

# Trajectory Pattern Mining

Fosca Giannotti<sup>1</sup>Mirco Nanni<sup>1</sup>Dino Pedreschi<sup>2</sup>Fabio Pinelli<sup>1</sup>

Pisa KDD Laboratory

<sup>1</sup>ISTI - CNR, Area della Ricerca di Pisa, Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy<sup>2</sup>Computer Science Dep., University of Pisa, Largo Pontecorvo, 3 - 56127 Pisa, Italy

## ABSTRACT

The increasing pervasiveness of location-acquisition technologies (GPS, GSM networks, etc.) is leading to the collection of large spatio-temporal datasets and to the opportunity of discovering usable knowledge about movement behaviour, which fosters novel applications and services. In this paper, we move towards this direction and develop an extension of the sequential pattern mining paradigm that analyzes the trajectories of moving objects. We introduce *trajectory patterns* as concise descriptions of frequent behaviours, in terms of both space (i.e., the regions of space visited during movements) and time (i.e., the duration of movements). In this setting, we provide a general formal statement of the novel mining problem and then study several different instantiations of different complexity. The various approaches are then empirically evaluated over real data and synthetic benchmarks, comparing their strengths and weaknesses.

## Categories and Subject Descriptors

H.2.8 [Database Applications]: Data mining

## General Terms

Algorithms

## Keywords

Trajectory patterns, Spatio-temporal data mining

## 1. INTRODUCTION

Spatio-temporal patterns that succinctly show the cumulative behaviour of a population of moving objects are useful abstractions to understand mobility-related phenomena. In particular, a form of pattern, which represents an aggregated abstraction of many individual trajectories of moving objects within an observed population, would be extremely useful in the domain of sustainable mobility and traffic management in metropolitan areas, where the discovery of traffic

flows among sequences of different places in a town (origin-destination flows) is a key issue [2].

Nowadays, the movement of people or vehicles within a given area can be observed from the digital traces left behind by the personal or vehicular mobile devices, and collected by the wireless network infrastructures. For instance, mobile phones leave positioning logs, which specify their localization, or cell, at each moment they are connected to the GSM network; analogously, GPS-equipped portable devices can record their latitude-longitude position at each moment they are exposed to a GPS satellite, and transmit their trajectories to a collecting server. The pervasiveness of ubiquitous technologies guarantees that there will be an increasing availability of large amounts of data pertaining to individual trajectories, at increasing localization precision; therefore, there is an opportunity – and a challenge – to discover automatically, from these trajectories, spatio-temporal patterns that convey useful knowledge.

In this paper, we precisely address this problem, and introduce a novel form of spatio-temporal pattern, which formalizes the mentioned idea of aggregate movement behaviour. The new pattern, that we call a *trajectory pattern*, represents a set of individual trajectories that share the property of visiting the same sequence of places with similar travel times. Therefore, two notions are central: (i) the *regions of interest* in the given space, and (ii) the *typical travel time* of moving objects from region to region. In fact, in our approach a trajectory pattern is a sequence of spatial regions that, on the basis of the source trajectory data, emerge as frequently visited in the order specified by the sequence; in addition, the transition between two consecutive regions in such a sequence is annotated with a typical travel time that, again, emerges from the input trajectories. For instance, consider the following two trajectory patterns over regions of interest in the centre of a town:

$$\text{Railway Station} \xrightarrow{15min} \text{Castle Square} \xrightarrow{2h15min} \text{Museum} \quad (a)$$

$$\text{Railway Station} \xrightarrow{10min} \text{Middle Bridge} \xrightarrow{10min} \text{Campus} \quad (b)$$

Here, pattern (a) may be interpreted as a typical behaviour of tourists that rapidly reach a major attraction from the railway station and spend there about two hours before getting to the adjacent museum. Pattern (b), instead, may highlight the pedestrian flow of students that reach the university campus from the station: for them, the central bridge over the river is a compulsory passage. It should be observed that a trajectory pattern does not specify any particular route among two consecutive regions: instead, a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'07, August 12-15, 2007, San Jose, California, USA.

Copyright 2007 ACM 978-1-59593-609-7/07/0008 ...\$5.00.

typical travel time is specified, which approximates the (similar) travel time of each individual trajectory represented by the pattern. Also, it should be observed that the individual trajectories aggregated in a pattern are not necessarily simultaneous: we only require that such trajectories visit the same sequence of places with similar transition times, even if they start at different absolute times<sup>1</sup>. Trajectory patterns are a spatio-temporal variant of the *temporally-annotated sequences*, or *TAS* in short, introduced in [5], where only the time dimension is taken into account and the elements of a frequent sequence are generic events without any specific spatial semantics.

Obviously, once defined a potentially useful pattern, it is necessary to provide an effective mining algorithm to extract trajectory patterns from the source trajectory data. We actually provide three algorithms for trajectory pattern mining, with reference to three approaches of increasing complexity for dealing with the spatial dimension. The first two solutions share the idea that the problem of identifying the regions of interest is a discretization problem for the spatial dimension, to be tackled *before* the actual mining of trajectory patterns: in this approach, the input trajectories are transformed from sequences of points into sequences of regions of interest in a preprocessing step; then, in the pattern mining step, the temporal dimension only is taken into account, in order to mine sequences with temporal annotations, by adopting the method introduced in [5]. The first two methods differ in the form of spatial discretization adopted:

- Pre-conceived regions of interest: in this case, some subjective background knowledge is used to specify a set of places of interest, which are known as movement attractors. This approach is used, for instance, in selecting the zones used in origin-destination matrices, a classical tool of transportation engineering for describing traffic flows (see, e.g., [2]).
- Popular regions: in this case, regions of interest are identified as the mostly visited places in the input trajectories. Accordingly, we introduce a density-based spatial discretization method to discover popular regions from the input trajectories.

The third trajectory pattern mining algorithm is genuinely spatio-temporal, in the sense that the identification of the regions of interest is dynamically intertwined with the mining of sequences with temporal information. This approach is capable to detect more precise trajectory patterns, as the regions of interest are incrementally identified as locally dense regions, i.e., with respect to the trajectories in the patterns found so far.

The methods have been put in practice in an empirical experiment over synthetic and real datasets of GPS trajectory data, as a preliminary demonstration of the computational feasibility of our approach, as well as of the usefulness of the introduced pattern. Summarizing, the contributions presented in this paper are: (i) the definition of the novel trajectory pattern (Section 3); (ii) a density-based algorithm for discovering regions of interest (Section 4); (iii) a trajectory pattern mining algorithm with predefined regions of

<sup>1</sup>It is however possible to focus on specific time intervals, e.g., rush hours, by temporal selection over the input trajectories.

interest (Section 5); (iv) a trajectory pattern mining algorithm which dynamically discovers regions of interest (Section 6). The paper is completed by a discussion of related work (Section 2) and a report of preliminary empirical experiments (Section 7).

## 2. BACKGROUND AND RELATED WORK

In this section we summarize some relevant research related to the topic of this paper, and introduce some basic concepts and terminology.

### 2.1 Spatio-temporal sequential patterns

The basic *frequent sequential pattern* (FSP) problem, originally introduced in [1], is defined over a database of sequences  $D$ , where each element of each sequence is a time-stamped set of items — i.e., an *itemset*. Time-stamps determine the order of elements in the sequence. Then, the FSP problem consists in finding all the sequences that are *frequent* in  $D$ , i.e., appear as subsequence of a large percentage of sequences of  $D$ . A sequence  $\alpha = \alpha_1 \rightarrow \dots \rightarrow \alpha_k$  is a subsequence of  $\beta = \beta_1 \rightarrow \dots \rightarrow \beta_m$  ( $\alpha \preceq \beta$ ) if there exist integers  $1 \leq i_1 < \dots < i_k \leq m$  such that  $\forall_{1 \leq n \leq k} \alpha_n \subseteq \beta_{i_n}$ . Then we can define the support  $supp_D(S)$  of a sequence  $S$  as the percentage of transactions  $T \in D$  such that  $S \preceq T$ , and say that  $S$  is frequent w.r.t. threshold  $s_{min}$  if  $supp_D(S) \geq s_{min}$ . Since its first definition, many algorithms for sequential patterns have been proposed, from the earliest in [1], to the more recent PrefixSpan [9] and SPADE [12].

Moving from the discrete realm of items and events to the continuous context of spatio-temporal sequences, such as the sequence of positions of a moving object, the standard notion of sequential pattern borrowed from transactional data mining, i.e., a pattern that *exactly* occurs several times in the data, usually cannot be applied and thus some kind of tolerance to small perturbations is needed. To the best of our knowledge, the existing literature on this subject is composed of only a few recent works, that tackle the problem from different viewpoints.

The work in [3] considers patterns that are in the form of trajectory segments and searches approximate instances in the data; on the opposite, the work in [7] provides a clustering-based perspective, and considers patterns in the form of moving regions within time intervals, such as spatio-temporal cylinders or tubes and counts as occurrences all trajectory segments partially contained in the moving regions. Finally, a similar goal, but focused on cyclic patterns, is pursued in [8]: the authors propose an effective and fast mining algorithm for retrieving *maximal* periodic patterns, treating time as discrete, yet dealing with continuous spatial locations that are discretized dynamically through density-based clustering.

We conclude this section mentioning a different line of research focused on the extraction of patterns over sequences of events that describe also the temporal relations between events, e.g., sequences with characteristic transition times between consecutive events (as first proposed in [11] and later more deeply explored by [5]), or more general sets of events with temporal constraints between them, such as chronicles [10]. We notice that these methods are not specific for moving object data, and so far no specific work has been presented in literature for the spatio-temporal context.

This paper, instead, focusses on a method for extracting patterns containing both spatial and temporal information,

building on the work in [5], which is purely temporal. The rest of this section briefly summarizes the key aspects of this work.

## 2.2 Temporally Annotated Sequences

Temporally annotated sequences ( $\mathcal{TAS}$ ), introduced in [5], are an extension of sequential patterns that enrich sequences with information about the typical transition times between their elements.  $\mathcal{TAS}$ 's have the following form:

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

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

As an example, a  $\mathcal{TAS}$  over the web pages visited along several user navigation sessions can be the following:  $\{ '/' \} \xrightarrow{2} \{ '/papers.html' \} \xrightarrow{90} \{ '/kdd.html' \}$ , that represents a sequence that starts from the root, then after 2 seconds continues with page 'papers.html' and finally, after 90 seconds ends with page 'kdd.html'.

Similarly to traditional sequential pattern mining, the notion of frequency is based on the notion of support of a  $\mathcal{TAS}$ , which in turn is defined as the number of input sequences that *contain* the  $\mathcal{TAS}$ . The key notion of containment can be defined as follows:

*Definition 1.* ( $\tau$ -containment ( $\preceq_\tau$ )) Given a time threshold  $\tau$ , a  $\mathcal{TAS}$   $T = s_0 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} s_n$  is  $\tau$ -*contained* (or *occurs*) in an input sequence  $I = \langle (I_0, t_0), \dots, (I_m, t_m) \rangle$ , denoted as  $T \preceq_\tau I$ , if and only if there exists a sequence of integers  $0 \leq i_0 < \dots < i_n \leq m$  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  $\forall_{1 \leq k \leq n}. \alpha'_k = t_{i_k} - t_{i_{k-1}}$ .

Essentially, a  $\mathcal{TAS}$   $T$  is  $\tau$ -contained into an input sequence  $I$  if there is an occurrence of  $T$  in  $I$  (condition 1) having transition times similar to the annotations in  $T$  (condition 2). An example of  $\tau$ -containment is the following:

$$\begin{array}{c} T : \{ a \} \xrightarrow{4} \{ b \} \xrightarrow{9} \{ c \} \\ I : \{ \hat{a} \}, 0 \xrightarrow{3} \{ \hat{b} \}, 3 \xrightarrow{14-3=11} \{ \hat{c} \}, 14 \end{array}$$

where the sequence in  $T$  occurs in  $I$ , and the transition times of the occurrence differ at most of 2 time units. Therefore, if  $\tau \geq 2$  we have that  $T \preceq_\tau I$ .

Now, frequent sequential patterns can be easily extended to the notion of *frequent TAS*'s, which are simply defined as  $\mathcal{TAS}$ 's that are  $\tau$ -contained in at least  $s_{min}$  input sequences,  $s_{min}$  being a minimum support threshold provided by the user. However, introducing time in sequential patterns gives rise to a novel issue: in general the number of frequent  $\mathcal{TAS}$ 's for a dataset is infinite. For instance, in the single-sequence example given above, if  $\tau = 2$  and  $s_{min} = 1$ , not only  $T$  is frequent, but also any other variant of  $T$  having transition times  $\langle t_1, t_2 \rangle$ , where  $t_1 \in [1, 5]$  and  $t_2 \in [9, 13]$ .

In [5] it was shown that discovering the frequent  $\mathcal{TAS}$ 's ( $S, A$ ) for any given sequence  $S$  can be neatly formalized as a density estimation problem (in particular, a kernel-based

estimation problem), and that the infinite set of frequent  $\mathcal{TAS}$ 's can be represented in a finite and concise way.

Finally, an efficient prefix projection-based algorithm for extracting frequent  $\mathcal{TAS}$ 's was provided, that interleaves prefix extension steps and detection of frequent annotations for each prefix, exploiting the monotonicity properties of density over the annotation space and the relations between frequent sequences and frequent  $\mathcal{TAS}$ 's.

## 3. PROBLEM DEFINITION

The basic object of our investigation is the trajectory that describes the movement of an object. To our purpose, a trajectory of an object is a sequence of time-stamped locations, representing the traces collected by some wireless/mobility infrastructure, such as the GSM mobile phone network, or GPS traces recorded by portable devices and transmitted to a central server. The location, like a GSM cell or a lat-long pair, is abstracted using ordinary Cartesian coordinates, as formally stated by the following:

*Definition 2.* (ST-sequence) A *spatio-temporal sequence* (ST-sequence) or *trajectory* is a sequence of triples  $S = \langle (x_0, y_0, t_0), \dots, (x_k, y_k, t_k) \rangle$ , where  $t_i$  ( $i = 0..k$ ) is a timestamp,  $\forall_{0 \leq i < k} t_i < t_{i+1}$  and  $(x_i, y_i)$  are points in  $\mathbf{R}^2$ .

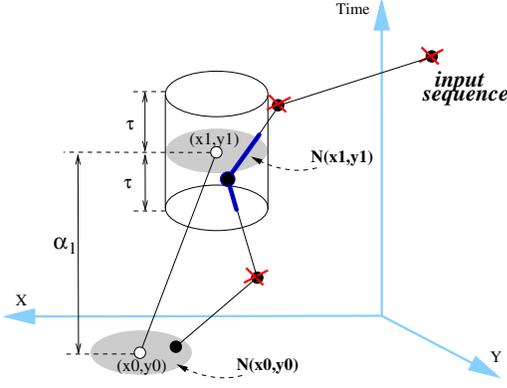
The fundamental step in moving from sequences to spatio-temporal sequences consists in replacing the discrete elements that form each sequence, usually taken from a predefined alphabet, with spatial locations. Therefore, mining spatio-temporal sequences will focus on the relations between (chronologically ordered) positions in space, whereas standard sequence mining focuses on relations between some given event types (taken from the above mentioned alphabet).

The key task in sequence mining consists in counting the occurrences of a pattern, i.e., those segments of the input data that match a candidate pattern. Matching the elements of a sequence in standard sequential patterns requires simple equality tests between symbols; instead, in our context it requires matching spatial locations, on the base of some notion of approximated match and error tolerance. That can be formally expressed in a simple and general way by means of a neighborhood function  $N : \mathbf{R}^2 \rightarrow \mathcal{P}(\mathbf{R}^2)$ , which assigns to each pair  $(x, y)$  a set  $N(x, y)$  of neighboring points.

*Definition 3.* (Spatial containment,  $\preceq_N$ ) Given a sequence of spatial points  $S = \langle (x_0, y_0), \dots, (x_k, y_k) \rangle$ , a spatio-temporal sequence  $T = \langle (x'_0, y'_0, t'_0), \dots, (x'_n, y'_n, t'_n) \rangle$  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  $0 \leq i_0 < \dots < i_k \leq n$  such that:  $\forall_{0 \leq j \leq k}. (x_j, y_j) \in N(x'_{i_j}, y'_{i_j})$ .

The inclusion of temporal information in a sequential pattern can be obtained by making the patterns include temporal constraints between consecutive elements of the sequence, following the same spirit of temporally annotated sequences ( $\mathcal{TAS}$ ) [5]:

*Definition 4.* (T-pattern) A *Trajectory pattern*, called T-pattern, is a pair  $(S, A)$ , where  $S = \langle (x_0, y_0), \dots, (x_k, y_k) \rangle$  is a sequence of points in  $\mathbf{R}^2$ , and  $A = \langle \alpha_1, \dots, \alpha_k \rangle \in \mathbf{R}_+^k$  is the (*temporal*) *annotation* of the sequence. T-patterns will also be represented as  $(S, A) = (x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_k} (x_k, y_k)$ .



**Figure 1: Matching T-pattern**  $(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$  **against an input ST-sequence.**

An occurrence of a T-pattern takes place when both spatial positions and transition times of the pattern approximately correspond to those found in an input sequence:

*Definition 5.* (Spatio-temporal containment,  $\preceq_{N,\tau}$ ) Given a spatio-temporal sequence  $T$ , time tolerance  $\tau$ , a neighborhood function  $N : \mathbf{R}^2 \rightarrow \mathcal{P}(\mathbf{R}^2)$  and a T-pattern  $(S, A) = (x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_k} (x_k, y_k)$ , 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  $T$ ,  $T' = \langle (x'_0, y'_0, t'_0), \dots, (x'_k, y'_k, t'_k) \rangle$  such that:

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

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

Intuitively, a T-pattern is contained in a trajectory if the latter contains an approximated instance of the former, the approximation being associated with both the spatial and the temporal dimensions. We notice that comparisons are not performed on absolute times, as spatio-temporal containment is based on the *transition times* between two consecutive elements in the sequence, expressed by the  $\alpha_i$  and  $\alpha'_i$  terms of condition 2 in Definition 5.

As an example, Figure 1 shows how the spatial and temporal constraints essentially form a spatio-temporal neighborhood around each point of the reference trajectory. Moreover, we notice that in our model the neighborhood  $N()$  of a point depends only on the spatial coordinates of the points, and therefore neighborhoods are time-independent. In the graphical example, this is reflected by the shape of the spatio-temporal neighborhood, which is obtained as extrusion of a spatial neighborhood along the time dimension.

From containment, a natural definition of support and frequent pattern can be assigned, as well as a general definition of the trajectory pattern mining problem.

*Definition 6.* (Trajectory pattern mining) Given a database of input trajectories  $\mathcal{D}$ , a time tolerance  $\tau$ , a neighborhood function  $N()$  and a minimum support threshold  $s_{min}$ , the *trajectory pattern mining* problem consists of finding all frequent T-patterns, i.e., all T-patterns  $(S, A)$  such that

$$\text{support}_{\mathcal{D},\tau,N}(S, A) \geq s_{min}$$

where the support  $\text{support}_{\mathcal{D},\tau,N}$  of a T-pattern  $(S, A)$  is the number of input trajectories  $T \in \mathcal{D}$  such that  $(S, A) \preceq_{N,\tau} T$ .

Notice that the neighborhood function is a parameter of the definition of containment, and different neighborhood functions yield different variants of frequent T-patterns. In particular, some choices lead to very complex mining problems, while others yield more tractable variants. Section 4 will introduce an approach of the latter kind, while Section 6.1 discusses a complex example of the former kind and is followed by a trade-off solution described in Section 6.2.

## 4. REGIONS-OF-INTEREST

The general problem defined in Section 3 flexibly allows to follow several different approaches, each corresponding to a different choice of the neighborhood function  $N(x, y)$ . Choosing a neighborhood function essentially means implementing some specific notion of spatial similarity that will be used in the spatio-temporal containment test. In this section, in particular, the neighborhood function is used to model *Regions-of-Interest* (RoI), that represent a natural way to partition the space into meaningful areas and, correspondingly, to associate spatial points with region labels. A straightforward solution is to reduce the problem of T-pattern mining to the problem of mining simple  $\mathcal{TAS}$ 's, for which an efficient solution was already provided. The details of such a reduction process are provided in Section 4.1.

### Integrating RoI and trajectories

Here we assume to receive as input a set  $R$  of disjoint spatial regions – each representing a *place* that is relevant for our analysis – which will be used to define a neighborhood function in the following way:

$$N_R(x, y) = \begin{cases} A & \text{if } A \in R \wedge (x, y) \in A \\ \emptyset & \text{otherwise} \end{cases} \quad (1)$$

The neighborhood of a spatial point is the whole region it falls in, i.e., two points are considered similar iff they fall in the same region. All points that are not covered by the regions in  $R$  have an empty neighborhood, meaning that they are not similar to any point (including themselves). The result is that points disregarded by  $R$  will be virtually deleted from trajectories and spatio-temporal patterns.

Static neighborhoods  $N_R()$  greatly simplify the problem of mining T-patterns. Indeed, it results that we can replace ST-sequences with corresponding sequences of regions, thus treating the spatial information only in a preprocessing step<sup>2</sup>:

**THEOREM 1.** *A T-pattern  $(S, A)$  is contained in a ST-sequence  $T = \langle (x_0, y_0, t_0), \dots, (x_n, y_n, t_n) \rangle$  w.r.t. a RoI neighborhood  $N_R()$  iff the  $\mathcal{TAS}$   $(S', A)$  is contained in sequence  $T'$ , where  $S'$  (resp.  $T'$ ) is obtained by mapping each spatial point  $(x, y)$  of  $S$  (resp.  $T$ ) to  $N_R(x, y)$ , removing empty regions.*

The regions associated with each point, i.e.,  $N_R(x, y)$ , are essentially used as labels representing events of the form “the trajectory is in region  $N_R(x, y)$  at time  $t$ ”. Thus, the methods developed for extracting frequent  $\mathcal{TAS}$ 's can be directly applied to the translated input sequences, and each pattern

<sup>2</sup>Due to space limitations, proofs of theorems are not given here, and are available in [6].

( $\mathcal{TAS}$ ) of the form  $A \xrightarrow{\alpha} B$  represents (i.e., can be translated back to) the set of T-patterns  $\{(x, y) \xrightarrow{\alpha} (x', y') \mid (x, y) \in A \wedge (x', y') \in B\}$ . As we can see, in the RoI approach to the problem, the output obtained by the translated input sequences (i.e.,  $A \xrightarrow{\alpha} B$ ) also provides a clear and compact representation of the real set of frequent T-patterns that are contained in the original dataset.

In several contexts the mining problem comes with an *a priori* knowledge of suitable Regions-of-Interests to apply, manually obtained by experts in the application domain or simply through commonsense. For instance, origin-destination matrices are a common tool for the analysis of urban mobility flows, and both origins and destinations are usually given as background knowledge or they are the results of some preliminary study [2]. However, in some cases we do not have these information in advance, and therefore they have to be derived somehow, as discussed in the rest of this section.

### Static preprocessed spatial regions

When Regions-of-Interest are not provided by external means (manually specified by the user, chosen by ad hoc algorithms exploiting some form of background knowledge, etc.) they have to be automatically computed through some heuristics.

This approach is very similar to the previous one, with the difference that regions are automatically derived from actual data, instead of being statically defined a priori. The underlying idea is that locations frequently visited by moving objects probably represent interesting places, as opposed to seldom-visited locations that represent occasional behaviors. E.g., the tourists of a historical town will probably visit the same places of interest, while the routes they follow to move from one interesting place to another may sensibly vary, depending on the specific domain (town or countryside).

The natural way of doing that is to consider the set of locations touched by any moving object, possibly taking into account also the duration of its stay on the locations, and to apply some form of clustering algorithm. In particular, the discussion above suggests to adopt methods that take into consideration the density of spatial regions. A detailed solution for the problem is provided in Section 4.2.

## 4.1 Trajectory preprocessing

Assuming to know a suitable set of RoI, applying them to the T-pattern mining problem simply consists in preprocessing the input sequences to corresponding sequences of RoI.

This process implicitly assumes that the observations that compose the input ST-sequences describe the movement of objects to a sufficient level of detail, which can be satisfactory, e.g. in the case of GPS trajectories, or not, as in the case of GSM trajectories.

Making assumptions about the movement of objects out of the observation points means to provide a model for such movement, for which a wide range of alternatives are available in the spatio-temporal data modeling literature, e.g., linear regression, Bezier’s curves, probabilistic models, constraints models, etc. One of the simplest and most frequently adopted models is the linear regression, which assumes a constant speed, constant direction movement between each pair of consecutive observations. When the full motion of objects is reconstructed, in general an object stays inside a region  $A$  for a time interval  $I$ , instead of a single

instant  $t$ , and therefore it is not obvious which time-stamp should be associated with the *event* “Region  $A$ ” in the translated sequence. A time-stamp should be chosen following some criteria that correctly models the kind of events described in the resulting  $\mathcal{TAS}$ ’s, since the simple one used so far, i.e., “the object is inside the region”, is not applicable any more. The basic solution, which will be adopted in the rest of this paper, consists in choosing the time-stamp in the following way:

- if the trajectory starts at time  $t$  from a point already inside a region  $A$ , yield the couple  $(A, t)$ ;
- in all other cases, take entering times of the trajectory for each region, and associate it with the region name. Notice that an object can enter several times in a region, and each entry will be associated with a different time-stamp.

More advanced solutions could consider exiting times in place of entering times, or both of them, by creating two distinct events  $R_{in}$  and  $R_{out}$  representing the events “enter  $R$ ” and “exit from  $R$ ”.

## 4.2 Discovering Regions-of-Interest

When Regions-of-Interest are not known a priori, some heuristics that enable to automatically identify them are needed. Several different methods are possible:

- selecting among a database of candidate places (e.g., a GIS containing features like restaurants, gyms, shops, etc.) a subset that satisfy some given criteria (e.g., all shops, or all restaurants close to an highway, etc.);
- automatically computing candidate places through the analysis of trajectories, for instance by selecting all minimal square regions that were visited by at least 10% of the objects;
- mixing the two approaches, e.g., by selecting all crossroads where more than the 50% of the crossing trajectories change their direction (i.e., they turn).

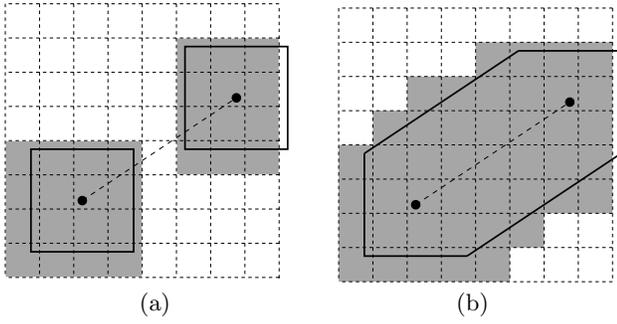
In this section we sketch an example of the second type, that consists in choosing *popular* regions of the space, thus making only use of the input ST-sequences and not considering any form of geographical knowledge.

The approach is developed in two steps: first, dense (i.e., popular) points in space are detected, and then a set of significant regions are extracted to represent them succinctly.

### 4.2.1 Popular points detection

The vague notion of popular region can be formulated in several different ways, one of the most natural being any region of space that is visited by several distinct individuals. Following the philosophy behind the notion of T-patterns, in particular, we can start reasoning at the finest level of granularity and consider first *popular points* in space, i.e., points that are visited by several individuals. Then, adopting the same kind of spatial tolerance introduced in the definition of T-patterns, we can refine the definition by modelling the popularity of a point as the number of distinct moving objects that pass close to it w.r.t. a neighborhood function.

Computing the popularity of points is a distinct count problem (a trajectory touching a point multiple times should



**Figure 2: Density with and without regression**

be counted only once) which becomes complex and computationally expensive if tackled over a continuous space. The complexity of the problem, moreover, can increase depending on the kind of neighborhood used to model the spatial uncertainty (e.g., spherical neighborhoods are more difficult to handle than square ones) and on whether some kind of regression between trajectory points is adopted or not (dealing with regression is more complex than dealing with only the points explicitly contained in the trajectories).

To efficiently compute popular points, we discretize the working space through a regular grid with cells of small size. A typical choice is to set cell width at a given fraction of the chosen neighborhood. Then, the density of cells is computed by taking each single trajectory and incrementing the density of all the cells that contain any of its points or, taking uncertainty into account, all the cells that intersect the neighborhood of any point of the trajectory. An example of that is provided in Figure 2(a), where a trajectory of two points is shown, with a square neighborhood, a grid having cell size equal to  $1/3$  of the neighborhood, and where the cells with incremented density are gray colored. In case trajectories are reconstructed through regression, moreover, the increment is performed over all the cells met along the interpolated curve, including the buffer around the curve induced by the uncertainty neighborhood, as exemplified by Figure 2(b). As discussed above, this process is performed ensuring to increment the density of each cell at most once for each trajectory.

#### 4.2.2 RoI construction

Popular cells represent an extremely fine-grained information that is difficult to handle properly, due to their (typically) large number, but they are a useful basic information for computing larger areas, easier to handle and more meaningful for a pattern extraction task.

In general, the set of popular regions can be extremely large – even infinite, if we work on a continuous space. Therefore, some additional constraints should be enforced to select a significant, yet limited, subset of them. Straight-forward constraints, such as minimality or maximality of the region, fail to provide a reasonable selection, since the single popular cells are all and the only minimal popular regions, while the whole space is the only maximal popular region.

In this paper we adopt a more complex, yet intuitive, definition of popular region, based on the popular cells introduced above.

*Definition 7.* (Popular region set) Given a grid  $\mathcal{G}$  of  $n \times m$  cells, each with its density  $\mathcal{G}(i, j)$  ( $1 \leq i \leq n$  and  $1 \leq j \leq$

$m$ ) and a density threshold  $\delta$ , a *popular region set* for  $\mathcal{G}$  is a collection  $R$  of sets of cells from  $\mathcal{G}$ , such that: (i) each  $r \in R$  forms a rectangular region; (ii) sets in  $R$  are pairwise disjoint; (iii) all dense cells in  $\mathcal{G}$  are contained in some set  $r \in R$ ; (iv) all  $r \in R$  have  $\text{avg}_{(i,j) \in r} \mathcal{G}(i, j) \geq \delta$ ; and (v) assuming that  $r \in R$  has size  $h \times k$ , all its rectangular supersets  $r' \supseteq r$  of size  $(h+1) \times k$  or  $h \times (k+1)$  violate (iv) or  $r$  and  $r'$  contain exactly the same number of dense cells.

Essentially, a popular region set provides a coverage of dense cells through disjoint, rectangular regions with some form of local maximality (condition (v)). In particular, for each region we consider the average density of its cells, instead of its overall density (which is generally higher), and larger rectangles are preferred only if they add dense regions. Finally, we remark that in general there exist more than one popular region set for a grid.

A simple algorithm for extracting one of the possible solutions is provided in Figure 3. The algorithm iteratively considers all popular cells (see cells filtering at (1) and cells scan at (3)) not yet covered by existing popular regions (step (4)). For each of them builds a corresponding single cell region (step (5)) and tries to extend it as much as possible along all four possible directions (steps (7–8)). Then, among the directions that yield admissible larger regions w.r.t. the requirements in Definition 7 (step (9)), if any, it chooses the one that maximizes the average density of the new region<sup>3</sup> (steps (10–12)). The process is repeated as long as the region can be extended, and at the end the resulting region is added to the output set and all its cells are marked to avoid overlaps with next regions (steps (14–15)).

---

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

---

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

```

1.  $R = \emptyset$ ;  $\mathcal{G}^* = \{(i, j) \in \mathcal{G} \mid \mathcal{G}(i, j) \geq \delta\}$ ;
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.          $ext = \{dir \mid r_{dir} \subseteq \mathcal{G} \wedge \text{avg\_density}(r_{dir}) \geq \delta$ 
            $\wedge \exists (h, k) \in (r_{dir} \setminus r). \mathcal{G}(h, k) \geq \delta$ 
            $\wedge \forall (h, k) \in r_{dir}. \neg used(h, k)\}$ ;
10.        if  $ext \neq \emptyset$  then
11.           $dir = \arg \max_{d \in ext} \text{avg\_density}(r_d)$ ;
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  $R$ ;
```

---

**Figure 3: Algorithm for Popular Regions extraction**

In Figure 4 an example of all the steps for the extraction of popular region sets is represented: in (a) we have (a sampling of) the input dataset of trajectories; in (b) the corresponding densities of the cells over a grid of size  $100 \times 100$

<sup>3</sup>Notice that also any alternative choice would yield results compliant with Definition 7.

---

**Algorithm: Static\_RoI-T-pattern**( $T_{in}, \mathcal{G}_0, \delta, \epsilon, \tau$ )

---

Input: A set of input trajectories  $T_{in}$ , a grid  $\mathcal{G}_0$ , a minimum support/density threshold  $\delta$ , a radius for spatial neighborhoods  $\epsilon$ , a temporal threshold  $\tau$ .

Output: A set of couples ( $S, \mathcal{A}$ ) of sequences of regions with temporal annotations.

1.  $\mathcal{G} = \text{ComputeDensity}(T_{in}, \mathcal{G}_0, \epsilon);$  (Sect. 4.2.1)
  2.  $\text{RoI} = \text{PopularRegions}(\mathcal{G}, \delta);$  (Sect. 4.2.2)
  3.  $\mathcal{D} = \text{Translate}(T_{in}, \text{RoI});$  (Sect. 4.1)
  4.  $\mathcal{TAS\_mining}(\mathcal{D}, \delta, \tau);$  ([5])
- 

**Figure 5: Mining frequent T-patterns with static Regions-of-Interests**

are plotted; finally, in (c) the dense cells (w.r.t. a given threshold) are highlighted in dark/red and the popular regions extracted are drawn. As we can see, the resulting regions can contain non-dense cells, yet they are not predominant (due to the average density threshold enforced in the algorithm) and there are no vertical or horizontal bands without at least one dense cell. Some dense cells, however, were too isolated to be merged with others, and so form very small regions. Finally, the regions obtained experimentally show a reasonably balanced shape.

**THEOREM 2.** *The complexity of the PopularRegions( $\mathcal{G}, \delta$ ) algorithm is  $O(|\mathcal{G}| \log |\mathcal{G}|)$ .*

We notice that when the density threshold adopted in this task is equal to the minimum support for the T-patterns, dense cells approximatively represent the set of frequent T-patterns of length 1, i.e., the simplest patterns formed by single spatial points  $(x_0, y_0)$ . However, in the approach outlined above, such dense cells are used to form the regions that will be used to extract all the T-patterns, not only those of length 1, thus making the strong assumption that such cells and groups of cells are meaningful for patterns of any length.

## 5. T-PATTERNS WITH STATIC ROI

The spatial discretization process described in the previous section can be readily integrated to the  $\mathcal{TAS}$  mining algorithm in [5] to obtain a simple method for extracting T-patterns following the approach based on (static) Regions-of-Interest. Figure 5 summarizes the resulting algorithm.

Projecting all the input trajectories over the given grid  $\mathcal{G}_0$ , and using a spatial neighborhood or radius  $\epsilon$ , step (1) computes the density of each cell of the grid. Next, a set of Regions-of-Interest are automatically computed by means of the popular region extraction method (step (2)). Then, exploiting Theorem 1, in step (3) the input trajectories are preprocessed to sequences of regions w.r.t. the RoI obtained at the previous step, and the preprocessed trajectories are used as input for the  $\mathcal{TAS}$  mining algorithm, whose output is also the output of our main algorithm.

## 6. T-PATTERNS WITH DYNAMIC ROI

In this section we discuss the complexity of the T-pattern mining problem in its more general variant and provide an approximated algorithm for them, based on Regions-of-Interest dynamically computed within the mining process.

## 6.1 Dynamic neighborhood approach

The RoI-based neighborhoods discussed in the previous sections, although not very intuitive at a first glance, lead to a simple, yet useful, instantiation of the general frequent T-pattern mining problem.

However, in some contexts such a solution, based on fixed or pre-computed regions, is not significant, while more standard and general neighborhood functions, on the contrary, can provide a better choice for the notion of spatial similarity we need to model. Examples of that are *spherical neighborhoods* ( $N_\epsilon(\bar{x}) = \{\bar{y} \mid \|\bar{x} - \bar{y}\|_2 \leq \epsilon\}$ , for a given  $\epsilon$ ) or *square neighborhoods* ( $N_\epsilon(\bar{x}) = \{\bar{y} \mid \|\bar{x} - \bar{y}\|_\infty \leq \epsilon\}$ ), respectively defining a circle of radius  $\epsilon$  and a square of side  $2\epsilon$  centered around the given point.

When we adopt such standard neighborhoods, we have to face the T-pattern mining problem in its full generality, which requires to return every T-pattern that *fits* in several input ST-sequences. Fitting in a ST-sequence  $T$ , in particular, means that  $T$  contains  $n$  distinct points (i.e., a sequence of  $n$  observations) that match the  $n$  points in the T-pattern in the sense of Definition 5.

As we can see, understanding which input sequences  $T$  support a T-pattern in principle requires to check all subsequences of  $T$  searching for anyone that matches with it, i.e., anyone whose  $n$  points and  $n - 1$  transition times have the corresponding items of the T-pattern within their neighborhoods. Similar to what was done with for  $\mathcal{TAS}$ 's, we can see the problem from the opposite viewpoint: 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  $T$* . Then, frequent T-patterns are those that fall in the neighborhood of several input ST-sequences, which is a plain formulation of a kernel-based 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<sup>4</sup>. The kernel adopted on this space is simply the product of the neighborhoods applied for each element of the  $n$ -ple (resp.  $(n - 1)$ -ple), i.e., an  $N_\epsilon()$  (or other spatial neighborhood) for each point and  $N_\tau()$  for each transition time. E.g., looking for all frequent T-patterns of length 3 with spatial neighborhood  $N_\epsilon()$  is equivalent to find all dense points in  $\mathbf{R}^8$ , where  $8 = 2 \times 3 + 2$ , i.e., 2 dimensions for each point in the T-pattern and 1 for each transition time.

In the general case extracting frequent T-patterns translates to a density-estimation problem over spaces of rapidly-growing dimensionality, i.e., over  $\mathbf{R}^{3n-1}$  ( $n$  being the pattern length). In particular, as compared to the  $\mathcal{TAS}$  mining problem, T-patterns exacerbate the difficulty of the density estimation task in two ways: (i) the dimensionality of working spaces of  $\mathcal{TAS}$ 's grows less quickly ( $\mathbf{R}^{n-1}$  instead of  $\mathbf{R}^{3n-1}$ ); (ii) the sequence component in each  $\mathcal{TAS}$  strongly limits the number of instances that can be found within each input sequence, making the density estimation task easier.

---

<sup>4</sup>More precisely, in our context the neighborhoods of all subsequences of a single ST-sequence  $T$  are aggregated through set union, i.e., they do not sum up their contributions to density, while contributions are summed up for neighborhoods of different ST-sequences. We are essentially facing a *multi-instance* density-estimation problem, that introduces a complication to the standard density-estimation setting.

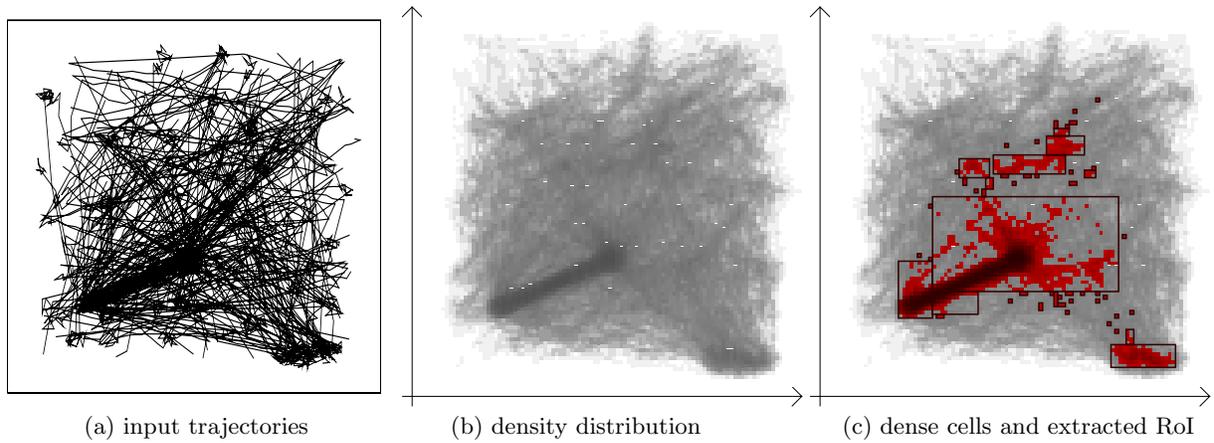


Figure 4: Example of RoI extraction

From the discussion given above, we can easily deduce that in a general setting the T-pattern mining problem becomes intractable even for small pattern lengths. That motivates the development of simpler heuristic approaches, that try to extract the same kind of information contained in *pure* T-patterns but with some simplifying approximation.

## 6.2 A step-wise heuristic

The main issue in the general T-pattern mining problem is that we need to consider density for space and transition times in parallel, making it a high-dimensional problem.

Here a trade-off solution is provided, that approximates the more general instantiations of the T-pattern problem, by adopting a step-wise approach in building T-patterns, and approximating sets of similar T-patterns by means of a RoI-based representation.

### 6.2.1 Principles of the method

We start by observing that the set  $P_0$  of T-patterns of length 1 for a dataset of trajectories  $T_0$  consists of all points  $(x_0, y_0)$  over  $\mathbf{R}^2$  that are dense w.r.t. the given neighborhood function, i.e., all points that are touched or that fall close to a sufficient number of the input trajectories.

In order to discover all patterns  $P_1$  of length 2, i.e., of the form  $(x_0, y_0) \xrightarrow{\Delta t_1} (x_1, y_1)$ , the general definition requires to examine the occurrences of each input trajectory in the representation space  $\mathbf{R}^5$  looking for dense points, as discussed in Section 6.1. However, that will yield patterns whose first points,  $(x_0, y_0)$ , are a subset of those found in  $P_0$ , and, more generally, any frequent T-pattern of length  $n + 1$  is the extension of some frequent T-pattern of length  $n$ , as stated by the following property.

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

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

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

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

This property implies that the support of a T-pattern is less than or equal to the support of any its prefixes, and thus

allows us to adopt a level-wise approach by mining step-by-step patterns of increasing length. Moreover, it also states that, in a similar way, transition times can be searched in a separate step, after finding the *interesting* spatial points.

A critical aspect of this approach is the fact that at each step we have in general an infinite number of possible points to consider, e.g., the patterns in  $P_0$  correspond to all the points that fall in any dense region of  $\mathbf{R}^2$  w.r.t. the neighborhood function. To deal with this problem, we introduce an approximation of the basic method, where points are not treated separately, but at each step are clustered together by following the approach described in Section 4.2, to form basic regions that are treated as an indivisible entity. As in the case of static RoI, the set of regions derived this way can be used to translate trajectories to sequences composed of such regions, which will be used to extend the actual pattern.

Notice that each occurrence of a region in the translated sequence is associated with a time stamp, which is the time stamp of the corresponding original point in the trajectory, or, in case the trajectory was reconstructed through regression, the entry time of the trajectory in the region.

### 6.2.2 Implementation of the method

Exploiting the monotonicity property provided by Theorem 3, we can safely search for any frequent pattern  $p_{n+1} = (x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_{n+1}} (x_{n+1}, y_{n+1})$  by analyzing only the trajectories that are already known to contain the subpattern  $p_n = (x_0, y_0) \xrightarrow{\Delta t_1} \dots \xrightarrow{\Delta t_n} (x_n, y_n)$ . Moreover, only a segment of such trajectories really needs to be searched, since we only need to find *continuations* of the pattern  $p_n$ , and no point occurring before the end time of  $p_n$  can be appended to  $p_n$  to obtain  $p_{n+1}$ . Therefore, any point occurring before such end time can be removed from the trajectory. That essentially means that we can follow a projection-based approach, as adopted by PrefixSpan [9] for sequential patterns and by MiSTA [5] for frequent  $\mathcal{IAS}$ 's.

The main difference from existing methods for mining sequential patterns or episodes is that our input data and the projections obtained at each step will contain trajectories, rather than sequences of events or itemsets. Then, at each step we dynamically derive the interesting regions from the trajectory segments of the actual projection, use such RoI to translate the whole projection to a set of sequences of

regions, and finally apply a standard prefix extension step over the resulting sequences.

The algorithm is summarized in Figure 6. Its structure closely follows the algorithm introduced in [5] for mining frequent  $\mathcal{TAS}$ 's and shares the same representation for temporal annotations (i.e., the transition times associated to each occurrence of the prefix in the sequences of a projection) and the same functions for handling them<sup>5</sup>(see steps (6), (8) and (14)). In particular, each projection is associated with a prefix (the pattern built so far) and is formed by a set of trajectories, each trajectory being associated with a set of annotations, representing the timings of all occurrences of the prefix in the trajectory – they are used by function *ExtractFrequentTimings* to extract frequent temporal annotations.

The algorithm proceeds in a level-wise style and iteratively consider all existing projections of increasing prefix size (steps (2–4)). If the the actual pattern has transition times to be evaluated (step (5)) – i.e., it contains at least two regions – we compute its frequent temporal annotations (step 6). The resulting set of frequent annotations is returned and used to prune the projection by removing useless trajectories, i.e., those that were not associated with any frequent annotation (steps (7–8)). Then, in order to extend the actual pattern by one step, we first analyze the trajectories in the projection to discover the new RoI (steps (9–10)) and use them to translate trajectories to corresponding sequences of RoI (step 11). Now, the translated trajectories are used to extend the prefix by all admissible regions (steps (12–13)), i.e., all those having a sufficiently high support. For each region  $r$ , the extension of the prefix is performed by function *ExtendProjection* in step (14), which selects the translated trajectories where  $r$  occurs and updates the corresponding temporal annotations (refer to [5] for further details). Then, the results of the prefix extension are reproduced in the trajectories of the original projection (step (15)), by removing the trajectories corresponding to the removed sequences of RoI (the link between them is kept by a trajectory ID) and by removing the segments of trajectory that cannot be used for future prefix extensions (i.e., those that precede the first end time of all prefix occurrences, as described at the beginning of this section). Finally, at step (18) the updated projection and the new prefix (one region longer than the original one) are queued for processing at the next iteration of the main cycle (step (2)). The computation stops when no more extensions are possible, and so no new projections are produced by steps (12–18).

## 7. EXPERIMENTS

In this section we summarize the results of a set of experiments aimed at showing some sample trajectory patterns obtained from real data and at empirically evaluate the scalability of the proposed algorithms.

The real data used in these experiments describe the GPS traces of a fleet of 273 trucks in Athens, Greece, for a total of 112203 points<sup>6</sup>. Running both the *Static\_RoI\_T-pattern* and *Dynamic\_RoI\_T-pattern* algorithms with various parameter

<sup>5</sup>More exactly, step (6) summarizes three operations introduced in [5]: *Extract\_annotation\_blocks*, *Compute\_density\_blocks*, and *Coalesce\_density\_blocks*; step (8) corresponds to function *Annotation-based-prune*; and step (14) corresponds to *extend\_proj*.

<sup>6</sup>Download at <http://is1.cs.unipi.gr/db/projects/rtreeportal/>

---

### Algorithm: *Dynamic\_RoI\_T-pattern*( $T_{in}, \mathcal{G}_0, \delta, \epsilon, \tau$ )

---

Input: A set of input trajectories  $T_{in}$ , a grid  $\mathcal{G}_0$ , a minimum support/density threshold  $\delta$ , a radius for spatial neighborhoods  $\epsilon$ , a temporal threshold  $\tau$ .

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  $\text{support}_{\mathcal{D}}(r) \geq \delta$  then
14.           $\mathcal{D}' = \text{ExtendProjection}(\mathcal{D}, r);$       ([5])
15.           $T' = \{ (traj, \mathcal{A}') \mid (traj, \mathcal{A}) \in T$ 
16.                 $\wedge (S', \mathcal{A}') \in \mathcal{D}' \wedge traj.ID = S'.ID$ 
17.                 $\wedge traj' = \text{Cut}(traj, \mathcal{A}') \}$ 
18.           $T_{L+1} = T_{L+1} \cup \{(T', \text{append}(prefix, r))\};$ 
19.       $L++;$ 

```

---

**Figure 6: Mining frequent T-patterns with dynamic Regions-of-Interest**

settings (the same for both algorithms), several frequent trajectory patterns were discovered. Among them, two patterns are shown in Figure 7(left) over a simplified plot of the roads of the area, and provide an example of how the same kind of pattern hidden in the data is detected by the two different approaches, static and dynamic. The T-pattern  $A \xrightarrow{\Delta t_1} B' \xrightarrow{\Delta t_2} B''$  was discovered through the dynamic approach, with main temporal annotations  $(\Delta t_1, \Delta t_2) \in [330, 445] \times [116, 190]$  and  $(\Delta t_1, \Delta t_2) \in [400, 513] \times [41, 61]$  (the detailed list of annotations is omitted here for lack of space), and essentially represents frequent movements similar to a round trip, since the first and last regions are adjacent. The corresponding static T-pattern  $A \xrightarrow{\Delta t_1} B \xrightarrow{\Delta t_2} B$  provides a coarser description of the same behavior, composed of larger regions ( $B$  replaces  $B'$  and  $B''$ ), with the presence of a loop and associated to a larger set of annotations (not listed for lack of space). It is quite clear that the dynamic T-pattern provides more precise information than its static counterpart.

The performances of the static and dynamic algorithms have been studied by means of synthetic data, generated by the CENTRE synthesizer described in [4]. Data contain 50% of purely random trajectories and 50% of trajectories that follow predefined patterns, randomly chosen among a set of 100 (random) patterns. Where not otherwise specified, the datasets contain 1000 trajectories, each one described on average by 200 points, and the algorithms are run with a minimum support/density threshold equal to 5%, spatial neighborhood of radius 10, time tolerance equal to 300 and density grid of size 100x100.

Figure 7(center) reports the execution times obtained by both algorithms over datasets with increasing number of input trajectories. The curves show an almost linear scalabil-

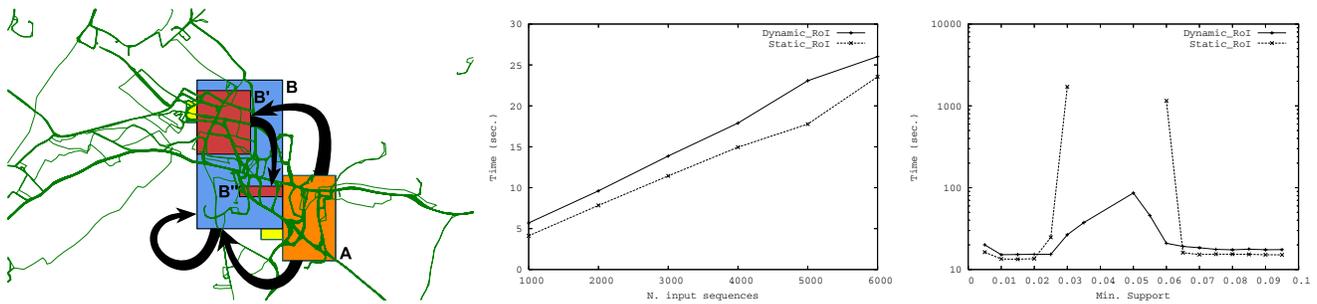


Figure 7: Sample T-patterns (left), algorithm scalability over input size (center) and min. support (right)

ity with very similar times for both algorithms. Observing the running times for different minimum support thresholds (also used as density threshold for the region extraction process), then, we can see in Figure 7(right) that, interestingly, running times grow when the minimum support is decreased, as expected, but then quickly decrease when it goes below a given value, thanks to the effect of density on the region extraction process, which partitions the space in a smaller number of regions, thus causing the main algorithm to generate less projections. We observe, moreover, that while the static algorithm performs better on extreme support threshold values, its execution times grow much faster when closing to the critical middle values, exceeding the time limits enforced in these experiments (60 minutes).

The impact of other parameters was also investigated, but for lack of space we give here only a brief summary of the results: increasing the radius of the spatial neighborhood ( $\epsilon$ ) we obtain irregular performances, yet very large values lead to very high execution times; changing the time tolerance threshold ( $\tau$ ) we obtain results very similar to computing  $\mathcal{TAS}$ 's (see [5]); finally, increasing the number of points in each trajectory causes a quasi-linear growth of times.

## 8. CONCLUSION

In this paper, we introduced the trajectory pattern mining problem, together with several different methods to extract T-patterns from trajectory data. The first empirical assessment reported in Section 7 exhibits promising results, from the point of view of both computational feasibility and usefulness of the mined T-patterns. In our view, T-patterns are a basic building block for spatio-temporal data mining, around which more sophisticated analysis tools can be constructed, including:

- integration with background geographic knowledge, such as road networks and other geographic information layers, at the level of trajectory pre-processing, RoI discovery, T-patterns mining and post-processing;
- adequate visualization metaphors for T-patterns, as well as integrations into visual analytics methods and tools for exploratory trajectory pattern mining;
- adequate mechanisms for spatio-temporal querying and reasoning mechanisms on both input trajectories and extracted T-patterns, including refinements of interesting T-patterns.

Research efforts are being put in these directions, as well as in a large-scale experiment aimed at testing the concrete

usefulness of trajectory pattern mining in the analysis of traffic flows in a metropolitan area.

## 9. REFERENCES

- [1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of ICDE*, 1995.
- [2] K. Ashok. *Estimation and Prediction of Time-Dependent Origin-Destination Flows*. PhD thesis, Massachusetts Institute of Technology, 1996.
- [3] H. Cao, N. Mamoulis, and D. W. Cheung. Mining frequent spatio-temporal sequential patterns. In *ICDM*, 2005.
- [4] F. Giannotti, A. Mazzoni, S. Puntoni, and C. Renso. Synthetic generation of cellular network positioning data. In *GIS '05: Procs of 13th ACM Int. Workshop on Geographic Information Systems*, pages 12–20, New York, NY, USA, 2005. ACM Press.
- [5] F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of sequences with temporal annotations. In *Proc. SIAM Conference on Data Mining*, pages 346–357. SIAM, 2006.
- [6] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. T-patterns: temporally annotated sequential patterns over trajectories. Technical report, ISTI-CNR, 2006.
- [7] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *Proceedings of 9th International Symposium on Spatial and Temporal Databases*, pages 364–381. Springer, 2005.
- [8] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *KDD*, 2004.
- [9] J. Pei et al. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–225, 2001.
- [10] A. Vautier, M.-O. Cordier, and R. Quiniou. An inductive database for mining temporal patterns in event sequences. In *Proceedings of the workshop on Mining Spatial and Temporal Data*, 2000.
- [11] M. Yoshida et al. Mining sequential patterns including time intervals. In *Data Mining and Knowledge Discovery: Theory, Tools and Technology II (SPIE Conference)*, 2000.
- [12] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.