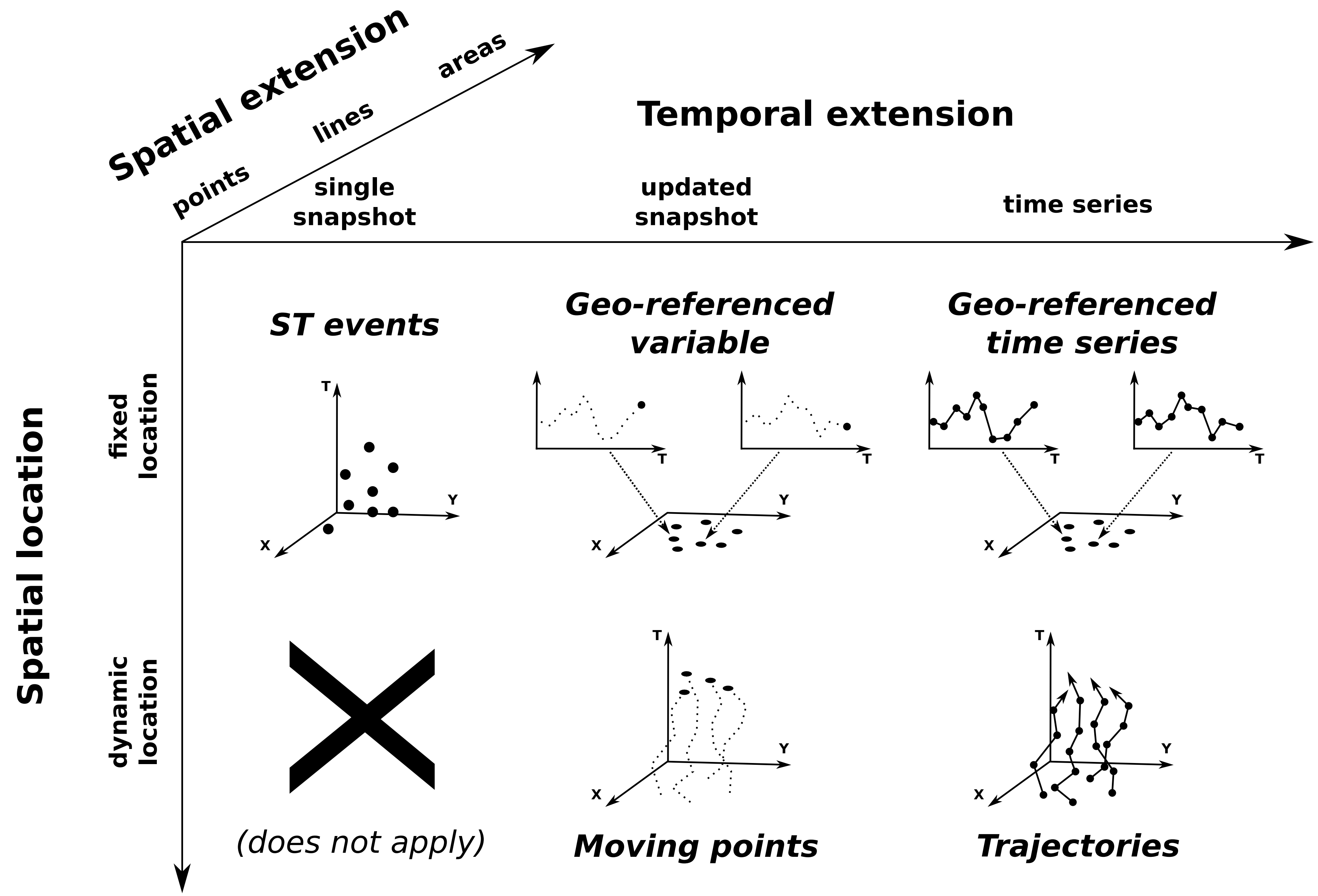
Trajectory Pattern Mining

Figures and charts are from some materials downloaded from the internet.

Outline

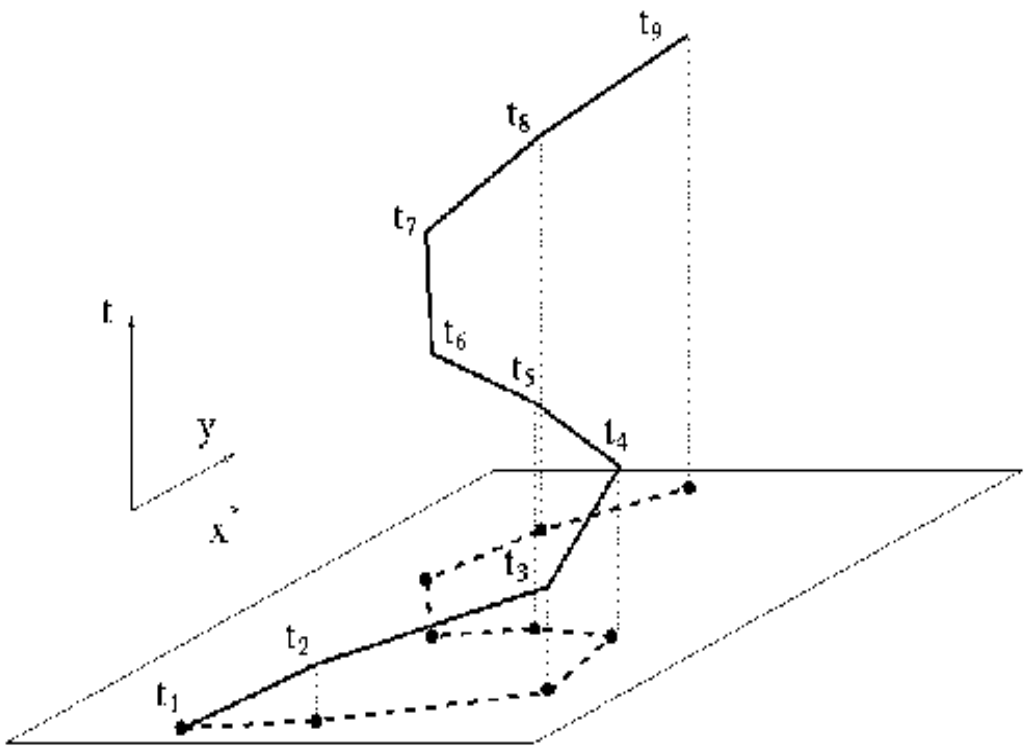
- Spatio-temporal data types
- Mining trajectory patterns

Spatio-temporal data types



Trajectory data

- **Spatio-temporal** Data
- Represented as a set of points, located in space and time
- T=(x1,y1, t1), ..., (xn, yn, tn) => position in space at time ti was (xi,yi)

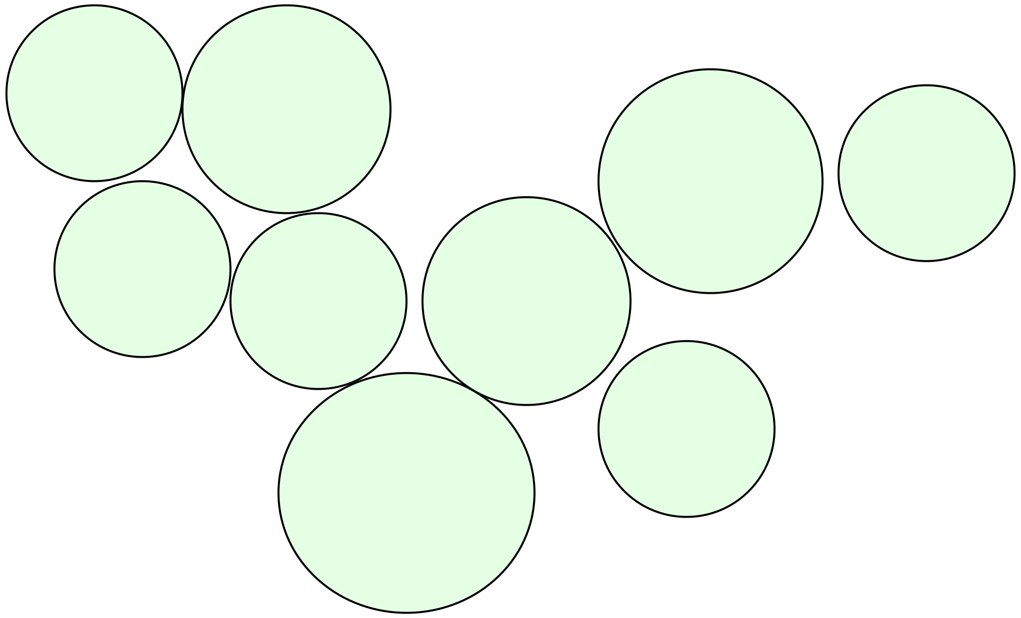


Tid	position (x,y)		time (t)
1	48.890018	2.246100	08:25
1	48.890018	2.246100	08:26
...	...		...
1	48.890020	2.246102	08:40
1	48.888880	2.248208	08:41
1	48.885732	2.255031	08:42
...	...		...
1	48.858434	2.336105	09:04
1	48.853611	2.349190	09:05
...	...		...
2	...		...

Analyzing trajectory patterns

- **Clustering**

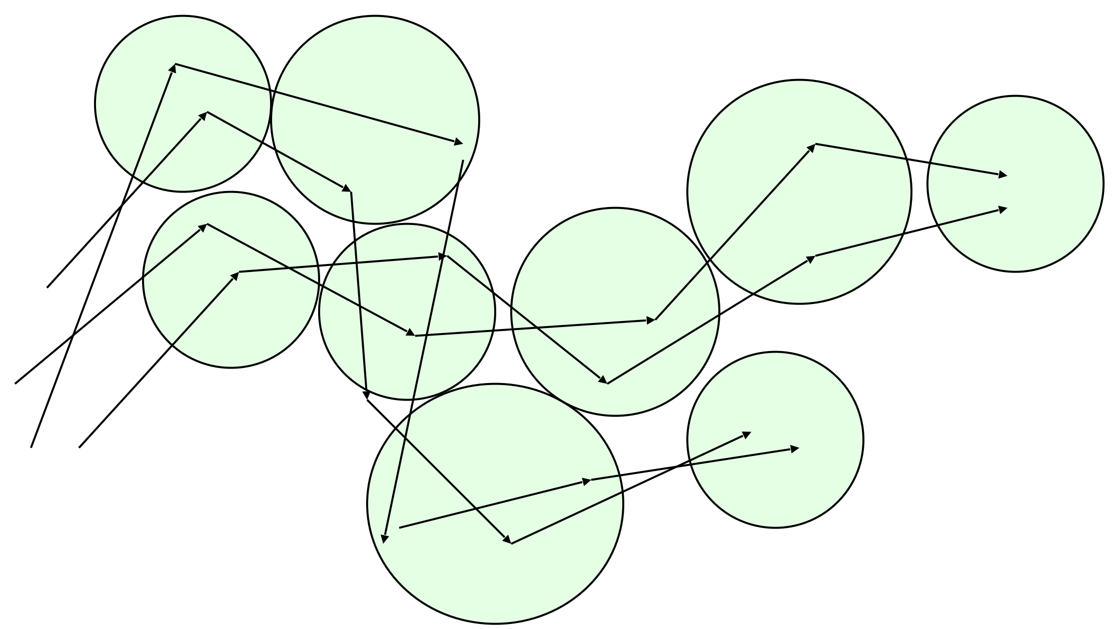
- Group together similar trajectories
- Analyze the each group



Analyzing trajectory patterns

- **Clustering**

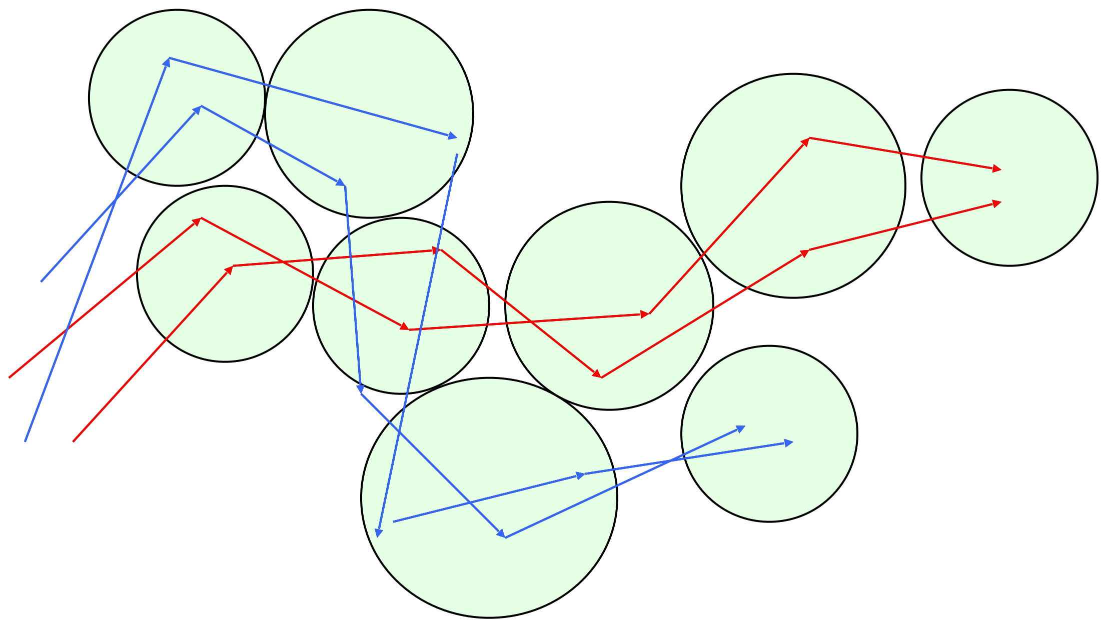
- Group together similar trajectories
- Analyze the each group



Analyzing trajectory patterns

- **Clustering**

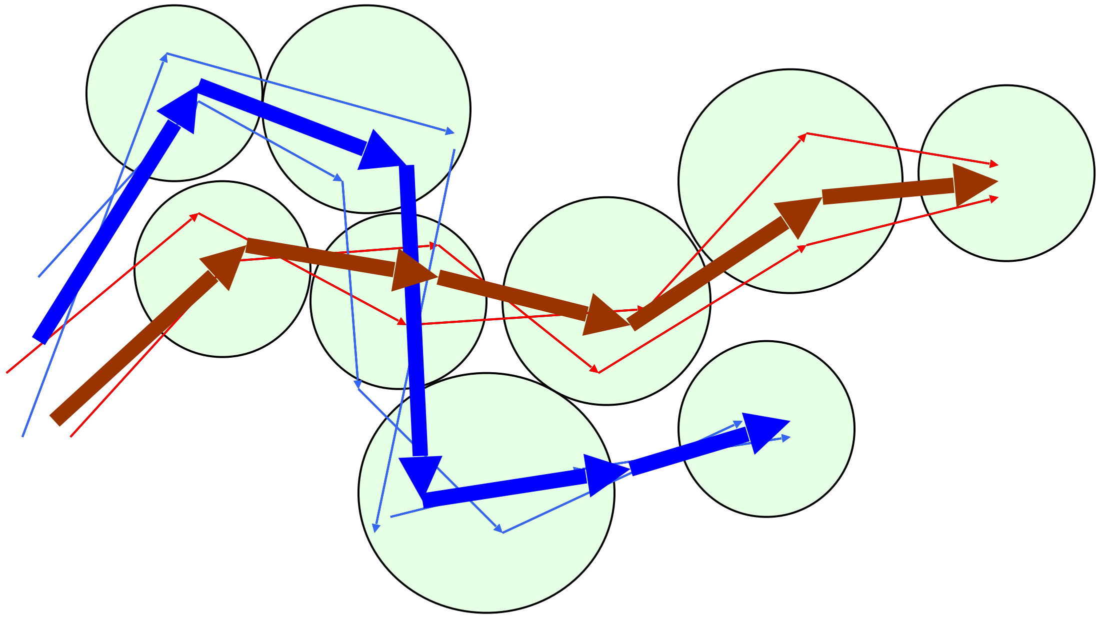
- Group together similar trajectories
- Analyze the each group



Analyzing trajectory patterns

- **Clustering**

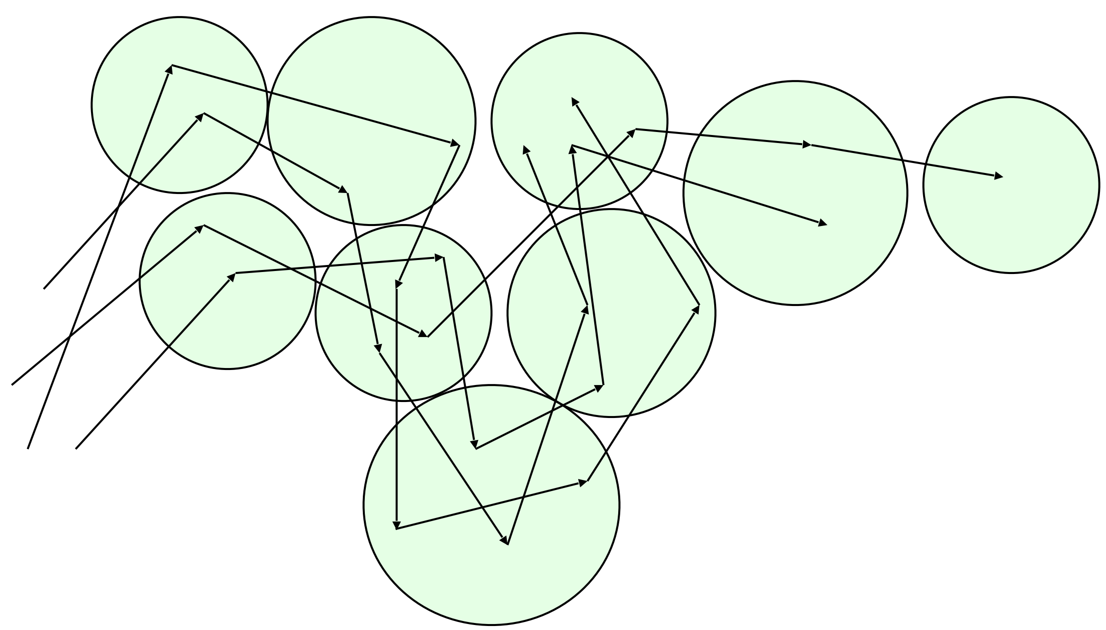
- Group together similar trajectories
- Analyze the each group



Analyzing trajectory patterns

- **Frequent trajectory pattern mining**

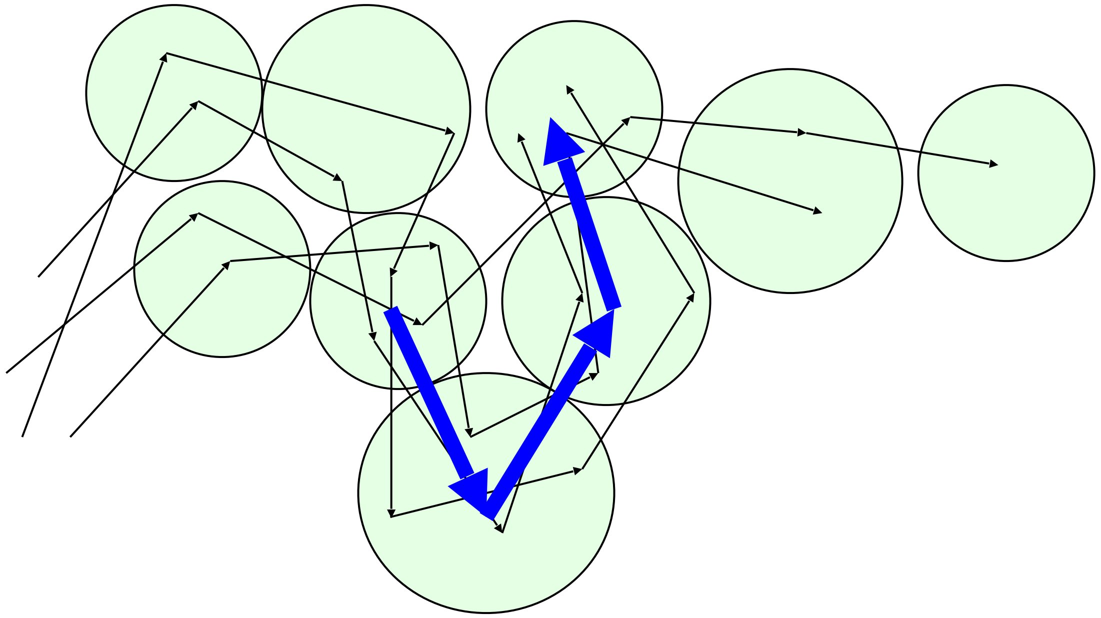
- The sub-trajectory that frequently appeared in the trajectories
- Consider both place and time



Analyzing trajectory patterns

- **Frequent trajectory pattern mining**

- The sub-trajectory that frequently appeared in the trajectories
- Consider both place and time



Trajectory Pattern Mining

Fosca Giannotti, etc. 2007

trajectory vs sequence

- **Purpose**: find the frequent trajectory in the dataset.

trajectory vs sequence

- **Purpose**: find the frequent trajectory in the dataset.
  - Difference Compared with Traditional Sequence**: TIME**

trajectory vs sequence

- **Purpose**: find the frequent trajectory in the dataset.

    - Difference Compared with Traditional Sequence**: TIME**

- **Temporally Annotated Sequences(TAS)**
- sequences with transition times between their elements

$$T = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} s_n$$

- represented as a couple T = (S, A) of a sequence S = $\langle s_0,\ldots,s_n\rangle$ with temporal annotations A = $\langle \alpha_1,\ldots,\alpha_n\rangle$.

Frequent sequences in TAS vs Frequent sequence in sequence

- **contain of Sequence**
- α⊆ β, if there exist integers 1≤j1<j2<…<jn≤m such that a1⊆bj1, a2 ⊆ bj2,…, an ⊆ bjn .

- Example:
- β =<a(abc)(ac)d(cf)>
- α1=<aa(ac)d(c)> yes
- α2=<(ac)(ac)d(cf)> yes
- α3=<ac> yes
- α4=<df(cf)> no

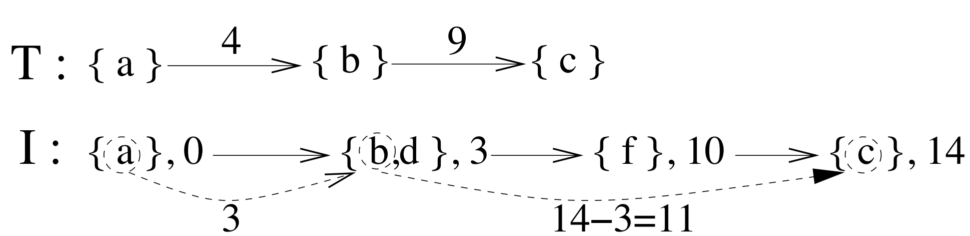
- **contain of TAS**

Definition 1. (τ-containment ($\preceq_\tau$)) Given a time threshold τ, a *TAS* $T = s_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} s_n$ is τ*-contained* (or *occurs*) in an input sequence $I = \langle (I_0, t_0), \ldots, (I_m, t_m) \rangle$, denoted as $T \preceq_\tau I$, if and only if there exists a sequence of integers $0 \leq i_0 < \cdots < i_n \leq m$ such that:

1. $\forall_{0 \leq k \leq n}.\ s_k \subseteq I_{i_k}$
2. $\forall_{1 \leq k \leq n}.\ |\alpha_k - \alpha'_k| \leq \tau$

where $\forall_{1 \leq k \leq n}.\ \alpha'_k = t_{i_k} - t_{i_{k-1}}$.

- Example:



Frequent sequences in TAS vs Frequent sequence in sequence

- **contain of Sequence**
- α⊆ β, if there exist integers 1≤j1<j2<…<jn≤m such that a1⊆bj1, a2 ⊆ bj2,…, an ⊆ bjn .

- Example:
- β =<a(abc)(ac)d(cf)>
- α1=<aa(ac)d(c)> yes
- α2=<(ac)(ac)d(cf)> yes
- α3=<ac> yes
- α4=<df(cf)> no

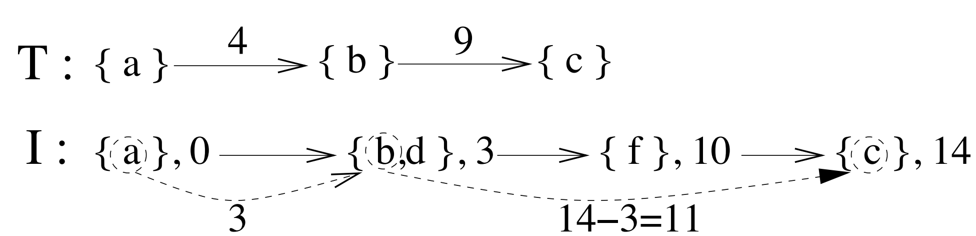
- **contain of TAS**

Definition 1. (τ-containment ($\preceq_\tau$)) Given a time threshold τ, a TAS $T = s_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} s_n$ is τ*-contained* (or *occurs*) in an input sequence $I = \langle (I_0, t_0), \ldots, (I_m, t_m) \rangle$, denoted as $T \preceq_\tau I$, if and only if there exists a sequence of integers $0 \leq i_0 < \cdots < i_n \leq m$ such that:

1. $\forall_{0 \leq k \leq n}.\ s_k \subseteq I_{i_k}$
2. $\forall_{1 \leq k \leq n}.\ |\alpha_k - \alpha'_k| \leq \tau$

where $\forall_{1 \leq k \leq n}.\ \alpha'_k = t_{i_k} - t_{i_{k-1}}$.

- Example:



Generalize the concept of contain with neighbourhood

Problem statement

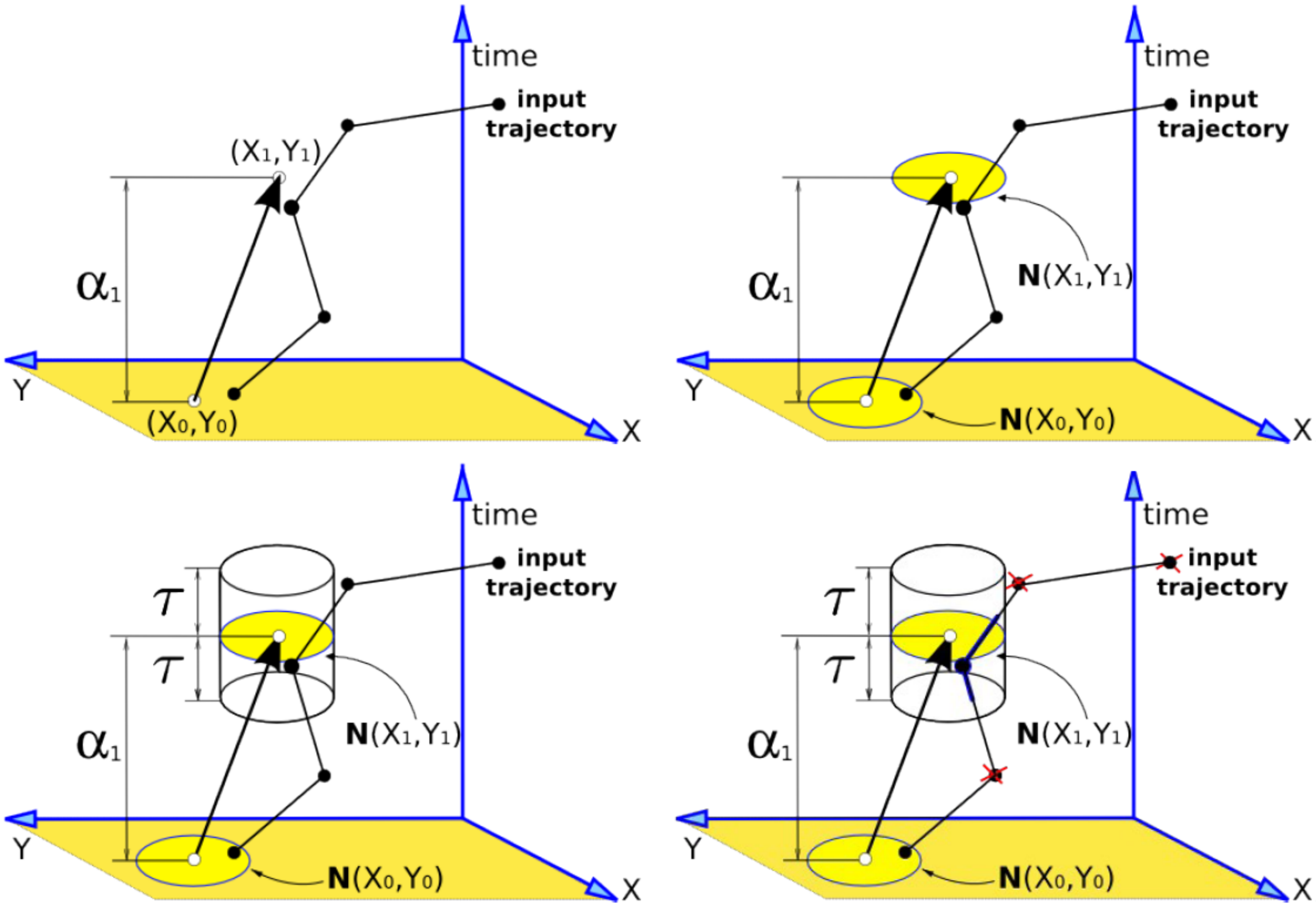
Definition 3. (Spatial containment, $\preceq_N$) Given a sequence of spatial points $S = \langle (x_0, y_0), \ldots, (x_k, y_k) \rangle$, a spatio-temporal sequence $T = \langle (x'_0, y'_0, t'_0), \ldots, (x'_n, y'_n, t'_n) \rangle$ and a neighborhood function $N : \mathbf{R}^2 \rightarrow \mathcal{P}(\mathbf{R}^2)$, we say that S *is contained in* T ($S \preceq_N T$, or simply $S \preceq T$, when N is clear from context) if and only if there exists a sequence of integers $0 \leq i_0 < \cdots < i_k \leq n$ such that: $\forall_{0 \leq j \leq k}.\ (x_j, y_j) \in N(x'_{i_j}, y'_{i_j})$.

Definition 5. (Spatio-temporal containment, $\preceq_{N,\tau}$) Given a spatio-temporal sequence T, time tolerance τ, a neighborhood function $N : \mathbf{R}^2 \rightarrow \mathcal{P}(\mathbf{R}^2)$ and a T-pattern $(S, A) = (x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} (x_k, y_k)$, we say that (S, A) *is contained in* T ($(S, A) \preceq_{N,\tau} T$, or simply $(S, A) \preceq T$, when clear from context) if and only if there exists a subsequence T' of T, $T' = \langle (x'_0, y'_0, t'_0), \ldots, (x'_k, y'_k, t'_k) \rangle$ such that:

1. $S \preceq_N T'$, and
2. $\forall_{1 \leq j \leq k}.\ |\alpha_j - \alpha'_j| \leq \tau$

where $\alpha'_j = t'_j - t'_{j-1}$.

Illustration of patio-temporal containment



Mining trajectory patterns with different N()

- different approaches correspond to a different choice of the neighborhood function N(x,y).

- RoI: (1) known RoI; (2) unknown RoI.

- Generalized N(x,y)

Mining trajectory patterns with known RoI

- Add the information of time into PrefixSpan — mining TAS

"Efficient mining of sequences with temporal annotations. In Proc. SIAM Conference on Data Mining, pages 346–357. SIAM, 2006."

PrefixSpan

- PrefixSpan: mining frequent sequences based on prefix-projection

- Sequence β is called a prefix of α iff:
- (1)bi= ai for i ≤ m-1; (2) bm ⊆ am.

- **e.g**. α =<a(abc)(ac)d(cf)>
- β =<a(abc)a>

- A subsequence α' of sequence α is called a projection of α w.r.t. β prefix iff:(1) α' has prefix β; (2) There exist no proper super-sequence α'' of α' such that α'' is a subsequence of α and also has prefix β.

- **e.g**. α =<a(abc)(ac)d(cf)>
- β =<(bc)a>
- α' =<(bc)(ac)d(cf)>

- The fundamental idea is that any pattern starting with a can be obtained by analyzing only D|a (projection of the initial dataset D on a), which in general is much smaller than D.

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

- Step two: Divide search space

- Step three: Find subsets of sequential patterns

“J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001.”

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	*1*

- Step two: Divide search space

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

Prefix

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

Prefix

<a>
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

Prefix

<a>
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<b>
<(_c)(ac)d(cf)>
<(_c)(ae)>
<(df)cb>
<c>

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

Prefix

<a>
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

<b>
<(_c)(ac)d(cf)>
<(_c)(ae)>
<(df)cb>
<c>

<c>
<(ac)d(cf)>
<(bc)(ae)>
<b>
<bc>

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

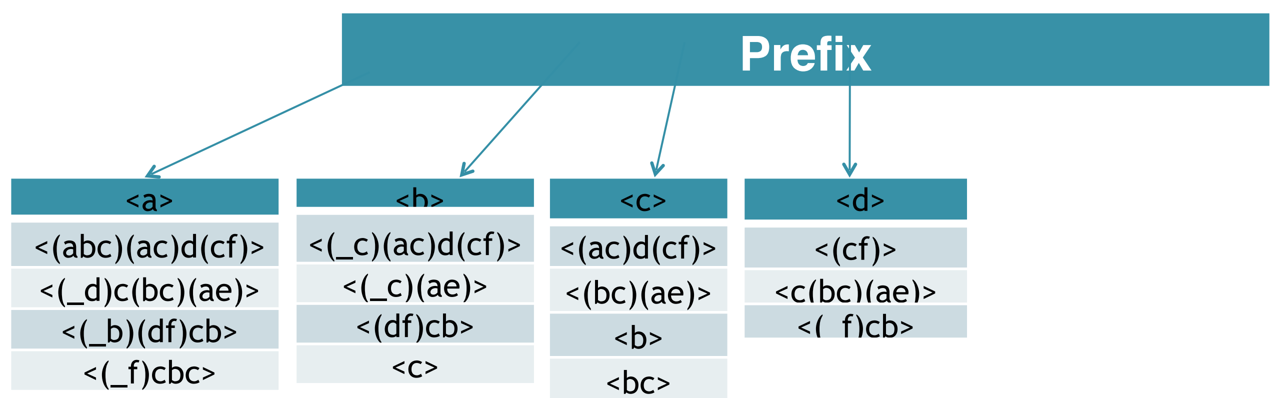
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

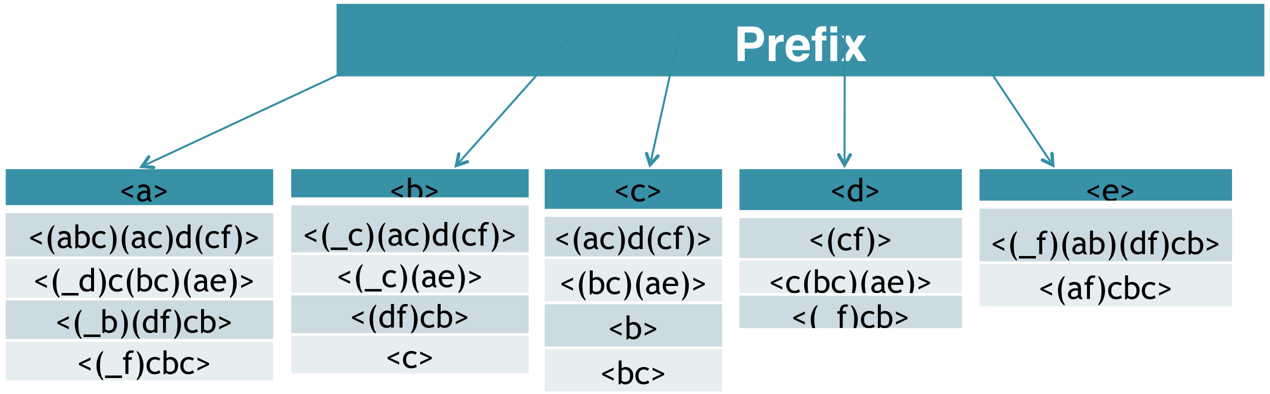
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space

Prefix

<a>	<b>	<c>	<d>	<e>	<f>
<(abc)(ac)d(cf)>	<(_c)(ac)d(cf)>	<(ac)d(cf)>	<(cf)>	<(_f)(ab)(df)cb>	<(ab)(df)cb>
<(_d)c(bc)(ae)>	<(_c)(ae)>	<(bc)(ae)>	<c(bc)(ae)>	<(af)cbc>	<cbc>
<(_b)(df)cb>	<(df)cb>	<b>	<(_f)cb>		
<(_f)cbc>	<c>	<bc>			

- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

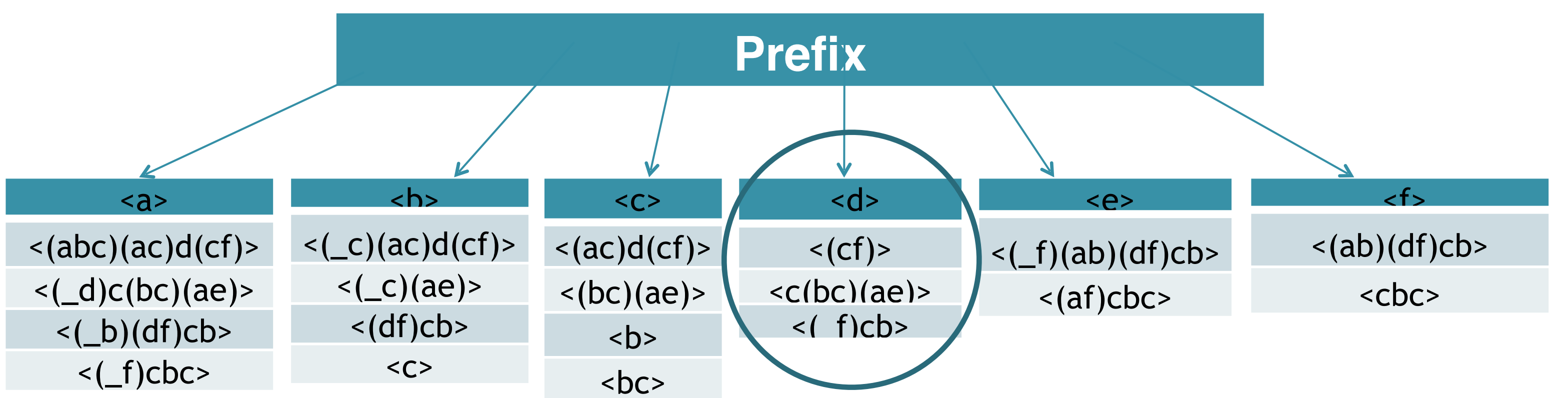
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns

"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

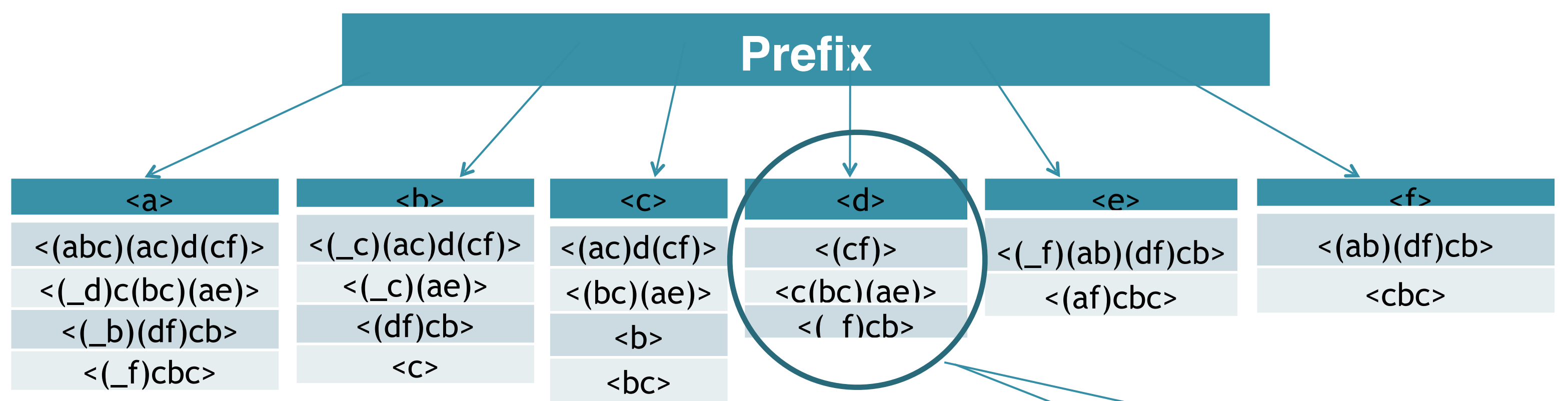
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

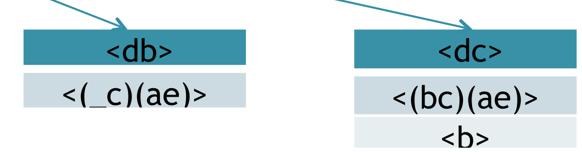
<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns



"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

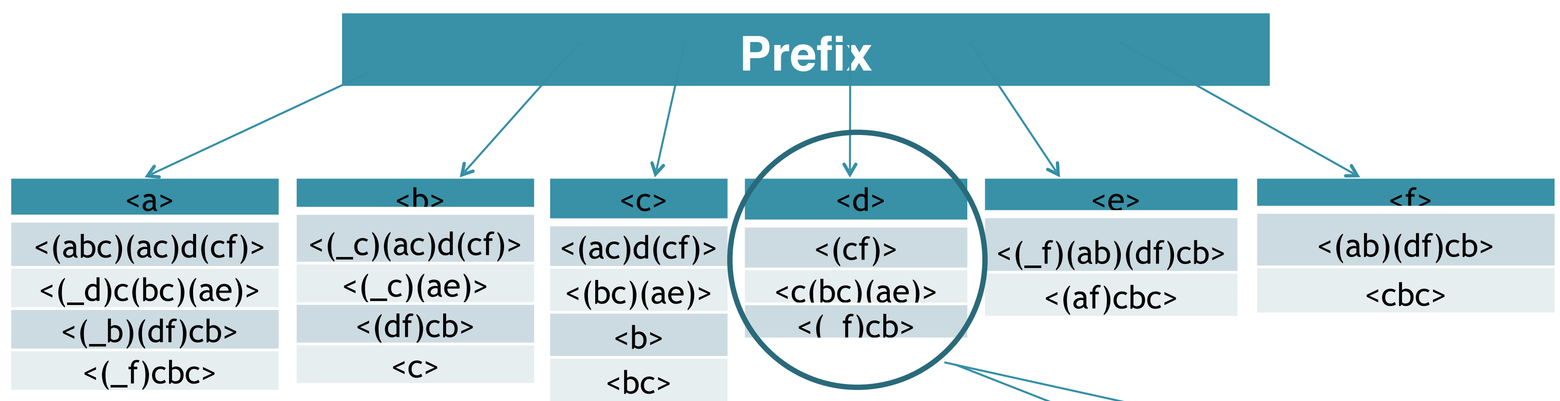
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

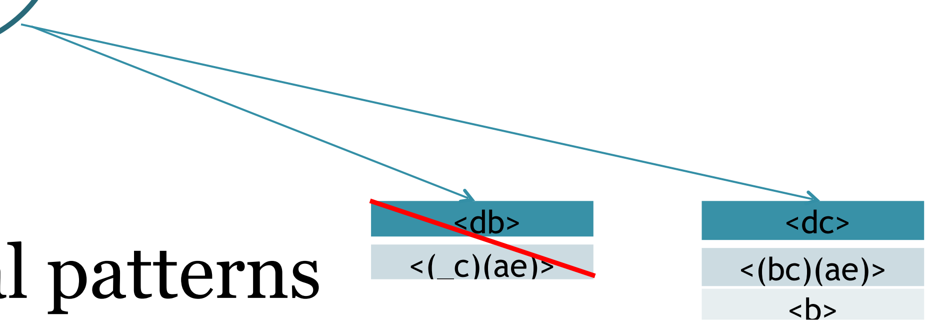
<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns



"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

PrefixSpan — examples

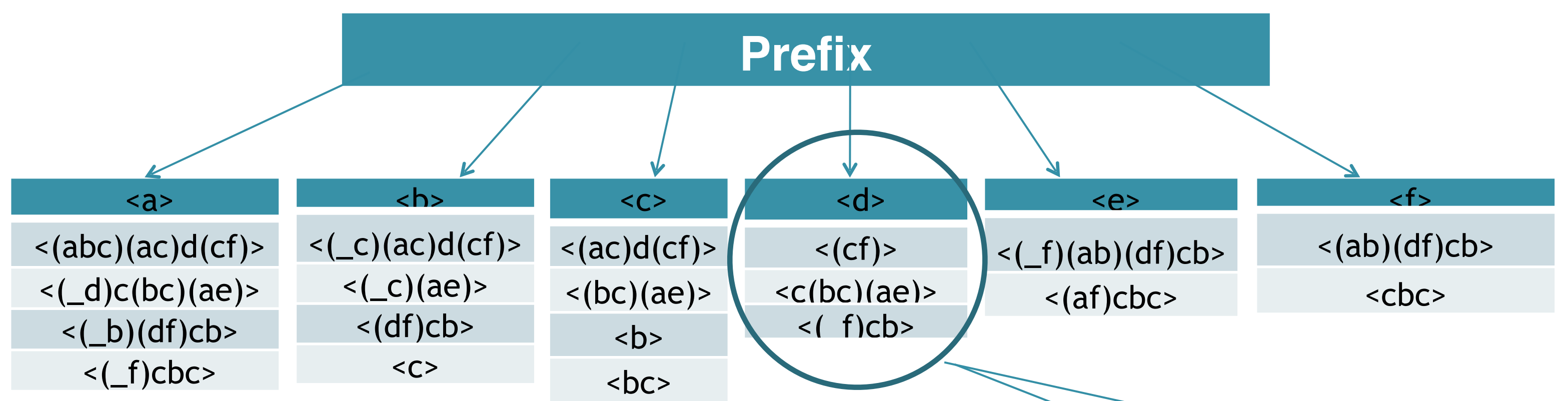
- Step one: Find length1 sequential patterns

id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

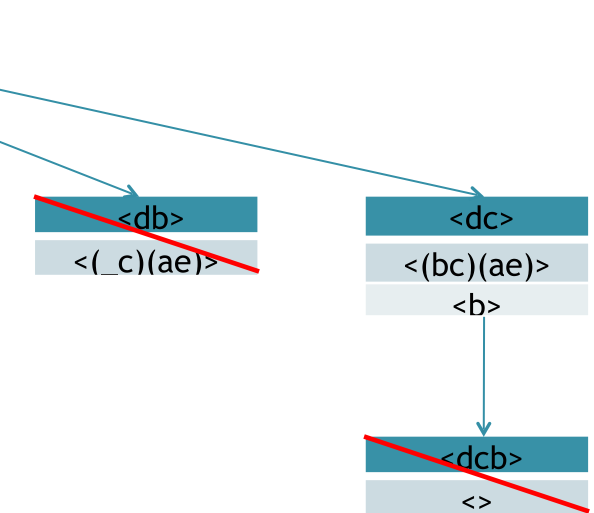
<a>	<b>	<c>	<d>	<e>	<f>	~~<g>~~
4	*4*	*4*	*3*	*3*	*3*	~~*1*~~

<a><b><c><d><e><f>

- Step two: Divide search space



- Step three: Find subsets of sequential patterns



"J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In ICDE, 2001."

Mining trajectory patterns with known RoI

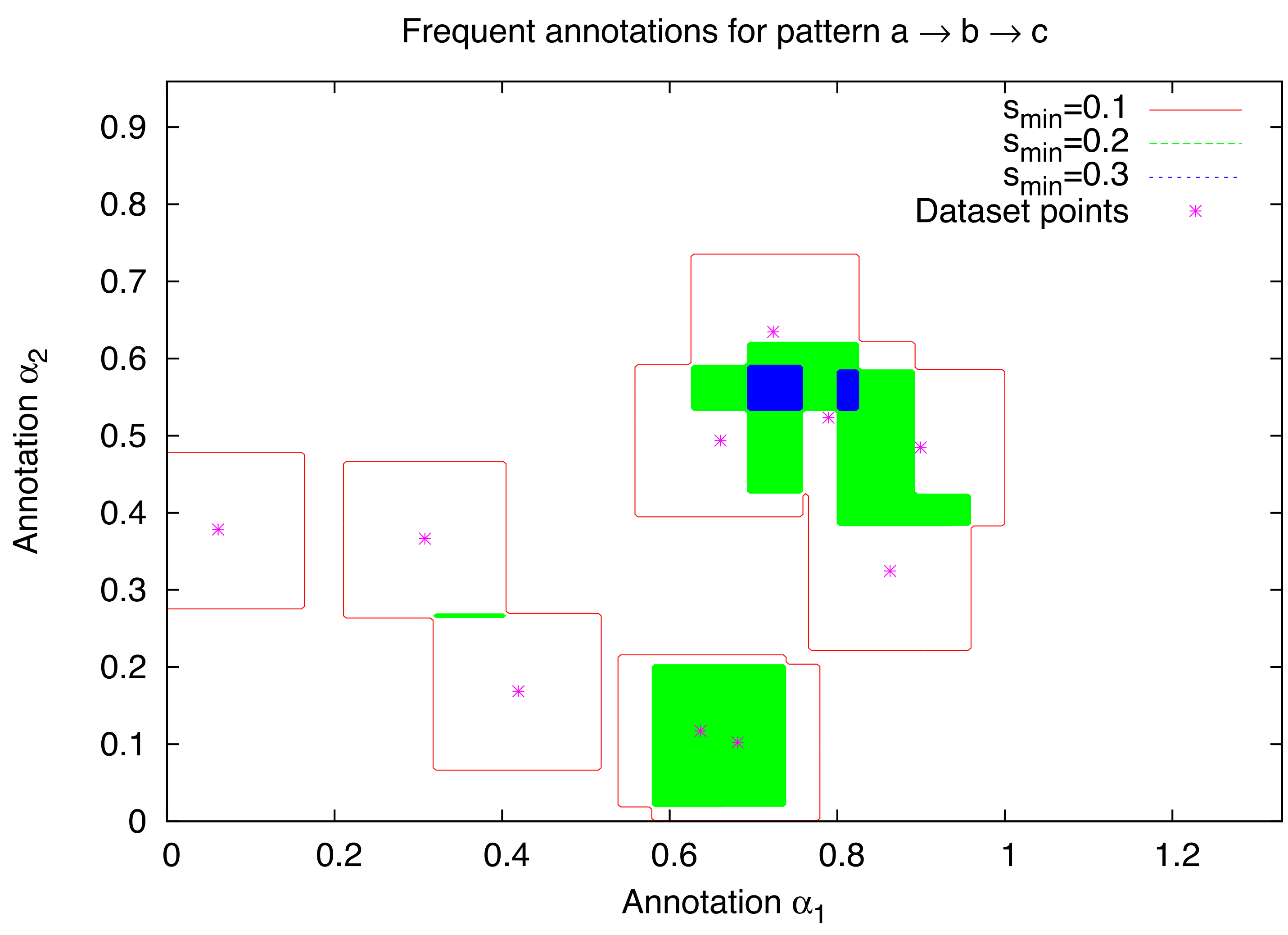
- Add **time** into PrefixSpan — mining TAS

- Expand the prefix with the limitation of the time dimension

"Efficient mining of sequences with temporal annotations. In Proc. SIAM Conference on Data Mining, pages 346–357. SIAM, 2006."

Mining trajectory patterns with known RoI

- Add **time** into PrefixSpan — mining TAS
- Expand the prefix with the limitation of the time dimension



"Efficient mining of sequences with temporal annotations. In Proc. SIAM Conference on Data Mining, pages 346–357. SIAM, 2006."

Mining trajectory patterns with unknown RoI

- Construct RoI
- Use the previous algorithm

Construct RoI (1)

- Split space into n x m grid with small cells
- Increment cells where trajectory passes
- Neighborhood Function NR() determines which surrounding cells
- Regression - increment continuously along trajectory

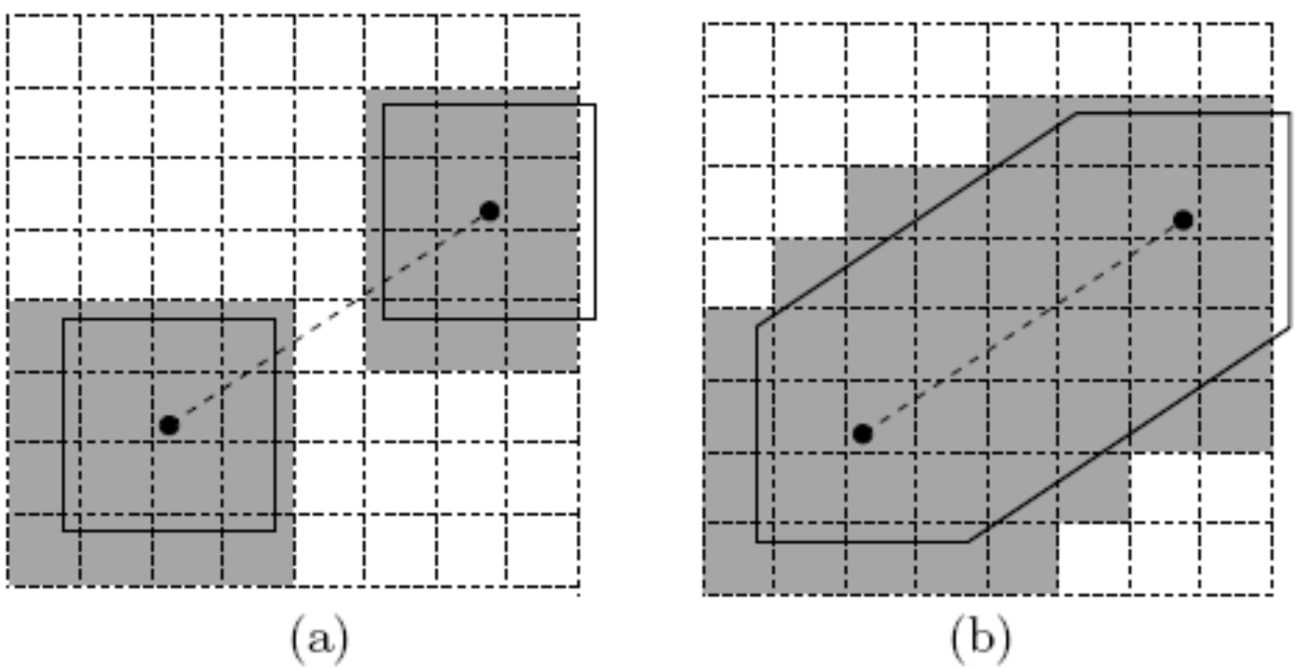
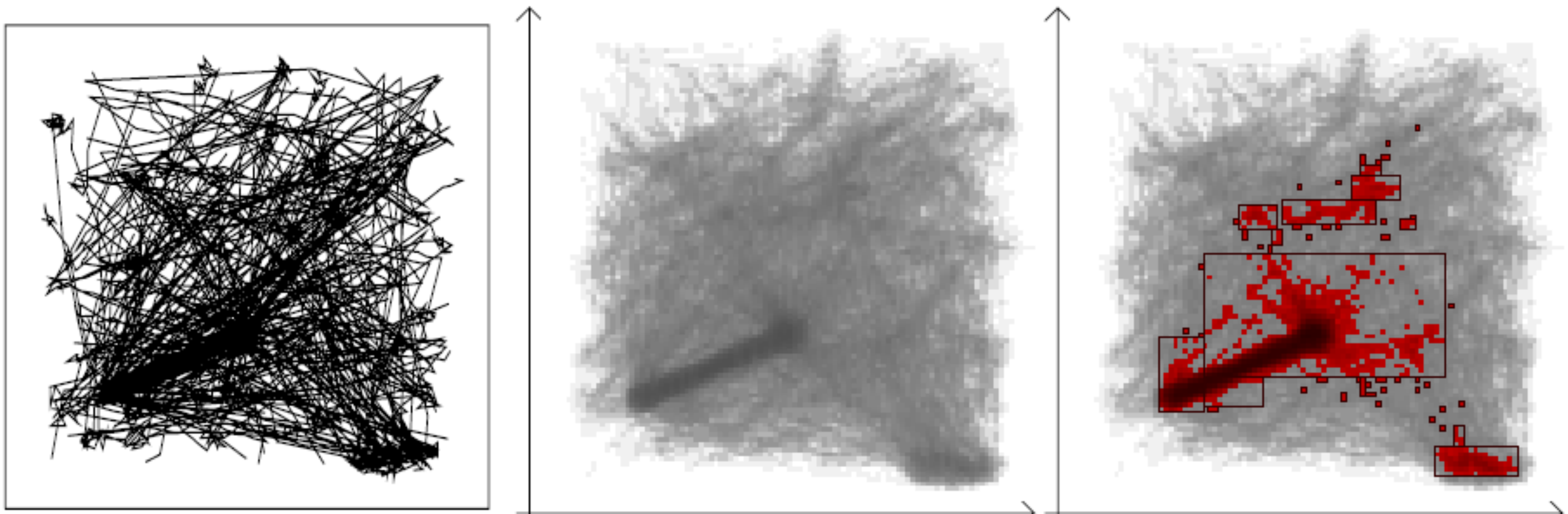


Figure 2: Density with and without regression

Construct RoI (2)

- Iteratively consider each dense cell
  - Expands in all four directions
  - Select expansion that maximizes density
  - Repeat until expansion would decrease below density threshold



(a) input trajectories (b) density distribution (c) dense cells and extracted RoI

Figure 4: Example of RoI extraction

Mining trajectory patterns with R(N)

- In principle, we need to search all-sequences of each trajectory to see if it contains a T-pattern in order to decide if it is a frequent pattern
- A T- pattern matches an input ST-sequence T when it falls in the neighborhood of any of its subsequences, which is equivalent to say that it falls in the union of the neighborhoods of all possible subsequences of T, that for convenience we will call the neighborhood of T
- frequent T-patterns are those that fall in the neighborhood of several input ST- sequences
- a density-estimation problem where we look for dense points in a space that represents T-patterns by means of tuples of points plus corresponding (n – 1)-ples of transition times

Mining trajectory patterns with R(N)

- In principle, we need to search all-sequences of each trajectory to see if it contains a T-pattern in order to decide if it is a frequent pattern
- A T- pattern matches an input ST-sequence T when it falls in the neighborhood of any of its subsequences, which is equivalent to say that it falls in the union of the neighborhoods of all possible subsequences of T, that for convenience we will call the neighborhood of T
- frequent T-patterns are those that fall in the neighborhood of several input ST- sequences
- a density-estimation problem where we look for dense points in a space that represents T-patterns by means of tuples of points plus corresponding (n – 1)-ples of transition times

However, the dimension grows rapidly

step-wise heuristic approach

- step by step: transition times can be searched in a separate step, after finding the interesting spatial points

THEOREM 3 (ANTI-MONOTONICITY). *Let T be an input trajectory, and let τ and the spatial neighborhood function $N()$ be the parameters for the T-pattern mining problem. Then:*

$$(x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_{n+1}} (x_{n+1}, y_{n+1}) \preceq_{N,\tau} T \tag{2}$$

$$\Rightarrow \quad (x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_n} (x_n, y_n) \preceq_{N,\tau} T \tag{3}$$

$$\Rightarrow \quad (x_0, y_0) \longrightarrow \dots \longrightarrow (x_n, y_n) \preceq_{N} T \tag{4}$$

step-wise heuristic approach

- step by step: transition times can be searched in a separate step, after finding the interesting spatial points

THEOREM 3 (ANTI-MONOTONICITY). *Let T be an input trajectory, and let τ and the spatial neighborhood function $N()$ be the parameters for the T-pattern mining problem. Then:*

$$(x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_{n+1}} (x_{n+1}, y_{n+1}) \preceq_{N,\tau} T \tag{2}$$

$$\Rightarrow \quad (x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_n} (x_n, y_n) \preceq_{N,\tau} T \tag{3}$$

$$\Rightarrow \quad (x_0, y_0) \longrightarrow \dots \longrightarrow (x_n, y_n) \preceq_N T \tag{4}$$

Another problem: infinite number of possible points

step-wise heuristic approach

- step by step: transition times can be searched in a separate step, after finding the interesting spatial points

THEOREM 3 (ANTI-MONOTONICITY). *Let T be an input trajectory, and let τ and the spatial neighborhood function $N()$ be the parameters for the T-pattern mining problem. Then:*

$$(x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_{n+1}} (x_{n+1}, y_{n+1}) \preceq_{N,\tau} T \quad (2)$$

$$\Rightarrow \quad (x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_n} (x_n, y_n) \preceq_{N,\tau} T \quad (3)$$

$$\Rightarrow \quad (x_0, y_0) \longrightarrow \dots \longrightarrow (x_n, y_n) \preceq_{N} T \quad (4)$$

Another problem: infinite number of possible points

- The same like to find SoI: points are not treated separately, but at each step are clustered together by following the approach used in finding SoI.

Summary

- Trajectory VS Sequence
  - the former have time information
- Mining Trajectory Pattern VS Mining Sequence Pattern
  - add the time information(constrain)
- The idea of PrefixSpan
- The idea of generating RoI

Q&A