’07)

- Existing algorithms group trajectories as a whole ⇒ They might not be able to find similar portions of trajectories
  - e.g., common behavior cannot be discovered since $TR_1$~$TR_5$ move to totally different directions

$TR_1$, $TR_2$, $TR_3$, $TR_4$, $TR_5$

- Partition-and-group: discovers common sub-trajectories
- Usage: Discover regions of special interest
  - Hurricane Landfall Forecasts: Discovery of common behaviors of hurricanes near the coastline or at sea (i.e., before landing)
  - Effects of Roads and Traffic on Animal Movements: Discover common behaviors of animals near the road
Two phases: *partitioning* and *grouping*

**Note**: A representative trajectory is a common sub-trajectory
The Partitioning Phase

- Identify the points where the behavior of a trajectory changes rapidly ⇒ characteristic points
  - An optimal set of characteristic points is found by using the minimum description length (MDL) principle

- Partition a trajectory at every characteristic point
Overview of the MDL Principle

- The MDL principle has been widely used in information theory.
- The MDL cost consists of two components: $L(H)$ and $L(D|H)$, where $H$ means the hypothesis, and $D$ the data.
  - $L(H)$ is the length, in bits, of the description of the hypothesis.
  - $L(D|H)$ is the length, in bits, of the description of the data when encoded with the help of the hypothesis.
- The best hypothesis $H$ to explain $D$ is the one that minimizes the sum of $L(H)$ and $L(D|H)$. 
MDL Formulation

- Finding the optimal partitioning translates to finding the best hypothesis *using the MDL principle*
  - $H \rightarrow$ a set of trajectory partitions, $D \rightarrow$ a trajectory
  - $L(H) \rightarrow$ the sum of the length of all trajectory partitions
  - $L(D|H) \rightarrow$ the sum of the difference between a trajectory and a set of its trajectory partitions

\[
L(H) = \log_2(len(p_1p_4)) \\
L(D|H) = \log_2(d_\perp(p_1p_4, p_1p_2) + d_\perp(p_1p_4, p_2p_3) + d_\perp(p_1p_4, p_3p_4)) + \\
\log_2(d_\theta(p_1p_4, p_1p_2) + d_\theta(p_1p_4, p_2p_3) + d_\theta(p_1p_4, p_3p_4))
\]

- $L(H)$ measures *conciseness*; $L(D|H)$ *preciseness*
Grouping Phase (1/2)

- Find the clusters of trajectory partitions using density-based clustering (i.e., DBSCAN)
  - A density-connect component forms a cluster, e.g., \{ L_1, L_2, L_3, L_4, L_5, L_6 \}

$$MinLns = 3$$
Grouping Phase (2/2)

- Describe the overall movement of the trajectory partitions that belong to the cluster

A red line: a representative trajectory,
A blue line: an average direction vector,
Pink lines: line segments in a density-connected set
Example: Trajectory Clustering Results

570 Hurricanes (1950~2004)
Seven clusters discovered from the hurricane data set

Red line: a representative trajectory

Two clusters discovered from a deer data set
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
Location Prediction for Moving Objects

- Predicting future location
  - Based on its own history of one moving object
    - Linear (not practical) vs. non-linear motion (more practical)
    - Vector based (predict near time, e.g., next minute) vs. pattern based (predict distant time, e.g., next month/year)
  - Based on all moving objects’ trajectories
    - based on frequent patterns
Recursive Motion Function
(Tao et al., SIGMOD’04)

- Non-linear model, near time prediction, vector-based method
- Linear model is not practical in prediction, so better to use non-linear model
  - Recursive motion function
    \[ o(t) = C_1 \cdot o(t-1) + C_2 \cdot o(t-2) + \ldots + C_f \cdot o(t-f) \]
    
    \( C_i \) is a constant matrix expressing several complex movement types, including polynomials, ellipse, sinusoids, etc.
- Use basic motion matrices to model unknown motion matrices

\[ \text{Figure 1.1: Failure of linear prediction} \]

\[ \text{Figure 6.1: Movements with known motion matrices} \]

\[ \text{Figure 6.3: Movements with unknown motion matrices} \]
Efficient Implementation: Recursive Motion Prediction

- Indexing expected trajectories using Spatio-Temporal Prediction Tree (STP-Tree)
- A combination of Time Parameterized R Tree (TPR-tree) and TPR*-tree

\[ \text{Figure 2.2: Entry representations in a TPR-tree} \]
\[ \text{Figure 2.3: The split algorithm of the TPR*-tree} \]
Experimental Results: Recursive Motion Prediction

- Effectiveness (wrt retrospect)

Figure 6.5: Improvements in peach with larger retrospect

- Efficiency using STP-tree indexing

Figure 6.10: Query costs vs. qilen (e=2.5%)
Hybrid Prediction (Jeung et al., ICDE’08)

- Combining pattern and vector to prediction locations
- Can predict both distant and near time locations
  Ex: inadequate prediction using vector-based method
- Mining frequent periodic patterns

Fig. 3. An Example of Trajectory Patterns
Hybrid Prediction: Implementation and Predication

- **Implementation**: Indexing patterns using trajectory pattern tree

- **Prediction**:
  - For non-distant query, use Forward Query Processing to retrieve all the trajectory patterns
    - The premise of the trajectory pattern is similar to that of the query pattern key
    - Its corresponding consequence time offset is the same as the query time
  - For distant query, use Backward Query Processing to retrieve patterns
    - Give up the premise key in FQP & relax time constraint
Prediction Using Frequent Trajectory Patterns (Monreale et al., KDD’09)

Use frequent T-patterns of other moving objects

If many moving objects follow a pattern, it is likely that a moving object will also follow this pattern

Method

- Mine T-Patterns
- Construct T-Pattern Tree
- Predict using T-pattern tree

Figure 2: T-pattern Tree construction
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
Trajectory Classification

- Task: Predict the class labels of moving objects based on their trajectories and other features
- Two approaches
  - Machine learning techniques
    - Studied mostly in pattern recognition, bioengineering, and video surveillance
  - The hidden Markov model (HMM)
  - Trajectory-based classification (TraClass): Trajectory classification using hierarchical region-based and trajectory-based clustering
Machine Learning for Trajectory Classification (Sbalzarini et al. 02)

- Compare various machine learning techniques for biological trajectory classification
- Data encoding
  - For the hidden Markov model, a whole trajectory is encoded to a sequence of the momentary speed
  - For other techniques, a whole trajectory is encoded to the mean and the minimum of the speed of a trajectory, thus a vector in \( \mathbb{R}^2 \)
- Two 3-class datasets: Trajectories of living cells taken from the scales of the fish Gillichthys mirabilis
  - Temperature dataset: 10° C, 20° C, and 30° C
  - Acclimation dataset: Three different fish populations
Machine Learning Techniques Used

- **k-nearest neighbors (KNN)**
  - A previously unseen pattern $x$ is simply assigned to the same class to which the majority of its $k$-nearest neighbors belongs.

- **Gaussian mixtures with expectation maximization (GMM)**

- **Support vector machines (SVM)**

- **Hidden Markov models (HMM)**
  - **Training**: Determine the model parameters $\lambda = (A, B, \pi)$ to maximize $P[x | \lambda]$ for a given observation $x$.
  - **Evaluation**: Given an observation $x = \{O_1, \ldots, O_T\}$ and a model $\lambda = (A, B, \pi)$, compute the probability $P[x | \lambda]$ that the observation $x$ has been produced by a source described by $\lambda$.
The measurement sequence is divided into overlapping segments.
In each segment, the trajectory of the car is approximated by a smooth function and then assigned to one of four categories: ahead, left, right, or stop.
The list of segments is reduced to a string of symbols drawn from the set \{a, l, r, s\}.
The string of symbols is classified using the hidden Markov model (HMM).
Use of the HMM for Classification

- Classification of the global motions of a car is carried out using an HMM.
- The HMM contains four states which are in order A, L, R, S, which are the true states of the car: ahead, turning left, turning right, stopped.
- The HMM has four output symbols in order a, l, r, s, which are the symbols obtained from the measurement segments.
- The Viterbi algorithm is used to obtain the sequence of internal states.

This measurement sequence means the driver stops and then turns to the right.
Motion Trajectory Classification
(Bashir et al. 07)

- Motion trajectories
  - Tracking results from video trackers, sign language data measurements gathered from wired glove interfaces, and so on
- Application scenarios
  - Sport video (e.g., soccer video) analysis
    - Player movements ⇒ A strategy
  - Sign and gesture recognition
    - Hand movements ⇒ A particular word
- The HMM-Based Algorithm
  1. Trajectories are segmented at points of change in curvature
  2. Sub-trajectories are represented by their Principal Component Analysis (PCA) coefficients
  3. The PCA coefficients are represented using a GMM for each class
  4. An HMM is built for each class, where the state of the HMM is a sub-trajectory and is modeled by a mixture of Gaussians
Use of the HMM for Classification

- Training and parameter estimation
  - The Baum-Welch algorithm is used to estimate the parameters

- Classification
  - The PCA coefficient vectors of input trajectories after segmentation are posed as an observation sequence to each HMM (i.e., constructed for each class)
  - The maximum likelihood (ML) estimate of the test trajectory for each HMM is computed
  - The class is determined to be the one that has the largest maximum likelihood

- Experiment: Datasets
  - The Australian Sign Language dataset (ASL)
    - 83 classes (words), 5,727 trajectories
  - A sport video data set (HJSL)
    - 2 classes, 40 trajectories of high jump and 68 trajectories of slalom skiing objects

- Accuracy

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ASL</th>
<th>HJSL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Classes : #Trajectories</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2:</td>
<td>4:</td>
</tr>
<tr>
<td></td>
<td>138</td>
<td>276</td>
</tr>
<tr>
<td>HMM</td>
<td>0.96</td>
<td>0.92</td>
</tr>
<tr>
<td>GMM</td>
<td>0.98</td>
<td>0.89</td>
</tr>
<tr>
<td>PCA-DE</td>
<td>0.94</td>
<td>0.93</td>
</tr>
</tbody>
</table>
A Critique of Previous Methods

- Common Characteristics of Previous Methods: Use the *shapes* of *whole* trajectories to do classification
  - Encode a whole trajectory into a feature vector;
  - Convert a whole trajectory into a string or a sequence of the momentary speed; or
  - Model a whole trajectory using the HMM

- **Note**: Although a few methods segment trajectories, the main purpose is to approximate or smooth trajectories before using the HMM
TraClass: Trajectory Classification Based on Clustering

- Motivation
  - Discriminative features are likely to appear at \textit{parts} of trajectories, not at whole trajectories
  - Discriminative features appear not only as common movement patterns, but also as \textit{regions}

- Solution
  - Extract features in a top-down fashion, first by \textit{region-based clustering} and then by \textit{trajectory-based clustering}
Intuition and Working Example

Parts of trajectories near the container port and near the refinery enable us to distinguish between container ships and tankers even if they share common long paths.

Those in the fishery enable us to recognize fishing boats even if they have no common path there.
Region-Based Clustering

Trajectory Partitions

Features

Region-Based Clustering

Trajectory-Based Clustering

Region-Based Clustering

Trajectory-Based Clustering

(1) A B C
   D E F
   G H I
(2) A B C
   D E F
   G H I
(3) A B C
   D E F
   G H I
(4) A B C
   D E F
   G H I
1. Trajectories are partitioned based on their shapes as in the partition-and-group framework.

2. Trajectory partitions are further partitioned by *the class labels*.

- The real interest here is to guarantee that trajectory partitions do not span the class boundaries.

![Diagram](image)

- Non-discriminative
- Discriminative

---

**Class-Conscious Trajectory Partitioning**

![Class A and Class B](image)

Additional partitioning points
Region-Based Clustering

- Objective: Discover regions that have trajectories mostly of one class regardless of their movement patterns
- Algorithm: Find a better partitioning alternately for the X and Y axes as long as the MDL cost decreases
  - The MDL cost is formulated to achieve both homogeneity and conciseness
Trajectory-Based Clustering

- Objective: Discover sub-trajectories that indicate common movement patterns of each class
- Algorithm: Extend the partition-and-group framework for classification purposes so that the class labels are incorporated into trajectory clustering
  - If an $\epsilon$-neighborhood contains trajectory partitions mostly of the same class, it is used for clustering; otherwise, it is discarded immediately
After trajectory-based clusters are found, highly discriminative clusters are selected for effective classification.

If the average distance from a specific cluster to other clusters of different classes is high, the discriminative power of the cluster is high.

- e.g.,

\[ C_1 \text{ is more discriminative than } C_2 \]
Overall Procedure of TraClass

1. Partition trajectories
2. Perform region-based clustering
3. Perform trajectory-based clustering
4. Select discriminative trajectory-based clusters
5. Convert each trajectory into a feature vector
   - Each feature is either a region-based cluster or a trajectory-based cluster
   - The $i$-th entry of a feature vector is the frequency that the $i$-th feature occurs in the trajectory
6. Feed feature vectors to the SVM
Classification Results

- **Datasets**
  - Animal: Three classes ← three species: elk, deer, and cattle
  - Vessel: Two classes ← two vessels
  - Hurricane: Two classes ← category 2 and 3 hurricanes

- **Methods**
  - *TB-ONLY*: Perform trajectory-based clustering only
  - *RB-TB*: Perform both types of clustering

- **Results**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Animal Version</th>
<th>Animal</th>
<th>Vessel Version</th>
<th>Vessel</th>
<th>Hurricane Version</th>
<th>Hurricane</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TB-ONLY</td>
<td>RB-TB</td>
<td>TB-ONLY</td>
<td>RB-TB</td>
<td>TB-ONLY</td>
<td>RB-TB</td>
</tr>
<tr>
<td><strong>Accuracy (%)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50.0</td>
<td>83.3</td>
<td></td>
<td>84.4</td>
<td>98.2</td>
<td>65.4</td>
<td>73.1</td>
</tr>
<tr>
<td><strong>Training Time (msec)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3542</td>
<td>2406</td>
<td></td>
<td>44683</td>
<td>22902</td>
<td>331</td>
<td>317</td>
</tr>
<tr>
<td><strong>Prediction Time (msec)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>98</td>
<td></td>
<td>722</td>
<td>608</td>
<td>48</td>
<td>46</td>
</tr>
</tbody>
</table>
Example: Extracted Features

Data (Three Classes)

Features:
- 10 Region-Based Clusters
- 37 Trajectory-Based Clusters

Accuracy = 83.3%
Part I. Moving Object Data Mining

- Introduction
- Movement Pattern Mining
- Periodic Pattern Mining
- Clustering
- Prediction
- Classification
- Outlier Detection
Trajectory Outlier Detection

- Task: Detect the trajectory outliers that are grossly different from or inconsistent with the remaining set of trajectories.

- Methods and philosophy:
  1. **Whole** trajectory outlier detection
     - A unsupervised method
     - A supervised method *based on classification*
  2. Integration with multi-dimensional information
  3. **Partial** trajectory outlier detection
     - A Partition-and-Detect framework
Outlier Detection: A Distance-Based Approach (Knorr et al. VLDBJ 00)

- Define the distance between two *whole* trajectories
  - A whole trajectory is represented by
    \[
    P = \begin{bmatrix}
    P_{\text{start}} \\
    P_{\text{end}} \\
    P_{\text{heading}} \\
    P_{\text{velocity}}
    \end{bmatrix}
    \]
    where
    \[
    P_{\text{start}} = (x_{\text{start}}, y_{\text{start}}) \\
    P_{\text{end}} = (x_{\text{end}}, y_{\text{end}}) \\
    P_{\text{heading}} = (\text{avg}_{\text{heading}}, \text{max}_{\text{heading}}, \text{min}_{\text{heading}}) \\
    P_{\text{velocity}} = (\text{avg}_{\text{velocity}}, \text{max}_{\text{velocity}}, \text{min}_{\text{velocity}})
    \]
  - The distance between two whole trajectories is defined as
    \[
    D(P_1, P_2) = \begin{bmatrix}
    D_{\text{start}}(P_1, P_2) \\
    D_{\text{end}}(P_1, P_2) \\
    D_{\text{heading}}(P_1, P_2) \\
    D_{\text{velocity}}(P_1, P_2)
    \end{bmatrix} \cdot \begin{bmatrix}
    w_{\text{start}} \\
    w_{\text{end}} \\
    w_{\text{heading}} \\
    w_{\text{velocity}}
    \end{bmatrix}
    \]
- Apply a distance-based approach to detection of trajectory outliers
  - An object $O$ in a dataset $T$ is a DB($p$, $D$)-outlier if at least fraction $p$ of the objects in $T$ lies greater than distance $D$ from $O$
Sample Trajectory Outliers

- Detect outliers from person trajectories in a room
A whole trajectory is encoded to a feature vector: \( \mathbf{F} = [ x, y, s(x), s(y), s(dx), s(dy), s(|d^2x|), s(|d^2y|) ] \)

- \( s() \) indicates a time smoothed average of the quantity
- \( dx = x_t - x_{t-1} \)
- \( d^2x = x_t - 2x_{t-1} + x_{t-2} \)

A self-organizing feature map (SOFM) is trained using the feature vectors of training trajectories, and a new trajectory is classified into novel (i.e., suspicious) or not novel

- *Supervised learning*
An Application: Video Surveillance

- Training dataset: 206 normal trajectories
- Test dataset: 23 unusual and 16 normal trajectories
- Classification accuracy: 92%

An example of a normal trajectory

An unusual trajectory; The unusual points are shown in black
Anomaly Detection (Li et al. ISI’06, SSTD’07)

- Automated alerts of abnormal moving objects
- Current US Navy model: manual inspection
  - Started in the 1980s
  - 160,000 ships
Conditional Anomalies and Motif Representations

- Raw analysis of collected data does not fully convey “anomaly” information
- More effective analysis relies on higher semantic features
- Examples:
  - A speed boat moving quickly in open water
  - A fishing boat moving slowly into the docks
  - A yacht circling slowly around landmark during night hours
- Motif representation

A sequence of motifs with motif attributes
Motif-Oriented Feature Space

- Each motif expression has attributes (e.g., speed, location, size, time)
- Attributes express how a motif was expressed
  - A right-turn at 30mph near landmark Y at 5:30pm
  - A straight-line at 120mph (!!!) in location X at 2:01am

Motif-Oriented Feature Space
- Naïve feature space
  1. Map each distinct motif-expression to a feature
  2. Trajectories become feature vectors in the new space
- Let there be $A$ attributes attached to every motif, each trajectory is a set of motif-attribute tuples
  \[
  \{(m_i, v_1, v_2, \ldots, v_A) \cup \ldots \cup (m_j, v_1, v_2, \ldots, v_A)\}
  \]
- Example:
  - Object 1: \{(right-turn, 53mph, 3:43pm)\} $\rightarrow$ (1, 0)
  - Object 2: \{(right-turn, 50mph, 3:47pm)\} $\rightarrow$ (0, 1)
Motif Feature Extraction

- Intuition: Should have features that describe general high-level concepts
  - “Early Morning” instead of 2:03am, 2:04am, …
  - “Near Location X” instead of “50m west of Location X”
- Solution: Hierarchical micro-clustering
  - For each motif attribute, cluster values to form higher level concepts
  - Hierarchy allows flexibility in describing objects
    - e.g., “afternoon” vs. “early afternoon” and “late afternoon”
Feature Clustering

- Rough, fast micro-clustering method based on BIRCH (SIGMOD’96)
- Extracts a hierarchy for every motif-attribute combination
- Trajectories can be represented at arbitrary level of granularity
Trajectory Outlier Detection: A Partition-and-Detect Framework (Lee et al. 08)

- Existing algorithms compare trajectories as a whole ➔ They might not be able to detect outlying portions of trajectories
  - e.g., $TR_3$ is not detected as an outlier since its overall behavior is similar to those of neighboring trajectories

The partition-and-detect framework is proposed to detect outlying sub-trajectories

$TR_5$  $TR_4$  $TR_3$  $TR_2$  $TR_1$

An outlying sub-trajectory
Usefulness of Outlying *Sub-Trajectories*

- Example: Sudden changes in hurricane’s path

Since Hurricane Charley (Aug. 2004) was expected to hit the land closer to Tampa, many residents around Punta Gorda, Fla., were caught unprepared.
Overall Procedure

- Two phases: **partitioning** and **detection**
A trajectory is partitioned *at a base unit*: the smallest meaningful unit of a trajectory in a given application.

- e.g., The base unit can be *every single point*

**Pros:** High detection quality in general

**Cons:** Poor performance due to a large number of t-partitions

Propose a two-level partitioning strategy
Two-Level Trajectory Partitioning

- **Objective**
  - Achieves much higher performance than the simple strategy
  - Obtains the same result as that of the simple strategy; *i.e.*, does not lose the quality of the result

- **Basic idea**
  1. Partition a trajectory in *coarse granularity* first
  2. Partition a coarse t-partition in *fine granularity only when necessary*

- **Main benefit**
  - Narrows the search space that needs to be inspected in fine granularity ⇒ Many portions of trajectories can be *pruned* early on
Intuition of Two-Level Trajectory Partitioning

- If the distance between coarse t-partitions is very large (or small), the distances between their fine t-partitions are also very large (or small)

- The lower and upper bounds for fine t-partitions are derived in the paper
Outlier Detection

- Once trajectories are partitioned, trajectory outliers are detected based on both distance and density.

- An *outlying t-partition* is defined as:

  \[ \begin{align*}
  \text{Not close:} & \quad L_i \text{ is an outlying t-partition} \\
  \text{Close:} & \quad L_i \text{ is not an outlying t-partition}
  \end{align*} \]

- A trajectory is an *outlier* if it contains a sufficient amount of outlying t-partitions.
Incorporation of Density

- The number of close trajectories is adjusted by the density of a t-partition

- Dense region ➔ Decreased, Sparse region ➔ Increased
Experiments: Sample Detection Results

13 outliers detected from the hurricane data

Three outliers found from the Elk Data
Summary: Moving Object Mining

- Pattern Mining
  - Trajectory patterns, flock and leadership patterns, periodic patterns,

- Clustering
  - Probabilistic method, density-based method, partition-and-group framework

- Prediction
  - linear/non-linear model, vector-based method, pattern-based method

- Classification
  - Machine learning-based method, HMM-based method, TraClass using collaborative clustering

- Outlier Detection
  - Unsupervised method, supervised method, partition-and-detect framework
Tutorial Outline

- Part I. Mining Moving Objects
- Part II. Mining Traffic Data
- Part III. Conclusions
Part II. Traffic Data Mining

- Introduction to Traffic Data
- Traffic Data Warehousing
- Route Discovery by Frequent Path Pattern Analysis
Trillion Miles of Travel

- MapQuest
  - 10 billion routes computed by 2006
- GPS devices
  - 18 million sold in 2006
  - 88 million by 2010
- Lots of driving
  - 2.7 trillion miles of travel (US – 1999)
  - 4 million miles of roads
  - $70 billion cost of congestion, 5.7 billion gallons of wasted gas
Abundant Traffic Data

- Google Maps provides live traffic information
Traffic Data Gathering

- Inductive loop detectors
  - Thousands, placed every few miles in highways
  - Only aggregate data
- Cameras
  - License plate detection
- RFID
  - Toll booth transponders
  - 511.org – readers in CA
Road Networks

Driving Pattern: Preferred Routes

Node: Road intersections

Edge: Road segment

Speed Pattern:
- 65 mph non rush
- 35 mph rush hour
Part II. Traffic Data Mining

- Introduction to Traffic Data
- Traffic Data Warehousing
- Route Discovery by Frequent Path Pattern Analysis
Traffic Cube: Motivation Examples

- Ex. 1. Bob is a backpack traveler and he is new to Los Angeles. He wants to know
  - Where are the places the traffic jams are most likely to happen in weekend?
  - When is the best time to visit the Hollywood to avoid heavy traffic?

- Ex. 2. Jim is the head of the transportation department in Los Angeles, the department recently got limited funds to improve roads
  - On which highway the traffic is usually heavy during the morning rush hours?
Problem of Traditional Query

- **Select** highway name **from** traffic table **where** speed < 40 mph and Region is Los Angeles
  - #101, segment id 2, 36 mph, 3:50 pm
  - #10, segment id 5, 33 mph, 3:50 pm
  - # 101, segment id 3, 34 mph, 4:00 pm

  - The results are not organized
  - Too trivial

- Google Traffic is good to visualize current traffic, it also provides prediction function, but no analysis on historical data
User Requirements

- Users demand summaries in their interested time, region, scale, etc.
  - Bob is only interested in Hollywood region on weekend
  - Jim is more concerned on the whole Los Angeles on weekdays
- Users are more interested in the information related to traffic jams, incidents, slow traffic—the congestions
Features of Traffic Data

- Huge Size
  - Thousands of road sensors, reporting the data in a time frequency of 30 seconds
  - The traffic databases contain Giga-bytes, even Tera-bytes of data
  - Most of them are normal records (the speed reading is close to the speed limit of the road)
  - Congestions are dwarfed by normal data

- Complex Object
  - A congestion is a complex object with several road segments and varied time length – hard to model
Traffic Monitoring Systems

- **PeMS**: collects data in California highway
- **CarWeb**: collects real time GPS data from cars
- **Google Traffic**: Toolkit on Google Map
- **CubeView** by Shekhar et al: Implement traditional OLAP on the traffic data
- **AITVS**: based on CubeView, using two more distinct views to support investigation
- Most focus on **SQL based queries**, lacking analysis power
- Build on the whole dataset – **huge I/O** overhead, atypical data are dwarfed
Spatial/ Traffic/ Trajectory Data Cubes

- **Spatial Cube** (Stefanovic et al. 2000)
  - Dimension members are spatially referenced and can be represented on a Map

- **Trajectory Cube** (Giannotti et al. 2007)
  - Include temporal, spatial, demo-graphic and techno-graphic dimensions, two kinds of measures: spatial measure and numerical measure

- **Flow Cube** (Gonzalez et al. 2007)
  - Analyzing item flows in RFID applications

- **Congestion Cube**: On-going work
  - Multidimensional analysis of traffic congestions
A congestion is a dynamic process:
- start from a single segment of the streets
- expand along the road and influence nearby roads
- may cover hundred road segments when reaching the full size
- As time passes by, those fragments shrink slowly and eventually disappear.

Group the congestion records that are spatially close and timely relevant to be a congestion event.
Base Congestion Cluster

- For each road segment in the congestion event, record the seg_id, total_duration and avg_speed

- Congestion Event:
  - Seg_1, 9:00 am, 30 mph
  - Seg_2, 9:00 am, 35 mph
  - Seg_1, 9:05 am, 40 mph
  - ...

- Base Congestion Cluster
  - Seg_1, 30 mins, 32.5 mph
  - Seg_2, 20 mins, 35 mph
Congestion Cluster and Congestion Cube

- Natural and distinguishable
- Congestion cube: Constructed based on congestion clusters
Part II. Traffic Data Mining

- Introduction to Traffic Data
- Traffic Data Warehousing
- Route Discovery by Frequent Path Pattern Analysis
Route Planning
Heuristic Shortest Path Algorithms for Transportation Networks (Fu et al.’06)

- **Problem**
  - Route Guidance System (RGS)
  - Route Planning System (RPS)
  - Second level response, queries on large networks

- **Solution: Heuristic search**
  1. Limit Search Area
  2. Search Decomposition
  3. Limit Examined Links
General Algorithm

Initialization
- o: Origin, d: Dest
  - L(i) = inf, L(o) = 0
  - Q.push(o)

For each link (i,j)
  - If L(i) + C(i,j) < L(j)
    - L(j) = L(i) + C(i,j)
    - Q.push(j)

Label Setting (LS):
  - return min(L(i)) node

Label Correcting (LC):
  - return first node
Method 1. Limit Search Area

- Branch Pruning [Fu et al. 96, Karim 96]
  
  **Q.select_node()**
  
  - Select node if $L(i) + e(i,d) < E(o,d)$
  - Prune unpromising nodes
  - Expands: 20% of LS
- A-Star [Hart 68, Nilsson 71]
  
  - Select node with min $L(i) + e(i,d)$
  - Prioritize promising nodes
  - Expands: 10% of LS
- Branch Pruning vs. A-Star
Method 2. Search Decomposition

- Algorithm cost grows faster than graph size
- Decompose problem
  - Subgoals
    - [Bander et al. 91]
    - [Dillenburg et al. 95]
  - Bidirectional Search
    - [Dantzig 60, Nicholson 66]
Method 3. Limit Links: Hierarchical Search

- Divide a graph into layers
  - Top Layer:
    - Small
    - Large roads
  - Bottom Layer:
    - Large
    - Contains all roads
- Strategy
  - Search an entry point to top layer
  - Run the standard A-Star on top layer
- Orders of magnitude faster than A-star
Hierarchical Search

- Speedup
  - Linear in search space
  - Orders of magnitude faster than A-Star
  - Only viable option

- Sub-optimal solution
  - Path 9% - 50% slower than optimal
  - No shortcuts between top/bottom layers
  - Bad for short trips
    - Better to avoid highways
Mining Frequent Routes: When in Rome, Do as the Romans Do

Adaptive Fastest Path Computation on a Road Network [Hector et al. 07]

- Fastest Route
- Frequent Route

- Data is the King
- No model can anticipate all variables
Mining for Fast and Popular Routes

Query:
- Start: Node A
- End: Node B
- Time: T

Route Planning

Speed Patterns
Driving Patterns

Road Network

Driving Conditions
Forecast

New

Fast but also Popular
Road Network Partitioning

- Road hierarchy provides natural partition

![Road hierarchy levels]

- Highway Country Level
- Arterial Roads City Level
- Collector Roads Block Level
## Traffic Data

<table>
<thead>
<tr>
<th>car_id</th>
<th>eid</th>
<th>Time</th>
<th>Speed</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
<td>30</td>
<td>rain, no construction, accident</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>12</td>
<td>25</td>
<td>rain, no construction, no accident</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>11</td>
<td>60</td>
<td>good weather, no construction, accident</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **RFID:** Yes
- **Loop Sensor:** No
- **Computed Successive Observations**
- **National Weather Service Transportation Management Centers**
Mining Speed Patterns

- Model of speed changes
- Classification
  - Given conditions predict speed (discrete)
  - We use decision tree
- Regression problem
  - Given conditions predict speed (continuous)
Mining Driving Patterns

For each area:
- Define minimum support
- Mine frequent trajectories
  - Length $k$: For RFID Data
  - Length 1: For loop detectors

<table>
<thead>
<tr>
<th>Driving Conditions</th>
<th>Area</th>
<th>Popular Routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>$A_1$</td>
<td>{r1,r3,r5}</td>
</tr>
<tr>
<td>Snow</td>
<td>$A_2$</td>
<td>{r13,r29}</td>
</tr>
<tr>
<td>Rush Hour, Any Weather</td>
<td>$A_3$</td>
<td>{r6,r6}</td>
</tr>
</tbody>
</table>
Hierarchical Route Planning

- **Ascending Phase**
  - Move from start through successively bigger roads towards goal
  - Move only through large roads
- **Descending Phase**
  - Move towards goal through successively smaller roads
- At each step
  - Select frequently traveled roads
  - Use dynamic speed model
Example: Hierarchical Route Planning
Result: San Joaquin Route

Shortest path
Suggested path
Summary: Traffic Data Mining

- Traffic Warehousing
  - Congestion cluster based approach
  - Heuristic methods to mine significant clusters
- Route Discovery
  - Heuristic methods (Limit Search Area, Search Decomposition, Limit Examined Links), adaptive method
- Hot-Route Detection
  - FlowScan
Tutorial Outline

- Part I. Mining Moving Objects
- Part II. Mining Traffic Data
- Part III. Conclusions
Conclusions

- Mining moving object data, trajectory data and traffic data are important tasks in data mining
  - Lots of rich and exciting results
- This tutorial has presented an overview of recent approaches in this direction
- Promising research directions
  - Moving object/traffic mining in cyber-physical networks
  - Integration with heterogeneous information networks
  - Exploration of diverse applications
References: Moving Object Databases and Queries

- N. Jing, Y.-W. Huang, and E. A. Rundensteiner. Hierarchical optimization of optimal path finding for transportation applications. *CIKM'96*.
- L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. *AAAI'04*. 
References on Moving Object Pattern Mining (I)

- J. Gudmundsson and M. J. van Kreveld. Computing longest duration flocks in trajectory data. *GIS’06*.
References on Moving Object Pattern Mining (II)

- Y. Li, J. Han, and J. Yang. Clustering moving objects. *KDD'04*.
- Z. Li, et al., “MoveMine: Mining Moving Object Databases”, SIGMOD’10 (system demo)
- Z. Li, B. Ding, J. Han, and R. Kays, “Swarm: Mining Relaxed Temporal Moving Object Clusters”, in submission
- Z. Li, B. Ding, J. Han, and R. Kays, “Mining Hidden Periodic Behaviors for Moving Objects”, in submission
References on Outlier Detection

- E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. *UAI'05*
- J.-G. Lee, J. Han, and X. Li, "Trajectory Outlier Detection: A Partition-and-Detect Framework", ICDE 2008
- J.-G. Lee, J. Han, and K.-Y. Whang, “Trajectory Clustering: A Partition-and-Group Framework”, SIGMOD'07
- X. Li, J. Han, S. Kim, "Motion-alert: Automatic anomaly detection in massive moving objects", ISI 2006
- X. Li, J. Han, S. Kim, and H. Gonzalez, “ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets”, SDM'07
References on Prediction and Classification

- J.-G. Lee, J. Han, X. Li, and H. Gonzalez, “TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering”, VLDB 2008.
- Anna Monreale, Fabio Pinelli, Roberto Trasarti, Fosca Giannotti: WhereNext: a location predictor on trajectory pattern mining. KDD 2009
- Y. Tao, C. Faloutsos, D. Papadias, B. Liu: Prediction and Indexing of Moving Objects with Unknown Motion Patterns. SIGMOD 2004
Reference on Traffic Mining

- H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, “Adaptive Fastest Path Computation on a Road Network: A Traffic Mining Approach”, VLDB'07.
- X. Li, J. Han, J.-G. Lee, and H. Gonzalez, “Traffic Density-based Discovery of Hot Routes in Road Networks”, SSTD'07.