Trajectory Pattern Mining

Fosca Giannotti, **Mirco Nanni**,
Dino Pedreschi, Fabio Pinelli

Knowledge Discovery and Delivery Lab
(ISTI-CNR & Univ. Pisa)
www-kdd.isti.cnr.it

Plan of the talk

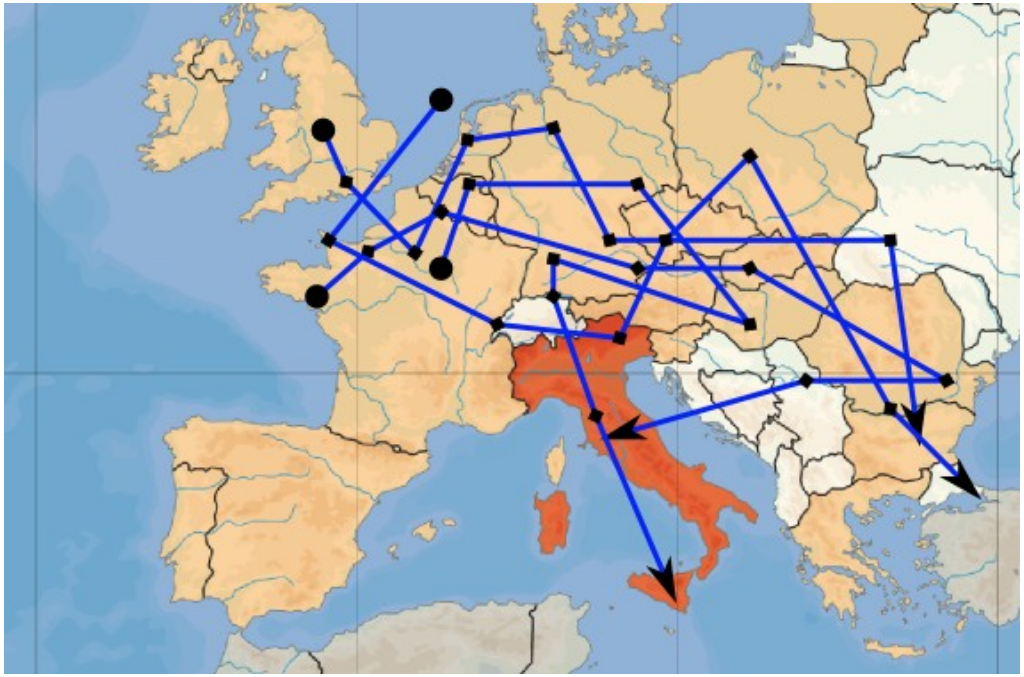
- Motivations
- T-Patterns: definition
- T-Patterns: the approach(es)
  - Regions-of-Interest approach
  - RoI extraction
  - Step-wise refinement of RoI
- Experiments
- Conclusions

Motivations

- Large diffusion of mobile devices, mobile services and location-based services

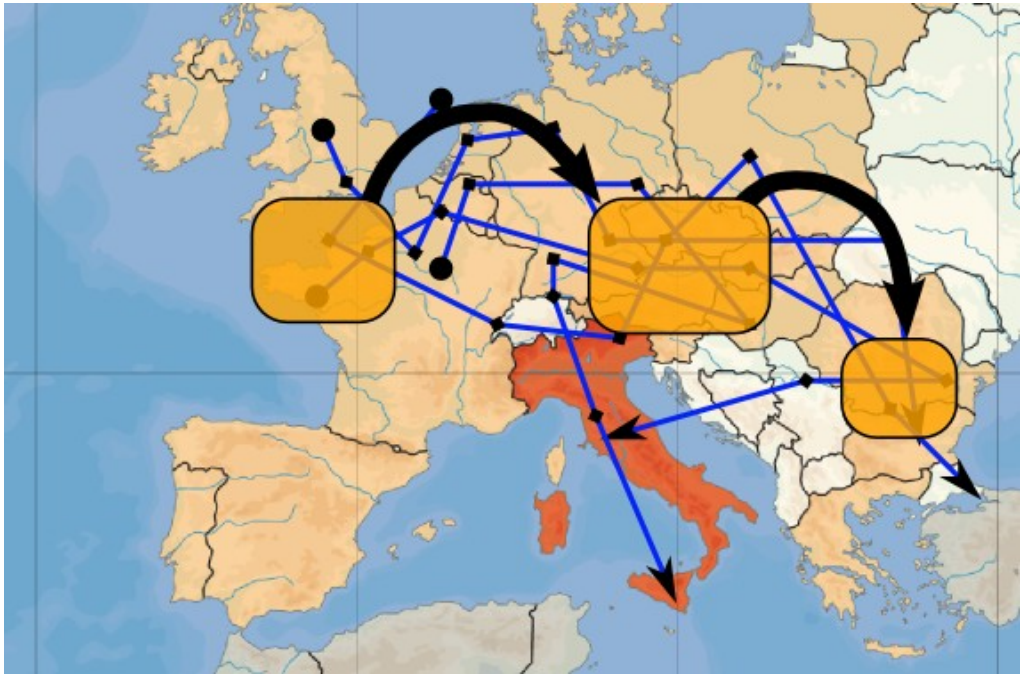
Motivations (2)

- Such devices leave digital traces that can be collected to for trajectories describing the mobility behavior of its owner



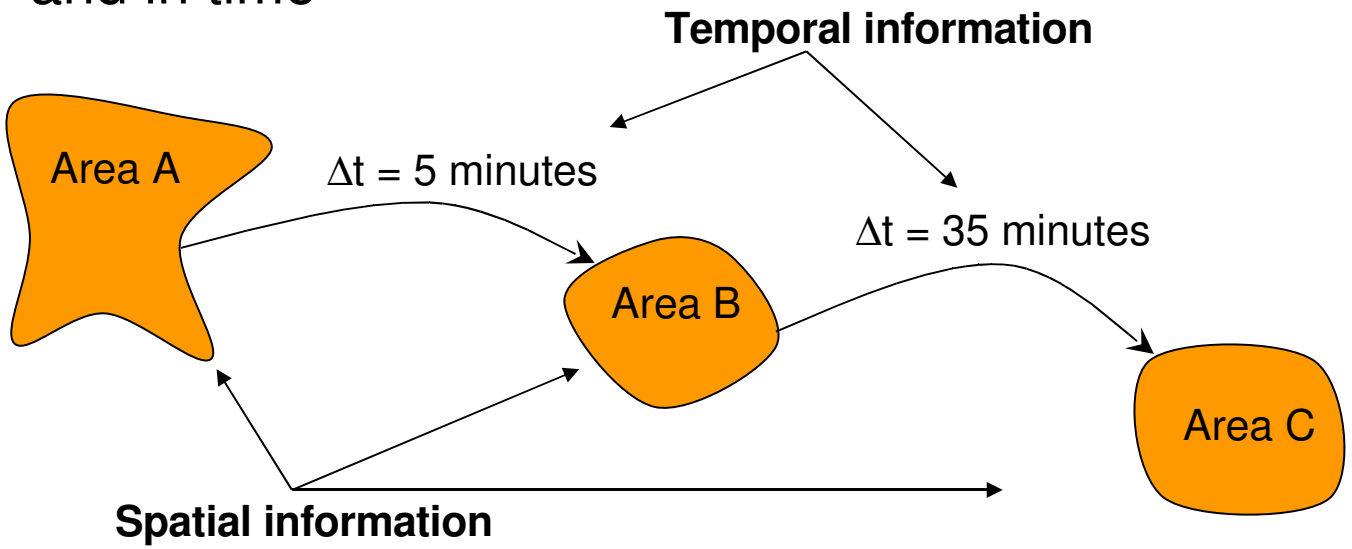
Motivations (3)

- From this large amount of data, high level information should be extracted, e.g., patterns describing mobility behaviors



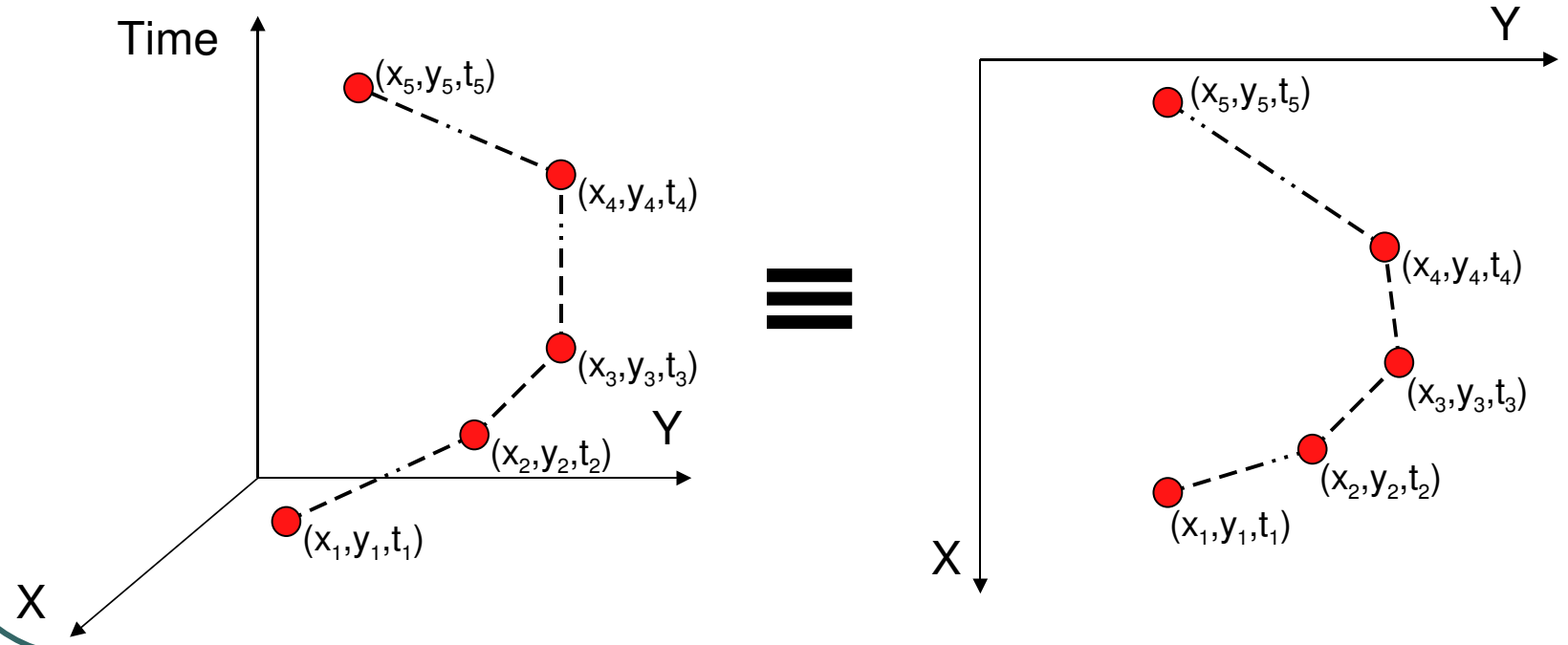
Sequential patterns for trajectories

- Question: what should a sequential pattern about moving objects look like?
  - Answer: it should describe their movements in space and in time



Sequential patterns for trajectories

- Trajectories are usually given as *spatio-temporal (ST) sequences:* $<(x_1,y_1,t_1), \ldots, (x_n,y_n,t_n)>$



T-Patterns for trajectories

- A **Trajectory Pattern** (T-pattern) is a couple (**s**, **α**):
  - **s** = $<(x_0,y_0),\ldots, (x_k,y_k)>$ is a sequence of k+1 locations
  - **α** = $<\alpha_1,\ldots, \alpha_k>$ are the transition times (*annotations*)

  also written as: $(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} (x_k, y_k)$

- A T-pattern T_p **occurs** in a trajectory if it contains a sub-sequence 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

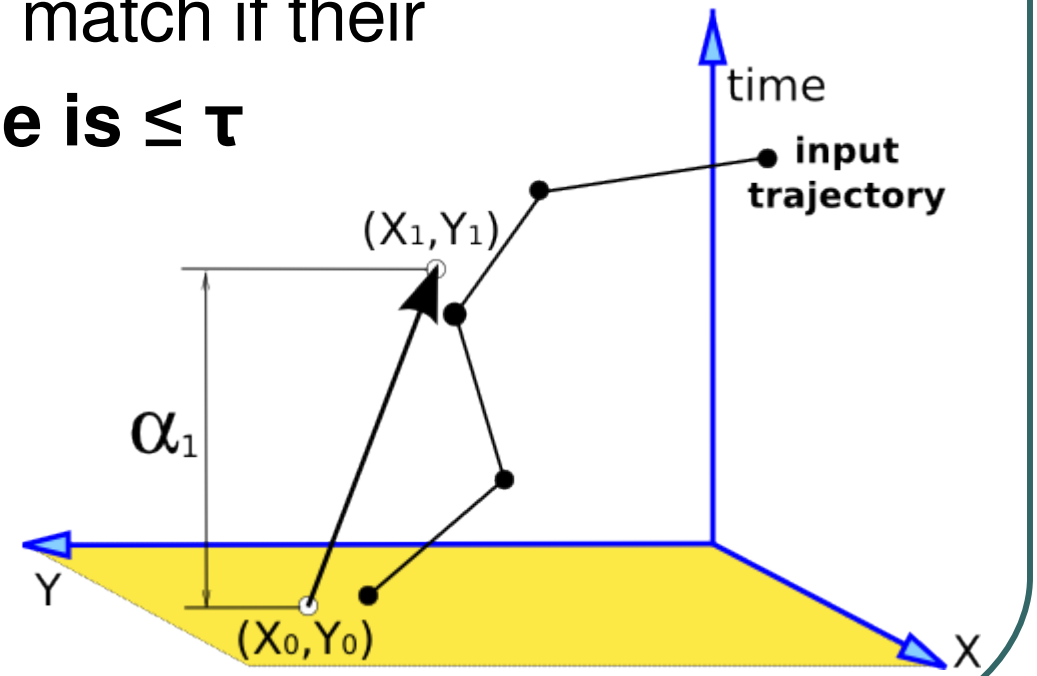
Continuity issues (space & time)

- The same exact spatial location (x,y) usually never occurs twice
  - yet, close locations essentially represent the same place, so they should match
- The same exact transition times usually do not occur often
  - same as above
- Solution: allow approximation
  - a notion of *spatial neighborhood*
  - a notion of *temporal tolerance*

T-Pattern: *approximate* occurrence

- Two points match if one falls within a **spatial neighborhood N()** of the other
- Two transition times match if their **temporal difference is ≤ τ**
- Example:

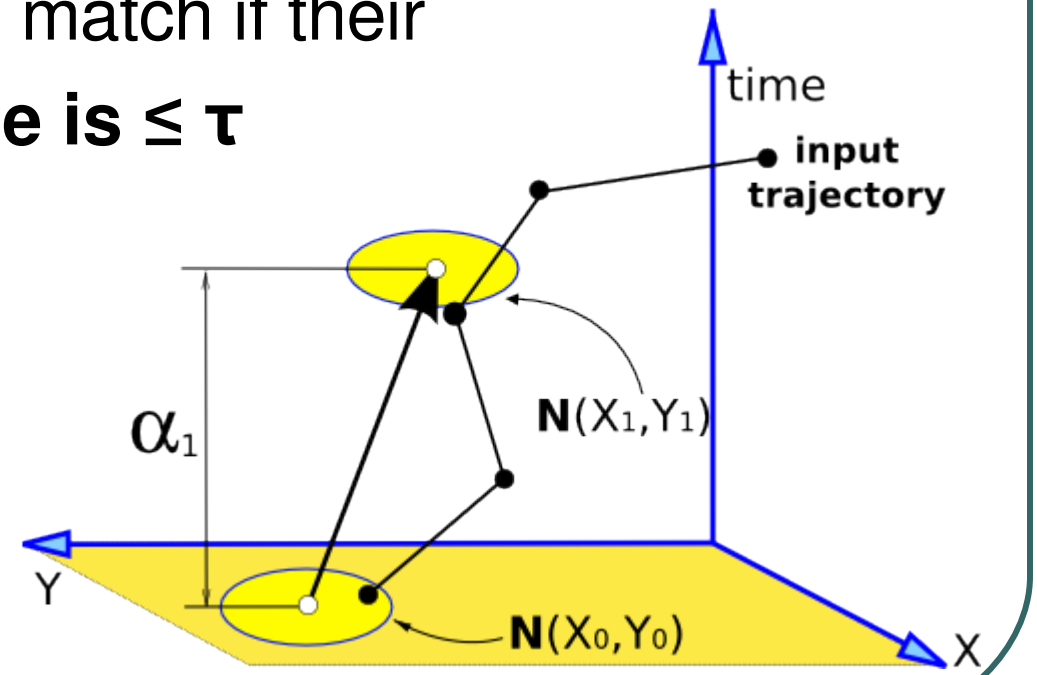
$$(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$$



T-Pattern: *approximate* occurrence

- Two points match if one falls within a **spatial neighborhood N()** of the other
- Two transition times match if their **temporal difference is ≤ τ**
- Example:

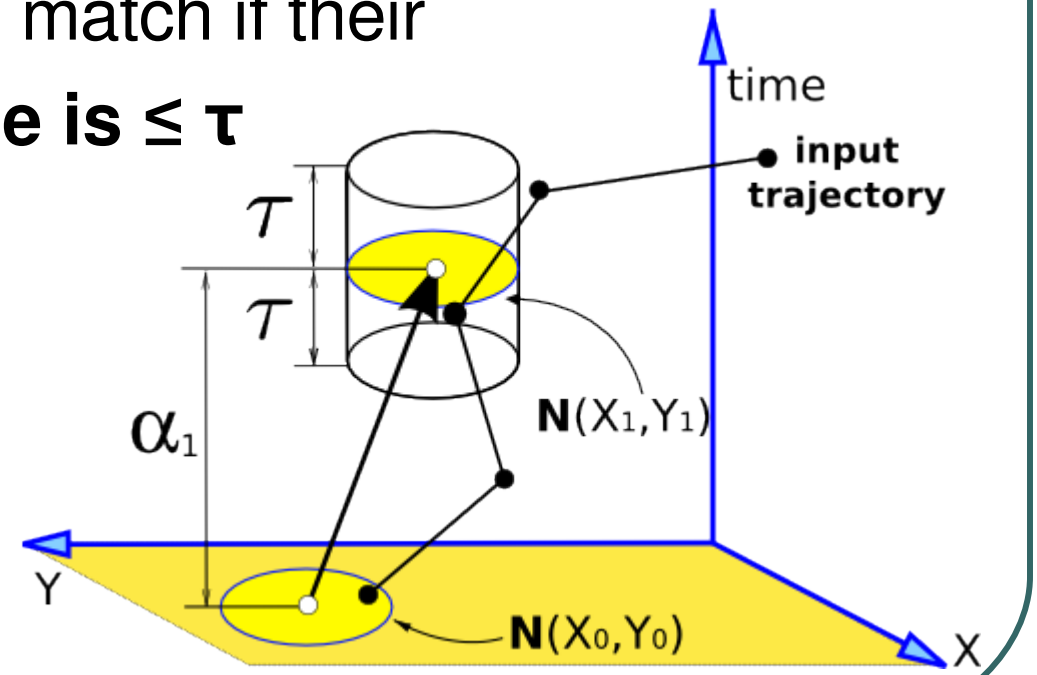
$$(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$$



T-Pattern: *approximate* occurrence

- Two points match if one falls within a **spatial neighborhood N()** of the other
- Two transition times match if their **temporal difference is ≤ τ**
- Example:

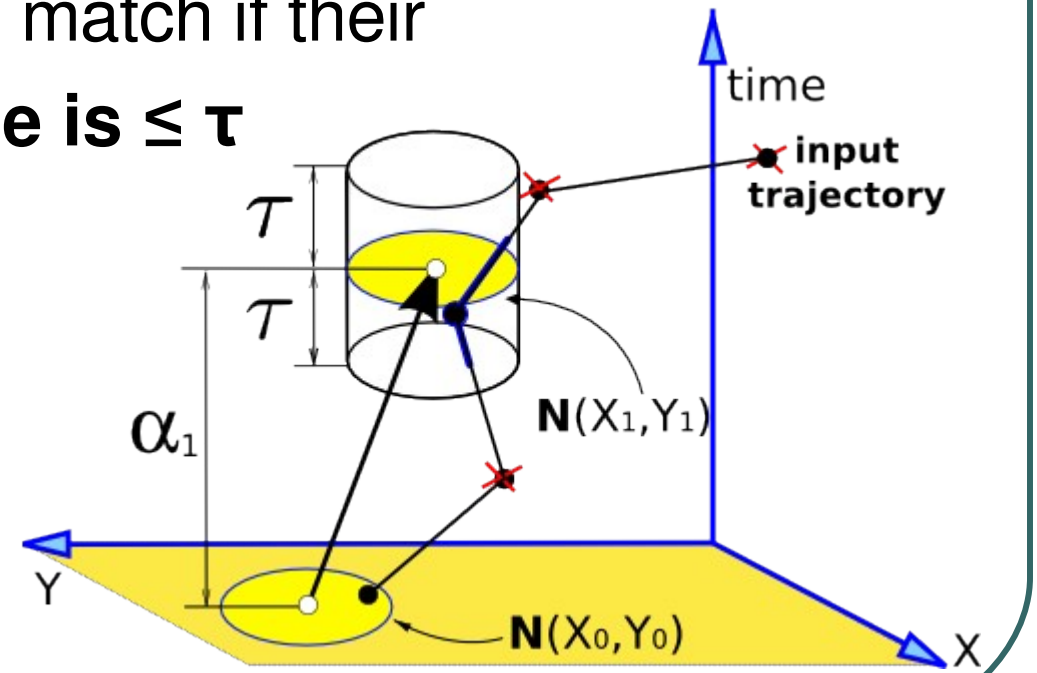
$$(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$$



T-Pattern: *approximate* occurrence

- Two points match if one falls within a **spatial neighborhood N()** of the other
- Two transition times match if their **temporal difference is ≤ τ**
- Example:

$$(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$$



Computing general T-Patterns

- T-pattern mining can be mapped to a density estimation problem over R^{3n-1}
  - 2 dimensions for each (x,y) in the pattern (2n)
  - 1 dimension for each transition (n-1)
- Density computed by
  - mapping each sub-sequence of n points of each input trajectory to R^{3n-1}
  - drawing an influence area for each point (composition of **N()**s and **τ**s), that sums up with all others
- Too expensive !!!

Simple forms of T-Pattern

- Spatial neighborhood is a parameter of the definition
- Some neighborhood functions yield tractable versions of the T-Pattern mining problem
  - “Static neighborhoods”: Regions-of-Interest

Static Neighborhoods
Regions-of-Interest (RoI)

- Given a set of *Regions of Interest* *R*, define the neighborhood of (x,y) as:

$$N_R(x,y) = \begin{cases} A & \text{if } A \in R \ \& \ (x,y) \in A \\ \varnothing & \text{otherwise} \end{cases}$$

  - Neighbors ⇔ belong to the same region
  - Points in no region have no neighbors

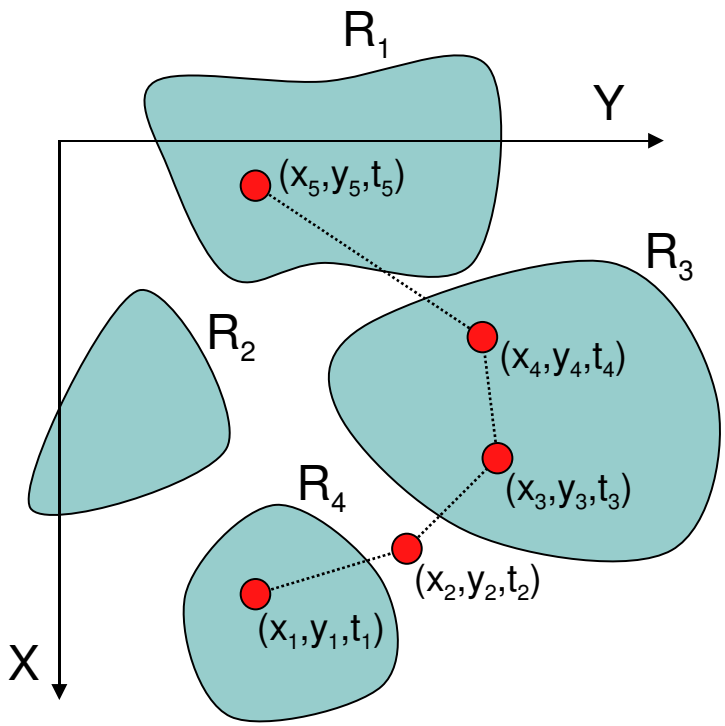
From ST-sequences to sequences

- With static neighborhoods $N_R()$ ST-sequences replaced by corresponding seqs of regions:

  *A T-pattern (**s**,**α**) is contained in a ST-sequence S=<(x_1,y_1,t_1), ..., (x_n,y_n,t_n)> ⇔ the TAS (**s'**,**α**) is contained in sequence S'*

  - ***s'*** (resp. *S'*) is obtained by mapping each element (x,y) of ***s*** (resp. *S*) to N_R(x,y)
  - TAS = Temporally annotated seq. of labels
    - E.g.: $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} s_n$
    - Mining TAS = previous work –> efficient algs

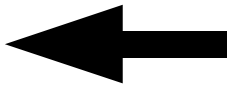
Translating ST-sequences
Example



$S=<(x_1,y_1,t_1), \ldots, (x_5,y_5,t_5)>$

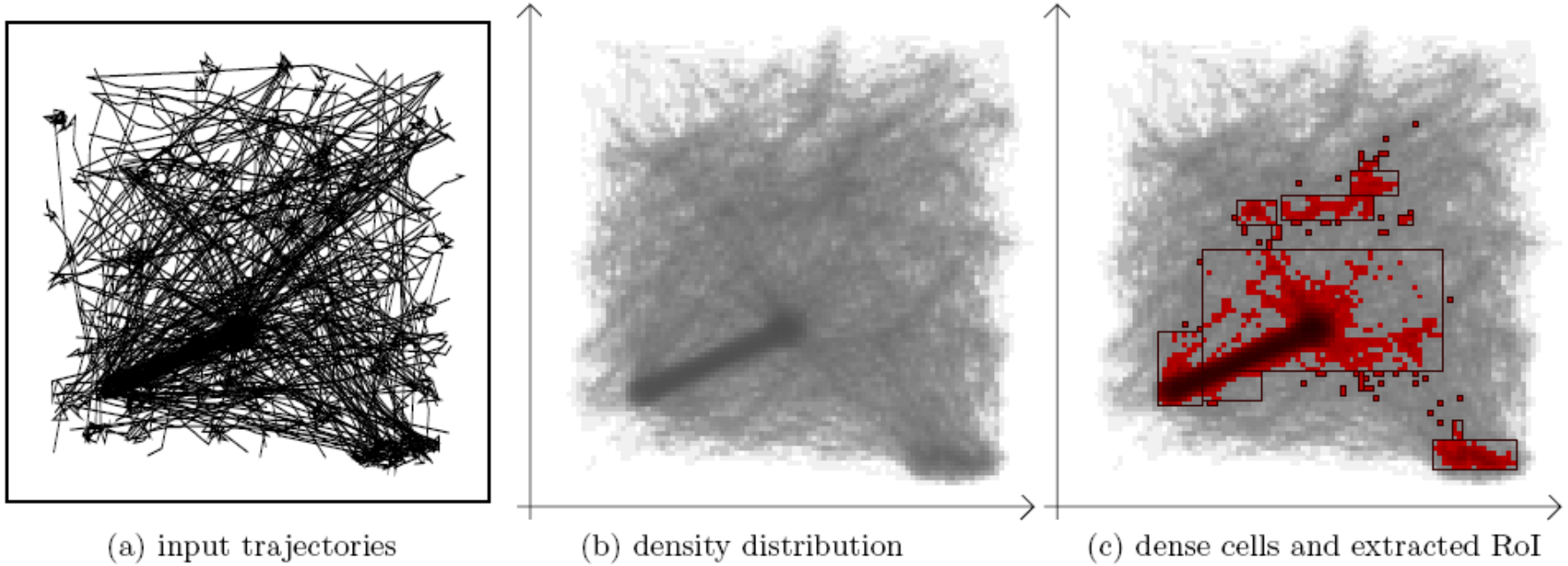
$<(R_4,t_1), (R_3,t_3), (R_3,t_4), (R_1,t_5)>$

Static Neighborhoods: issue

- What if RoI are not known a priori?
- Solution: define heuristics for automatic RoI extraction from data
- Wide range of heuristics:
  - Geography-based (e.g., crossroads)
  - Usage-based (e.g., popular places) 
  - Mixed (e.g., popular squares)

Static Neighborhoods

A usage-based heuristic



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

1. Impose a regular grid over space
2. Find dense cells (i.e., touched by many trajs.)
3. Coalesce cells into rectangles of bounded size

Static Neighborhoods

A usage-based heuristic

Algorithm: PopularRegions($\mathcal{G}$,δ)

Input: A grid $\mathcal{G}$ with densities $\mathcal{G}(i,j)$, a density threshold δ
Output: A set R of rectangular regions over $\mathcal{G}$.

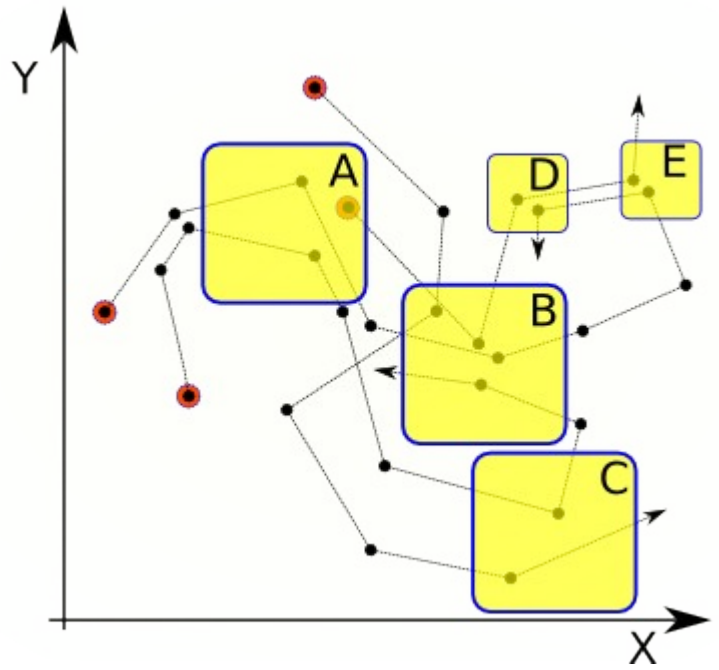
1. $R = \emptyset$; $\mathcal{G}^* = \{(i,j) \in \mathcal{G} | \mathcal{G}(i,j) \geq \delta\}$;
2. **foreach** $(i,j) \in \mathcal{G}$ **do** $used(i,j) = false$;
3. **foreach** $(i,j) \in \mathcal{G}^*$ in descending order of $\mathcal{G}(i,j)$ **do**
4. **if** $\neg used(i,j)$ **then**
5. $r = \{(i,j)\}$;
6. **repeat**
7. **foreach** $dir \in \{left, right, up, down\}$ **do**
8. $r_{dir} = r$ extended on direction dir;
9. $ext = \{\ dir |\ r_{dir} \subseteq \mathcal{G} \wedge avg\_density(r_{dir}) \geq \delta$
$\wedge\ \exists (h,k) \in (r_{dir} \setminus r).\mathcal{G}(h,k) \geq \delta$
$\wedge\ \forall (h,k) \in r_{dir}.\neg used(i,j)\ \}$;
10. **if** $ext \neq \emptyset$ **then**
11. $dir = \arg\max_{d \in ext} avg\_density(r_d)$;
12. $r = r_{dir}$;
13. **until** $ext = \emptyset$;
14. **foreach** $(i,j) \in r$ **do** $used(i,j) = true$;
15. $R = R \cup \{r\}$;
16. **return** R;

- start from densest cell
- consider any direction that (i) adds a dense cell, (ii) keeps avg density high, (iii) avoids overlap of regions
- select locally best direction

Multi-step refinement RoI

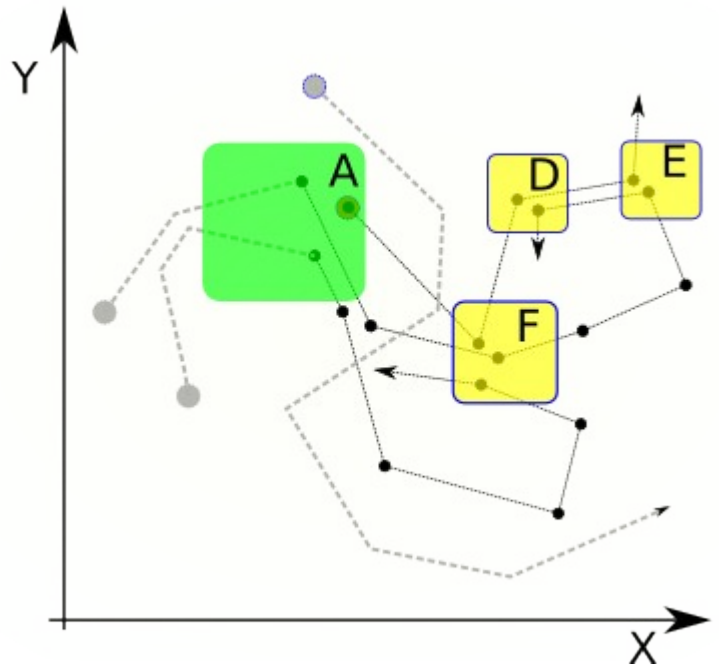
- Static RoI
  - Cells approximate single points, regions group points that are likely to form similar patterns
  - Yet, they should regard only trajectories that support the discovered pattern, not all database
- Towards general T-patterns
  - Check & update dense cells and regions of each pattern against the trajectories that support it
  - Approximation: Perform the update as step-wise refinement as patterns grow

Step-wise dynamic RoI
Example



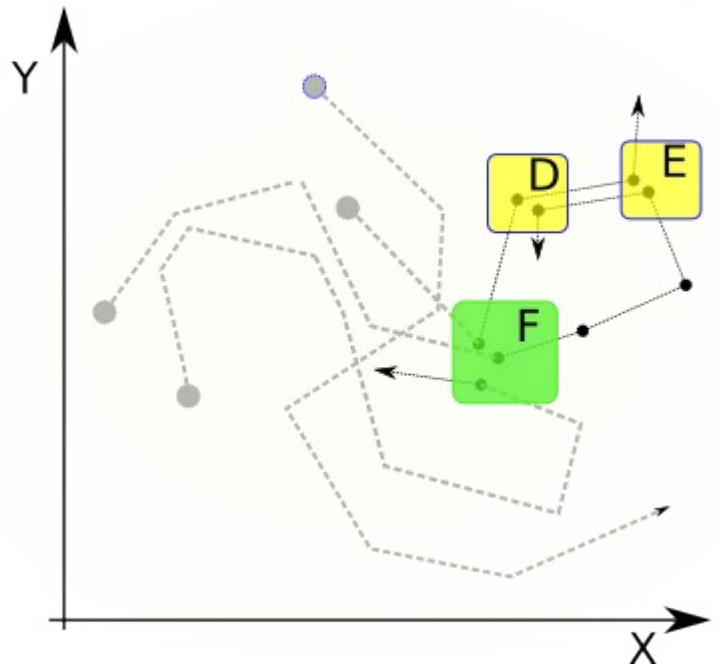
- Start computing regions as basic RoI approach
- Regions describe interesting places of *everybody*

Step-wise dynamic RoI
Example



- Focusing on A, we consider only the subset of relevant trajectories
- Regions can change (usually shrink/split)
- They are interesting only for who passes thru A

Step-wise dynamic RoI
Example



- Focusing on A->F (with some transition time), we further restrict the set of trajectories involved
- The process is repeated as far as possible

Step-wise dynamic RoI

Algorithm: Dynamic_RoI_T-pattern(T_{in}, $\mathcal{G}_0$, δ, ϵ, τ)

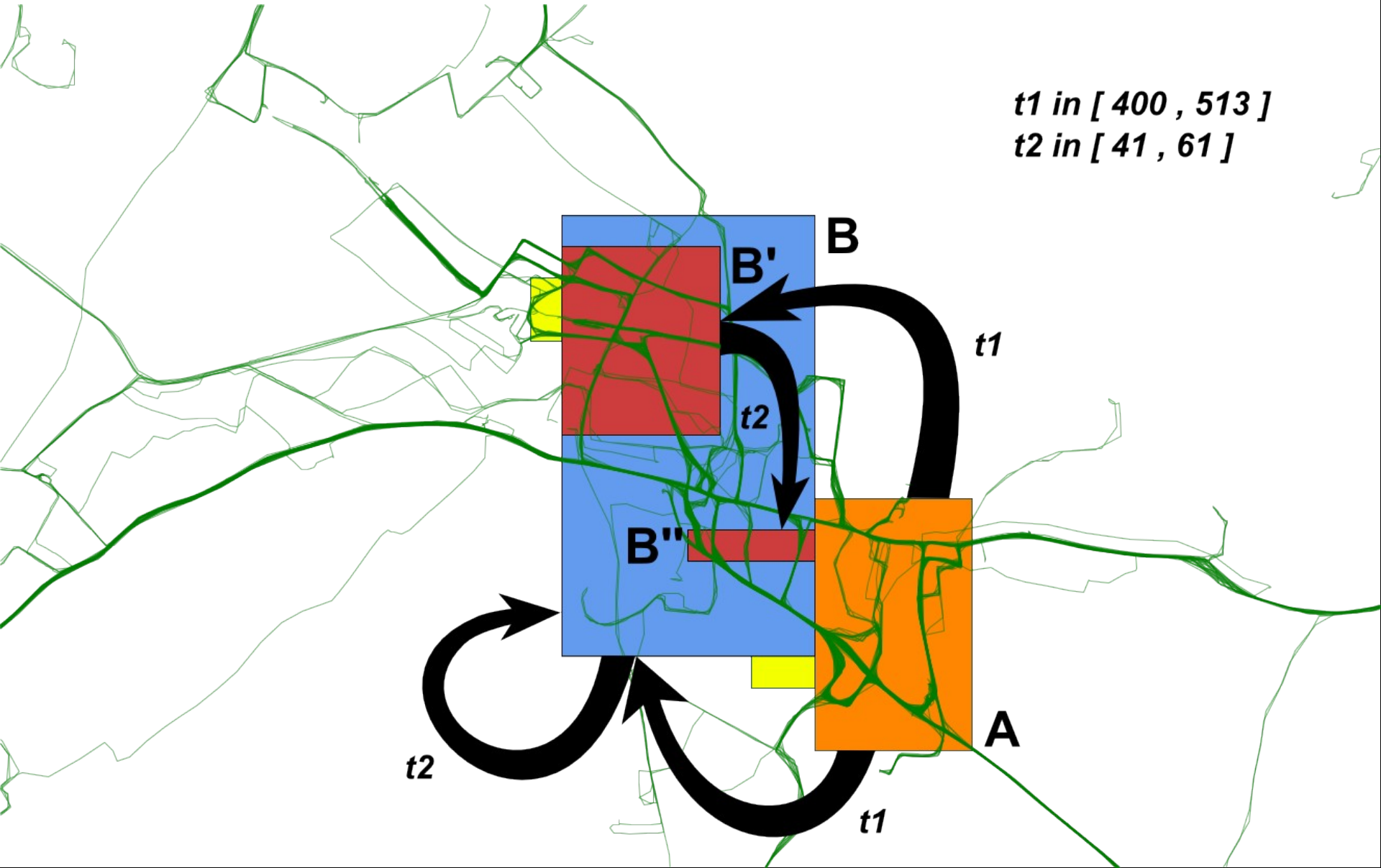
Input: A set of input trajectories T_{in}, a grid $\mathcal{G}_0$, a minimum support/density threshold δ, a radius for spatial neighborhoods ϵ, a temporal threshold τ.
Output: A set of couples $(S, \mathcal{A})$ of sequences of regions with temporal annotations.

1. $L = 0$; $T_0 = \{(T_{in} \times \{\emptyset\}, \langle\rangle)\}$;
2. **while** $T_L \neq \emptyset$ **do**
3. $T_{L+1} = \emptyset$;
4. **foreach** $(T, prefix) \in T_L$ **do**
5. **if** $|prefix| \geq 2$ **then**
6. $\mathcal{A} = \text{ExtractFrequentTimings}(T)$; *([5])*
7. Output $(prefix, \mathcal{A})$;
8. $T = \text{PruneEmptyAnnotations}(T, \mathcal{A})$; *([5])*
9. $\mathcal{G} = \text{ComputeDensity}(T, \mathcal{G}_0, \epsilon)$; *(Sect.4.2.1)*
10. $RoI = \text{PopularRegions}(\mathcal{G}, \delta)$; *(Sect.4.2.2)*
11. $\mathcal{D} = \text{Translate}(T, RoI)$; *(Sect.4.1)*
12. **foreach** $r \in RoI$ **do**
13. **if** $support_{\mathcal{D}}(r) \geq \delta$ **then**
14. $\mathcal{D}' = \text{ExtendProjection}(\mathcal{D}, r)$; *([5])*
15. $T' = \{ (traj, \mathcal{A}') \mid (traj, \mathcal{A}) \in T \wedge (S', \mathcal{A}') \in \mathcal{D}' \wedge traj.ID = S'.ID \wedge traj' = \text{Cut}(traj, \mathcal{A}') \}$
16. $T_{L+1} = T_{L+1} \cup \{(T', append(prefix, r))\}$;
17. L++;

- Extract freq. transition times
- Compute up-to-date RoI
- Extend patters w.r.t. new RoI
- Focus on patterns found

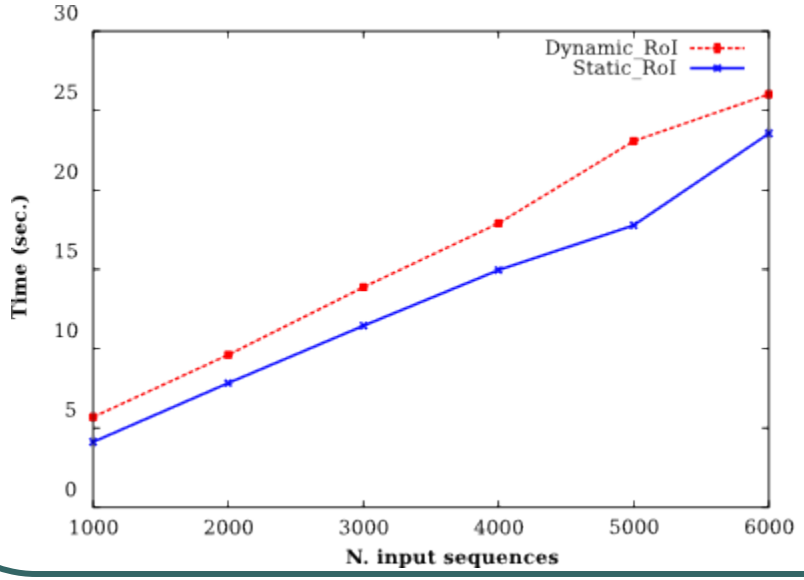
Sample T-patterns

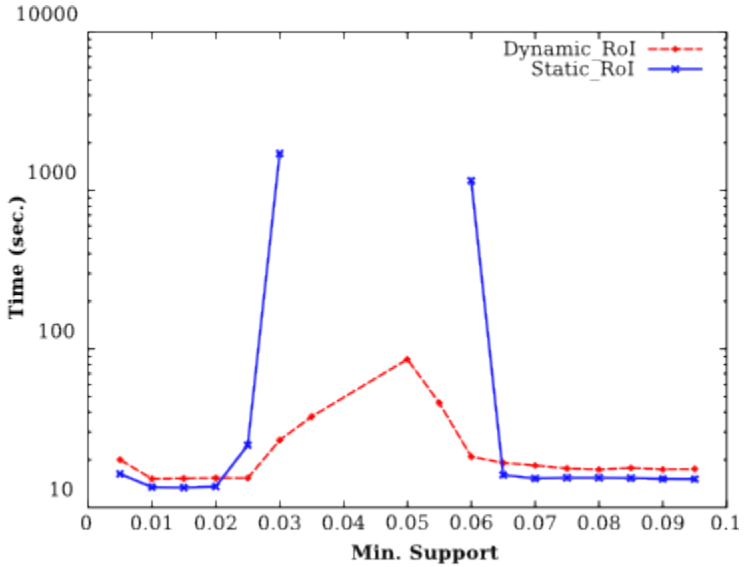
(Data source: trucks in Athens – 273 trajectories)



Performances

- Linear scalability w.r.t. number of trajs
- Quickly growing cost around (left& right) critical support thresholds
  - Dynamic approach prunes better





Ongoing work

- Application-oriented tests on large, real datasets
- Study relations with
  - Geographic background knowledge
  - Privacy issues
  - Reasoning on trajectories and patterns
- Simplification of output transition times
  - The most complex info for end users

End of the talk

- Thanks for your attention
- Questions and remarks are welcome

Have a look at our poster:

- ***this evening*** *(Monday, 13th August)*
- ***board 27***

Contact me at: mirco.nanni @ isti.cnr.it

- *software available*
- *download page and user manuals under construction*