Unsupervised Trajectory Sampling

Nikos Pelekis[1], Ioannis Kopanakis[2], Costas Panagiotakis[3], and Yannis Theodoridis[4]

[1] Dept. of Statistics and Insurance Science, Univ. of Piraeus, Greece
npelekis@unipi.gr
[2] Tech. Educational Inst. of Crete, Greece
i.kopanakis@emark.teicrete.gr
[3] Dept. of Computer Science, Univ. of Crete, Greece
cpanag@csd.uoc.gr
[4] Dept. of Informatics, Univ. of Piraeus, Greece
ytheod@unipi.gr

Abstract. A novel methodology for efficiently sampling Trajectory Databases (TD) for mobility data mining purposes is presented. In particular, a three-step unsupervised trajectory sampling methodology is proposed, that initially adopts a symbolic vector representation of a trajectory which, using a similarity-based voting technique, is transformed to a continuous function that describes the representativeness of the trajectory in the TD. This vector representation is then relaxed by a merging algorithm, which identifies the maximal representative portions of each trajectory, at the same time preserving the space-time mobility pattern of the trajectory. Finally, a novel sampling algorithm operating on the previous representation is proposed, allowing us to select a subset of a TD in an unsupervised way encapsulating the behavior (in terms of mobility patterns) of the original TD. An experimental evaluation over synthetic and real TD demonstrates the efficiency and effectiveness of our approach.

Keywords: Trajectory Databases, Sampling, Symbolic Trajectories.

1 Introduction

Data analysis and knowledge discovery over trajectory databases [8] discovers behavioral patterns of moving objects that can be exploited in several fields. Example domains include traffic engineering, climatology, social anthropology and zoology, studying vehicle position data, hurricane track data, human and animal movement data, respectively. In the literature, there have been proposed several works that try to analyze trajectory data either in an exploratory way [1], [3], [4] or by mining movement-aware patterns, such as clusters of moving objects [5], [15], [12], [19], [14], [10], sequential trajectory patterns [7], and flock patterns [9].

All of these approaches usually operate on large TD and it is natural for an analyst to wonder whether she could extract the same patterns operating on a much smaller and representative subset of the TD. In other words, the question is rephrased to whether one can appropriately reduce a large TD, taking only a sample of it, whose size is automatically computed, in an optimized and unsupervised way, and which

J.L. Balcázar et al. (Eds.): ECML PKDD 2010, Part III, LNAI 6323, pp. 17–33, 2010.

encapsulates the mobility patterns hidden in the whole TD. Fig. 1 illustrates the above discussion. Let us assume that mining a TD (Fig. 1(a)) results in a set of movement patterns (say, those illustrated by different colours in Fig. 1(a)). The question is whether we could extract an appropriate subset of the original TD (Fig. 1(b)) that would again capture the same patterns. If the answer to the above question is yes, such a methodology would radically speed up several analysis and mining tasks in the field.

Fig. 1. (a) original TD, (b) sample TD

The problem of trajectory sampling is very challenging due to the complexity of movement data (e.g. lack of ordering, lack of compact representation) that makes standard (e.g. uniform or stratified sampling) or point density-biased sampling techniques (e.g. [17], [11], [16]) inappropriate for the TD domain. A naïve solution for trajectory sampling would first cluster the TD and select the centroids of each cluster as the TD samples. However, existing trajectory-oriented clustering algorithms provide a single representation associated with each cluster (called *Representative* [12] or *Centroid* trajectory [19]). In addition, the aim of clustering is to partition the data into groups and not to downsize the TD. More importantly, such a solution would fail to select trajectories important for mobility patterns other than clusters (e.g. sequential trajectory patterns [7], or trajectory outliers [13]). To the best of our knowledge, it is only explorative, supervised sampling techniques that have been proposed for TD [3], [4], which however suffer from the above mentioned problems, as they imply a-priori knowledge of the underlying trajectory clusters.

In this paper, we propose a three-step approach to tackle the problem of trajectory sampling. Initially, we adopt a symbolic representation of trajectories that allows us to model all trajectories of the TD in an approximate way as vectors (i.e., sequences of regions wherefrom a moving object passes), whose dimensionality implies the intended space-time granularity of the application. This symbolic representation preserves the mobility pattern of each trajectory (by vanishing jerky movements), speeds up computations, and, moreover, turns out to be lossless in terms of mobility patterns, as shown in our experimental study.

On top of this representation, we propose an effective method to represent each trajectory by a continuous function that implicitly describes the *representativeness* of each constituent part of it (i.e., dimension) with respect to the whole TD. Given such an intuitive representation, we devise an algorithm, called *SyTra* (*Symbolic Trajectory*), which improves the initial representation of each trajectory by relaxing its vector representation. The idea is to adopt a merging algorithm that identifies the maximal time period wherein the mobility pattern of each trajectory is preserved, while in this augmented period it presents uniform behavior in terms of *representativeness*.

In other words, this merging process is an optimization trade-off between local (trajectory's mobility pattern) and global (representativeness in TD) criteria.

At the third step of our approach, we propose an automatic method for trajectory sampling, called *T-Sampling*, based on the representativeness of the trajectories. An important aspect of this method is that it takes into account not only the most (i.e., dense, frequent) but also the least representatives, which are quite interesting in various application scenarios (e.g. detecting movement outliers or sparse clusters), thus they should somehow survive in the sample TD.

The contributions and merits of our work are summarized below:

- Based on a symbolic vector representation of a trajectory, we propose a global voting method allowing us to quantify the *representativeness* of each trajectory in a TD as a continuous descriptor.
- We devise an algorithm (*SyTra*) that relaxes the time-based point representation of the centroid trajectories, allowing the modeling of the mobility pattern of each trajectory at a higher level abstraction, as well as in a 'homogenous' way according to its representativeness in TD.
- We propose a novel unsupervised method (*T-Sampling*) for sampling the representative trajectories in a TD. The cardinality of the sample is either given as input or selected by the method itself.
- Finally, we conduct a comprehensive set of experiments over synthetic and real trajectory datasets, in order to thoroughly evaluate our approach.

The rest of this paper is structured as follows: Section 2 discusses related work. In Section 3, we set the scene by formulating the problem. The proposed methodology for trajectory sampling is presented in Section 4. In Section 5, we conduct an experimental study over synthetic and real TD in order to evaluate our approach. Finally, Section 6 concludes the paper.

2 Related Work

Recently there is an increasing interest in TD mining. Trajectory sampling is a very challenging task with great potential applications in TD mining, however very limited work has been carried out. Among the various proposals for the discovery of mobility patterns, the works in [12], [19] further provide aggregate representation of the extracted patterns (representative trajectories).

In [12], the authors proposed TRACLUS, a partition-and-group framework for clustering 2D trajectories which enables the discovery of common sub-trajectories. TRACLUS clusters trajectories as line segments (sub-trajectories). The notion of the *representative trajectory* of a cluster is also defined. In this approach, the temporal information is not taken into consideration, while the partitioning is performed per trajectory and it does not use global criteria. This is in contrast to our approach where our merging algorithm, which can also be considered as a segmentation method, views trajectories as sequences of sub-trajectories that are identified using global criteria. Furthermore, in contrast to our work, the representative sub-trajectories identified by TRACLUS conform to straight movement and cannot identify complex (e.g. snakelike) patterns, which are common in real world applications.

In [19], an approach for clustering trajectories as a whole is proposed, using a symbolic representation of trajectories and local criteria, also with special care for handling uncertainty. In particular, a density- as well as similarity-based algorithm, called *CenTra*, is proposed in order to discover the *centroid* of a group of trajectories. Then, an FCM variation, called *CenTR-I-FCM*, clusters trajectories by utilizing CenTra. In comparison to the present work where we target at sampling, this approach also uses a global but static and predefined temporal segmentation of trajectories, which are symbolically represented as intuitionistic fuzzy vectors.

For trajectory segmentation, related work includes [1], [18]. In [1], global distance-based criteria have been proposed for the segmentation of trajectories using spatiotemporal information. The distance-based segmentation problem is given as a solution of a maximization problem that attempts to create minimum bounding rectangles (used as simplifications of parts of trajectories) in such a way that the original pairwise distances between all trajectories are minimized. In comparison with our merging approach, we also use global criteria; however these are entailed by the representativeness of each trajectory with respect to the TD. At the same time, our merging process gives special attention to preserve the movement pattern of each trajectory, as our ultimate goal is to succeed effective sampling. In [18], an approach for expressing the representativeness in a TD via a voting process is proposed. This process is improved in the present work in order to support the first step of our methodology, which is to describe the representativeness of each trajectory in the TD.

Explorative TD analysis using visual techniques is proposed in [3], [4]. The authors use uniform sampling to minimize the volume of the trajectories that can be clustered by a density-based clustering algorithm [15]. This approach suffers from the limitations discussed in the introduction, as sampling is supervised by the user and solely depends on the results of the clustering.

Finally, in [17], [11], [16], density biased sampling (DBS) techniques for point sets are proposed, which, obviously, cannot find straightforward application in TD. Nevertheless, in our approach we extend the idea of DBS in a way that density properties as well as the similarity of trajectories segments are taken into account.

Summarizing, although very interesting as a problem and with great potentials in the TD domain, to the best of our knowledge there is no related work on unsupervised trajectory sampling taking the complex nature of TD into consideration.

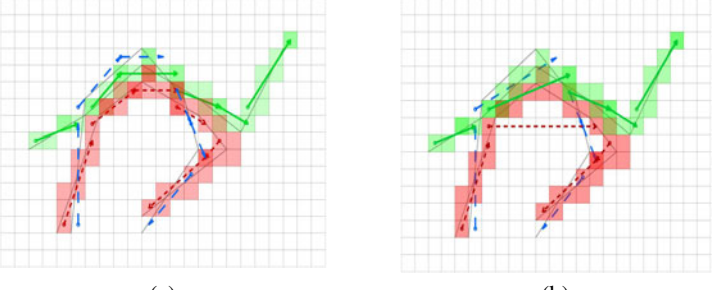
3 Setting the Scene

Let $D = \{T_1, T_2, \ldots, T_N\}$ be a dataset of N trajectories. Assuming linear interpolation between consecutive time-stamped positions, a trajectory $T_i = <(x_{i,0}, y_{i,0}, t_{i,0}), \ldots, (x_{i,n_i}, y_{i,n_i}, t_{i,n_i})>$, consists of a sequence of $n_i > 0$ line segments in 3D space, where the j-th segment interpolates positions sampled at time $t_{i,j-1}$ and $t_{i,j}$. In this paper, we adopt an approximate representation of trajectories, originally proposed in [19], by transforming trajectories in a space where each T_i is represented as a p-dimensional point. (In this way, as shown in [19], we speed up computations, at the same time vanishing negligent movements while preserving the shape designating the mobility pattern of each trajectory.) More specifically, we use an approximation technique and define the dimensionality of trajectories by dividing the lifespan of each trajectory in p sub-intervals (e.g. 1 min. periods). Regarding the spatial dimension, we assume a regular grid of equal rectangular

cells with user-defined size (e.g. 100×100 m^2). Given this setting, we extend the technique used in [19] where each trajectory T_i is partitioned into $p << n_i$ equi-sized temporal periods and substitutes the trajectory 3D line segments of each period with (a) the set of the grid cells that T_i crosses during this period and (b) a motion vector $\vec{d}$ that specifies the direction of the trajectory during the period. Formally:

Definition 1. *Given a regular grid G of granularity $m \times n$ consisting of cells $c_{k,l}$ ($1 \le k \le m$ and $1 \le l \le n$), the lifespan ls of all trajectories in the trajectory database D, a trajectory T_i in D as a sequence of n_i line segments, and a target dimension $p << n_i$, the approximate trajectory (ApTra) $\overline{T_i} = \langle (r_{i,1}, \vec{d}_{i,1}), \ldots, (r_{i,p}, \vec{d}_{i,p}) \rangle$ of T_i is the one resulted by T_i when all trajectory triplets $(x_{i,j}, y_{i,j}, t_{i,j})$ of T_i found inside a temporal period $p_j = \left[\frac{ls \cdot (j-1)}{p}, \frac{ls \cdot j}{p} \right]$, $1 \le j \le p$, are replaced by a pair $(r_{i,j}, \vec{d}_{i,j})$, called directed region $dr_{i,j}$, consisting of a region $r_{i,j}$, which is composed by the set of cells $c_{k,l}$ crossed by T_i during p_j and a directed segment $\vec{d}_{i,j}$ starting from the center of the first cell and ending at the center of the last crossed cell.* ■

This technique allows us to view all trajectories in D as vectors in the same, user-defined dimensionality p, where each value of the vector corresponds to a dynamic time-ordered list of cells crossed by the trajectory, accompanied with the corresponding motion vector. Note that, depending on the choice of the space-time granularity of grid G, intended by the application requirements, a trajectory may introduce *gaps* (i.e., regions with empty set of cells) when there is no motion during the particular period of time. Fig. 2(a) depicts three trajectories and their corresponding *ApTra* illustrated in different colours (red, green, blue). The dimensionality p is set to $p = 6$, the arrows imply direction, while, regarding the blue trajectory it is only the motion vectors that are coloured, for clarity.



(a) (b)

Fig. 2. (a) ApTra for 3 trajectories (b) Merged ApTra

(a) (b)

Fig. 3. (a) ReTra before merging (b) ReTra after merging

The *representativeness* of a trajectory in TD is then defined by extending the definition of density biased sampling (DBS) in point sets [11], [17] for trajectories segments. According to DBS, the local density for each point of the set is approximated by the number of points in a region, divided by the volume of the region. In our case, the representativeness of a trajectory corresponds to the number of other trajectories that are similar to that in terms of time, space, and direction. Technically speaking, the representativeness is calculated by a voting process that is applied for each directed region pair $dr_{i,j}$ of $\overline{T}_i$, extending and improving a preliminary version of the proposed method, presented in [18]. Thus, $dr_{i,j}$ will be voted by the approximate trajectories of TD, according to the distance of $dr_{i,j}$ to each trajectory of the TD. The sum of these votes is related to the number of trajectories that are similar to $dr_{i,j}$, called hereafter v($dr_{i,j}$). More formally:

Definition 2. *Given an ApTra* $\overline{T}_i = \langle (r_{i,1}, \vec{d}_{i,1}), \ldots, (r_{i,p}, \vec{d}_{i,p}) \rangle$ *and a voting function v(.) we define a representative trajectory (ReTra) as a set of triplets* $(r_{i,j}, \vec{d}_{i,j}, v(dr_{i,j}))$ *where the third value corresponds to the representativeness of each directed region (i.e., sub-trajectory)* $dr_{i,j}$ *of ApTra.* ■

Under this definition, the representativeness is formulated as a continuous descriptor $v(dr_{i,j})$ over $j = 1, \ldots, p$. Fig. 3(a) depicts the descriptors $v(dr_{i,j})$ for the trajectories of Fig. 2(a) where one can see the votes each trajectory collected in each of the six periods (vertical lanes), as well as to realize that the red and blue trajectories appear to have similar mobility patterns, while this is not the case for the green trajectory. A novel approach for the extraction of *ReTra* is given in Section 4.1.

Viewing dimensionality from a different perspective, one may consider it as a segmentation of the trajectories in time axis. Although intuitive, this segmentation is static and predefined for the whole TD. As we aim at sampling whole trajectories, we argue that a more intuitive representation would be to aggregate each trajectory along the temporal dimension. Such a representation would model each trajectory with a (possibly) different number of time periods (less than p) of varying longer duration. Each such period would be the result of merging successive periods (and the respective regions and motion vectors) of the initial representation, having as aim to identify the maximal temporal periods during which some uniformity criterions hold. These criteria are a trade-off between the preservation of the (local) mobility pattern of each trajectory and the uniformity of the (global) representativeness of the merged directed regions.

Definition 3. *Given a representative trajectory (ReTra) of p triplets* $\langle (r_{i,j}, \vec{d}_{i,j}, v(dr_{i,j})) \rangle$, $j = 1,\ldots, p$, *a threshold* Δd *over the distance* D_d *between successive motion vectors of ReTra (see Definition 7), and a threshold* Δv *over the difference between successive values of* $v(dr_{i,j})$, *we define its symbolic trajectory (SyTra) as a set of triplets* $(sr_{i,k}, \overrightarrow{sd}_{i,k}, v(sdr_{i,k}))$, $k = 1,\ldots, K_i$, *where* $K_i \leq p$ *denotes the number of directed regions* $dr_{i,j}$ *that are merged to define a symbolic directed region* $sdr_{i,k}$, $sr_{i,k}$ *is the union of successive* Δv*-similar regions* $r_{i,j}$, $\overrightarrow{sd}_{i,k}$ *is the motion vector of* $sr_{i,k}$ *where the motion vectors of the merged* $\vec{d}_{i,j}$ *are* Δd*-similar, and* $v(sdr_{i,k})$ *is the representativeness (see Definition 2) after the merging.* ■

In other words, the *symbolic trajectory* (*SyTra*) can be considered as a trajectory segmentation of into K_i sub-trajectories, where each sub-trajectory contains successive regions that are similar to each other with respect to trajectory representativeness (*Δv-similar*) and shape (*Δd-similar*), thus providing a compact representation of *ReTra*.

Similarly to Definition 2, the *SyTra based representativeness* descriptor shows how many objects follow the specific *k*-sub-trajectory at almost the same time, space and direction. Note here that, as the symbolic directed regions $sdr_{i,k}$ are different from directed regions $dr_{i,j}$ the voting process is applied again to *SyTra*. Fig. 2(b) illustrates the results of merging the three *ReTra* of Fig. 2(a), while Fig. 3(b) illustrates the updated *representativeness* of the new voting process. A novel algorithm for computing *SyTra* is provided in Section 4.2.

Definition 4. *The trajectory representativeness descriptor V_i of a SyTra is defined as the weighting average over $v(sdr_{i,k})$,*

$$V_i(D) = \frac{\sum_{k=1}^{K_i} \delta t_{i,k} \bullet v(sdr_{i,k})}{\sum_{k=1}^{K_i} \delta t_{i,k}} \tag{1}$$

where weight $\delta t_{i,k}$ denotes the duration of k-sub-trajectory. ■

The trajectory representativeness descriptor V_i is to be used for the trajectory sampling problem. Actually, the goal of trajectory sampling is to select the most representative trajectories of the TD. However, by selecting the top-voted sub-trajectories, which sounds to be an obvious decision, the high density regions of the TD will be oversampled, resulting in a non-representative sample. On the contrary, we propose the sampling to be considered as an optimization problem that aims to maximize a formula (Equation 2), taking into account the representativeness $V_i(D)$ in the original TD D as well as the representativeness $V_i(S)$ in the sample S ($V_i(S)$ is defined similarly to $V_i(D)$). So, our goal is that S should contain trajectories of high representativeness trying at the same time to cover the full space of D. Recalling Fig. 2, this implies that a reasonable decision of an intuitive sampling algorithm would be to select the green and either the red or the blue trajectory in case that the top-2 representative trajectories are requested (i.e., $|S| = 2$).

Definition 5. *Let S_i be equal to one when trajectory T_i of dataset D belongs to the sampling set, and equal to zero otherwise. The Trajectory Sampling problem is defined as the optimization of finding an appropriate subset S of D, which maximizes the function SR(S):*

$$SR(S) = \sum_{i=1}^{N} S_i \bullet V_i(D) \bullet (1 - V_i(S)) \tag{2}$$

This implies that the number of trajectories in D that find their representatives in S is maximized. ■

Obviously, the idea of $SR(S)$ is to favour $V_i(D)$ and penalize $V_i(S)$, at the same time producing a non-monotonously increasing $SR(S)$ with N. A novel solution to the *Trajectory Sampling problem* is provided in Section 4.3.

4 Sampling Trajectories

In this section, we present our approach for trajectory sampling in detail. The proposed methodology (consisting of three steps) is sketched by an algorithm, called *T-Sampling* and illustrated in Fig. 4. The input of the algorithm is a TD *D*, the parameters of our methodology (namely, a grid *G*, the dimensionality *p*, a control parameter σ – to be discussed in Section 4.1 – and thresholds Δv and Δd), and a number top*k* for the desired cardinality of the sampling set *S*; the output is the sampling set *S*. Although in the presented algorithm, top*k* is given as input, it is important to note that an 'optimal' value of it can be also recommended by the method itself (this interesting property will be discussed in Section 4.3).

The *T-Sampling* algorithm works as follows. First, trajectories are transformed into a low-dimensional symbolic space and their representatives (*ReTra*) are elected using a similarity-based voting technique (lines 1-2). Second, the maximal symbolic representative portions of trajectories (*SyTra*) are extracted (lines 3-5; line 5 is included only for clarity as its computation is incrementally performed in line 4). Third, sampling is achieved by a novel sampling algorithm (*TrajectorySampling*) in an unsupervised way (line 6).

```
Algorithm T-Sampling
(TD D, Grid G,int p, Real σ, Real Δv,
Real Δd, int topk)
     // STEP 1, see Sec. 4.1
 01. for i=1 to N T̄i=ApTra(G,p,Ti);
 02. for i=1 to N Ri=ReTra(T̄i,σ);
     // STEP 2, see Sec. 4.2
 03. for i=1 to N Si=SyTra(Ri,G,Δv,Δd);
 04. for i=1 to N Ri=ReTra(Si,σ);
 05. forall Ri compute Vi(D); //Eq.1
     // STEP 3, see Sec. 4.3
 06. S = TrajectorySampling(V(D),topk);
 07. return S;
```

Fig. 4. *T-Sampling* Algorithm

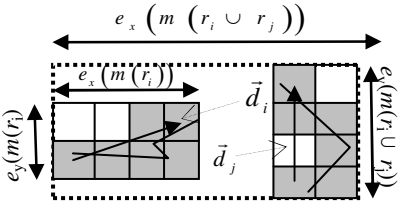


Fig. 5. $D_r(r_i, r_j)$ distance function

The three steps of the algorithm are described in the following sections.

4.1 Voting for Trajectory Representatives

As already discussed, in order to define the representativeness of a trajectory we measure distance D_{dr} between two directed regions. The key observation is that such a distance function can be decomposed in two parts, distance D_r between two regions and distance D_d between two motion vectors; then, we combine them into a single distance function using an aggregator, e.g. the average of the two components (for clarity we suppress the second subscript notating the trajectory id):

$$D_{dr}(dr_i, dr_j) = \left(D_d(\vec{d}_i, \vec{d}_j) + D_r(r_i, r_j)\right)/2 \quad (3)$$

Below we give the definition of the distance between two regions (i.e., sets of cells), taking also into account the time periods that they correspond to.

Definition 6. *Given two 3D regions r_i and r_j, their distance $D_r(r_i, r_j)$ is defined as follows:*

$$D_r(r_i, r_j) = 1 - \frac{1}{3}\left(\sum_{\dim \in \{x,y,t\}} \frac{e_{\dim}\left(m(r_i)\right) + e_{\dim}\left(m(r_j)\right)}{2 \cdot e_{\dim}\left(m\left(r_i \cup r_j\right)\right)} \right), \quad (4)$$

where e.g. $e_x(m(r_i))$ is the extent of the minimum bounding box (mbb) of r_i along the x- axis (similar for y- and t- axis). ■

Fig. 5 illustrates the 2D spatial projection of two regions (time is omitted for clarity), as well as the respective motion vectors. It is clear that $D_r(r_i, r_j)$ is bounded in [0,1]. Intuitively, $D_r(r_i, r_j)$ takes into account both the Euclidean distance between two regions and their extents, while it produces non-zero results in the case of overlapping (in space or time) but non identical regions.

Similarly, we give the definition of the distance between two motion vectors (i.e., directed segments) $\vec{d}_i$ and $\vec{d}_j$ as the minimum normalized energy of rotation of line segment $\vec{d}_i$ to $\vec{d}_j$, or vice-versa, which depends on the angle θ formed between the two directed segments.

Definition 7. *Given two motion vectors (directed segments) $\vec{d}_i$ and $\vec{d}_j$, their distance $D_d(\vec{d}_i, \vec{d}_j)$ is defined as the normalized distance, that a segment will cover, after its rotation of θ rads, $0 \le \theta \le \pi$, where θ is the angle between the two segments after they have been translated so as to have a common starting points. Formally:* ■

$$D_d(\vec{d}_i, \vec{d}_j) = \frac{\theta}{\pi} \quad (5)$$

According to the problem formulation presented in Section 3, the *representativeness* of a directed region dr_i depends on its distance to every trajectory in TD. We define distance $D_{dr}(dr_i, T_m)$ between dr_i and a trajectory T_m of the TD as the distance between two directed regions, namely dr_i and the closest directed region of T_m to dr_i. In the literature, several voting functions have been proposed, either step or continuous. In this work, we have selected to use the following continuous function of Gaussian kernel, which is widely used in a variety of applications in the field of pattern recognition [21].

$$v(dr_i, T_m) = e^{-\frac{D_{dr}^2(dr_i, T_m)}{2 \cdot \sigma^2}} \quad (6)$$

Since $0 \le D_{dr}(dr_i, T_m) \le 1$, the control parameter $0 \le \sigma \le ½$ tunes how fast the function (i.e., voting influence) decreases with distance. According to this definition, it holds that $0 \le v(dr_i, T_m) \le 1$. The lower the $D_{dr}(dr_i, T_m)$ the higher the value of the voting function and vice-versa.

Adopting a continuous voting function, like the Gaussian kernel, we get smooth results for slight changes on parameters (σ in our case), and getting decimal values as

results of the voting process increases the robustness of the method. Finally, $v(dr_i)$ is computed by summating the votes from all trajectories $T_{m.}$

$$v(dr_i) = \sum_{m=1..n, m \neq k}^{N} v(dr_i, T_m) \quad (7)$$

Lemma 1. The time complexity of the *Trajectory Voting* step of our methodology is $O(N \cdot p \cdot log(N \cdot p))$, where p is the dimensionality of the symbolic representation and N is the cardinality of D.

Proof: Assuming that L denotes the average number of directed regions per trajectory, the computational cost of representativeness $v(dr_i)$ is $O(N \cdot L)$. This is measured L times on the average for each of N trajectories in TD. As such, the total computation cost is $O(N^2 \cdot L^2)$. However, in order to be able to execute the algorithm in large databases, this computation cost may be reduced using a spatial index, such as the R-tree-like structures used in [6]. Using R-trees reduces the cost of voting for each dr_i to $O(log(N \cdot L))$, resulting in a total cost $O(N \cdot L \cdot log(N \cdot L))$. The voting method is applied twice: first, on the p-dimensional *ApTra* and, second, on the merged *SyTra*. In the first case, $L = p$ while in the second case $L << p$. Overall, the cost of the trajectory voting step is the sum of the two phases, hence $O(N \cdot p \cdot log(N \cdot p))$. ■

4.2 Global Symbolic Trajectories

As already discussed, a more intuitive representation for *ReTra* that certainly has added value for an analyst would be to provide an aggregate representation along the temporal dimension (i.e., *SyTra*). Fig. 6 presents an algorithm that transforms *ReTra* to *SyTra*, according to Definition 3. The main idea of the algorithm is, starting from an initial time period to follow a local time-based clustering approach in the form of a greedy time-expansion (merging) technique that concatenates a set of successive periods, as long as distance D_d of the directed segments is low enough to preserve the (local) mobility pattern (sketch) of the trajectory, but also the *representativeness* of the candidate (for merging) regions remains more or less the same. More specifically, after the initialization (lines 1-2), the algorithm starts an iterative procedure until all time periods are used (lines 3-15). At each iteration, it appends a local merged *ReTra* to the transformed *SyTra* (line 14). The new local *ReTra* is initialized with the corresponding old local *ReTra* of the first time period not used so far (line 4). Subsequently, using the previous region as seed, the merging (M) begins (line 5) by searching the best among the candidates periods (lines 6-8), which if concatenated to local *ReTra* the resulted region will be Δd-similar, while the relative difference in representativeness, before and after the merging is low (i.e., Δv-similar) (line 8). The best period to stop expansion is the one that minimizes the relative difference after merging (lines 9-12). The whole process continues until no more merging can be done (line 13).

Lemma 2. The time complexity of the *SyTra* algorithm is $O(p^2)$, where p is the dimensionality of the symbolic representation.

Proof: There are two loops that determine the complexity of the algorithm (lines 3 and 5). The loop (line 3) that revokes the merging process (line 5) may be repeated at most p times. This will happen if no actual merging occurs. This means that in this

```
Algorithm SyTra
(ReTra retra, Grid G, Real Δv, Real Δd)
01. sytra=∅; j=1; c=1;
02. forall i in [j,P] used(p_i)=false;
03. repeat
04.   retra_c = retra(p_j);
05.   repeat
06.     forall periods p_k, k in [j+1,p]
07.       M_j,k = retra_c expanded with
                  retra(p_k);
08.     M ={M_j,k | D_d(retra_c, M_j,k) < Δd and
                  |v(M_j,k)−v(retra_c)| / v(retra_c) < Δv };
09.     if M ≠ ∅
10.       retra_c= arg min_{M_j,k∈M} ( |v(M_j,k)−v(retra_c)| / v(retra_c) );
11.       forall i in [j,k] used(p_i)=true;
12.       j=k+1;
13.   until M ≠ ∅;
14.   sytra =sytra ∪ retra_c; c=c+1;
15. until used(p_p)==true;
16. return sytra;
```

Fig. 6. *SyTra* Algorithm

```
Algorithm TrajectorySampling
(Vector V(D),int topk)
01. for i=1 to N S_i=0;
02. V=sort_asceding(V);
03. for k=1 to topk
04.   SR_gain^max = -1;
05.   for i=1 to N
06.     if S_i == 0 AND
            SR_gain(i)>SR_gain^max
07.       SR_gain^max = SR_gain(i);
08.       id = i;
09.     end
10.     if SR_gain^max > V_i
11.       break;
12.     end
13.   end
14.   if  SR_gain^max <= 0
15.     topk = i-1;
16.     return S;
17.   end
18.   S_id=1;
19. end
20. return S;
```

Fig. 7. TrajectorySampling Algorithm

case the internal loop (line 5) will be revoked only once for each period (i.e., $M = \emptyset$) and, as such, its cost is only the cost of the loop through all (the remaining) time periods (lines 6-7), which is p; plus the cost of the filtering (line 8), which is also p, as the D_d calculation has constant cost and the aggregated calculations are computed incrementally. Consequently, in this case the complexity of the algorithm is $O(p(p+p))$, hence $O(p^2)$. In the other extreme case, the loop (line 3) that revokes the time expansion process (line 5) is revoked only once, implying that all periods are merged in one. In this case, the cost of the algorithm is dominated by the internal loop (line 5), which, in the worst case, will be revoked p times. Again the cost of each iteration is $O(p+p)$ as delineated in the previous paragraph, even if $M = \emptyset$ in this case, as the actual merging (line 10) can also be calculated incrementally for all iterations of the merging process. So, the overall complexity is again $O(p^2)$. ■

4.3 Trajectory Sampling Based on Representativeness

At the final step of our methodology, we provide a solution to the trajectory sampling problem defined in Section 3. In detail, the *TrajectorySampling* algorithm (illustrated in Fig. 7) takes the representativeness vector descriptor of the trajectories and returns the sampled subset. Recall that the goal of sampling is the maximization of the number of trajectories $SR(S)$ of the original TD that find their representatives in the sampling set (see Definition 5). The complexity of an exhaustive algorithm would search for all possible solutions in order to maximize Equation (2), resulting in a prohibitive cost $O\binom{N}{\text{top}k}$. On the other hand, the proposed algorithm suboptimally solves the problem in $O(N \bullet \text{top-}k)$ iterations adopting a greedy optimization approach.

In detail, the *TrajectorySampling* algorithm starts with an empty sampling set $S_i = 0$ for each $i = 1, \ldots, N$ (line 1). In each iteration, an unselected trajectory of D that maximizes Equation (2) is added in S, which is equivalent with the maximization of $SR_{gain}(i)$ that is defined as:

$$SR_{gain}(i) = V_i(D) \bullet (1 - V_i(S)) \tag{8}$$

$SR_{gain}(i)$ expresses the gain of $SR(S)$ if the i-th trajectory of D is added in S. According to the proposed algorithm, it holds that SR_{gain} is a monotonically decreasing function as sampling size increases, while it also holds that $SR_{gain}(i) \leq V_i(D)$. Therefore, an efficient way to reduce the computation cost of the maximization of $SR_{gain}(i)$, is to keep the trajectories in a list sorted in ascending order by $V_i(D)$ (line 2). Instead of computing $SR_{gain}(i)$ for each trajectory in D in order to find the maximum, we get the i-th trajectory from the list and we compare its *representativeness* with the current highest SR_{gain} (SR_{gain}^{max}) (lines 6-9). This loop terminates if SR_{gain}^{max} is higher than the *representativeness* of the i-th trajectory (line 10), because it holds that the SR_{gain} of each trajectory from the rest trajectories of the list would be lower than its *representativeness* and lower than SR_{gain}^{max}, since the contents of the list are in ascending order by trajectory *representativeness*. The algorithm terminates either when the size of sampling set reaches top$k$ (which is given as an input parameter); or when SR_{gain}^{max} becomes a non positive number, which means that $SR(S)$ has been maximized (lines 14-17), as the latter is increased in each step by SR_{gain}^{max}. In the second case, the algorithm has automatically found an 'optimal' top$k$, assuming it is lower than the given input.

An advantage of the proposed method is that it provides a deterministic solution in contrary with other probabilistic techniques [11], [16] that provide a randomly constructed sampling set trying to fit it to a desired distribution.

Lemma 3. Assuming top$k$ << N, the time complexity of the *TrajectorySampling* is $O(N \cdot log(L \cdot \text{top}k) + N \cdot logN)$, $\Omega(N \cdot log(N) + \text{top}k \cdot L \cdot log(L \cdot \text{top}k))$, where N is the number of trajectories in TD and L is the average number of directed regions per trajectory.

Proof: First, sorting $V(D)$ costs $O(N \cdot log(N))$. Next, in the worst case, the method computes $SR_{gain}(k)$ top$k \cdot N$ times. $SR_{gain}(k)$ computation requires the computation of $V_k(S)$. The cost for $V_k(S)$ is $O(L \cdot log(\text{top}k \cdot L))$ according to Lemma 1, since the maximum size of sampling set is top$k$. Therefore, in the worst case the cost of the method is $O(\text{top}k \cdot N \cdot log(L \cdot \text{top}k) + N \cdot log(N))$, which results in $O(N \cdot log(L \cdot \text{top}k) + N \cdot logN)$ if top$k$ << N. In the best case, the break of line 11 of the algorithm can stop the intrinsic loop in two ($O(1)$) instead of N steps ($O(N)$). In this case, the cost is $\Omega(N \cdot log(N) + \text{top}k \cdot L \cdot log(L \cdot \text{top}k))$. ■

5 Experimental Study

In this section, we present an experimental study in order to evaluate our approach over real and synthetic TD. In particular, we have used the *Athens trucks* real dataset (available at http://www.rtreeportal.org), which consists of 112,300 GPS-tracked positions from 50 trucks transporting concrete in the metropolitan area of Athens,

partitioned in 1100 trajectories. For further experimentation, we have also used synthetic datasets generated by a custom generator based on the popular GSTD [20]. Specifically, this generator produces trajectory datasets based on a given distribution of *spatio-temporal focal points*, to be visited by each trajectory in a specific order. The generated dataset then forms a natural cluster, since all trajectories follow more or less the same behavior. For example, Fig. 8(a) illustrates the 2D projection of a cluster generated with the above generator, using points 1 to 5 as focal points. The generator also allows adjusting the speed of each moving point (which may follow random or normal distribution), and also the temporal periods between sampled points (e.g., temporal gap between two sampled positions). Finally, by choosing focal points of varying distributions one may produce parts of a cluster with varying density (e.g. in Fig. 8(a) motions from focal point 1 to point 2 are sparser than those from point 4 to point 5). Using this methodology, we produced four synthetic clusters of trajectories, called C_1 to C_4, whose 3D visualization is presented in Fig. 8(b). Each cluster contains 50 trajectories, while each trajectory has 50 segments in average. It is obvious that the produced clusters were produced by mixing the focal points, and as such hiding the latent mobility patterns. Moreover, we produced three clusters, called C_2^{40}, C_3^{30}, C_4^{20}, by randomly removing trajectories from the original clusters. For example, C_2^{40} has been produced by removing 10 trajectories from C_2. Finally we concatenate clusters (i.e. C_1C_2, $C_1C_2^{40}$, $C_1C_2^{40}C_3^{30}$, $C_1C_2^{40}C_3^{30}C_4^{20}$) as such producing datasets having various numbers of clusters with diverse density.

(a) (b)

Fig. 8. Synthetic data: (a) cluster C_1 in 2D, (b) 4 clusters in 3D

The parameters introduced in our approach are: (a) those regarding the approximate representation of the trajectories (i.e., p and G), and which mainly affect the execution time of our approach as trajectories are initially transformed to vectors in various spatio-temporal granularities; (b) the control parameter $0 \leq \sigma \leq ½$ that in our experiments was set to 0.1, which corresponds to a smooth voting influence; (c) the thresholds Δv and Δd of *SyTra* algorithm that in our experiments were both set to 0.1, which corresponds to 10% relative difference to *representativeness* and 18° directional deviation, respectively; (d) the top$k$ parameter.

The experiments were run on a PC with Intel Core Duo at 2.53 GHz, 4 GB RAM and 240 GB hard disk. We implemented the proposed algorithms using C++.

5.1 Experimenting with Real Data

In this section, we evaluate the effectiveness and efficiency of our approach in the *Athens trucks* TD, which is visualized in Fig. 9(a) and where the majority of motions are around two dense areas (see the two ellipsoids in Fig. 9(a)).

In our first experiment, we sample the TD scaling the top*k* parameter, and we test the ability of our approach to capture the sketch of the whole TD by visualizing the sampled trajectories. Fig. 9(b) and (c) prove that our approach succeeds to preserve this sketch with various diverse granularities of approximation. The *cell size* is shown as percentage of the size of the total space.

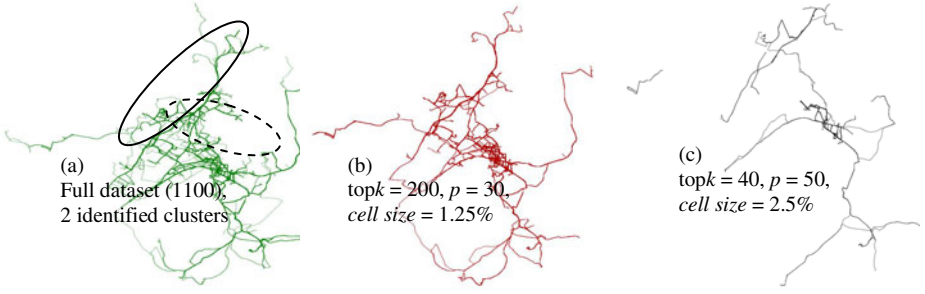


Fig. 9. Visualization of Athens trucks TD scaling top-k

Note that Fig. 9(b) with top*k* set to less than 20% of the TD cardinality captures almost completely the space, while with less than 4% Fig. 9(c) leads to the same comprehension (not only confining the main two patterns but also the behavior in the non-dense areas). Actually, this result is in accordance to our second experiment where we try to automatically identify the 'optimal' top*k* by scaling the size of the sampling size and computing the SR_{gain} which expresses the gain of including the *i*-th trajectory of the TD in the sampling set. Fig. 10(a) clearly depicts that a top*k* value between 35 and 40 (where SR_{gain} becomes very low) captures the most representative portions of the TD. Another conclusion is that the increase of dimensionality *p* slightly increases the 'optimal' top*k*, which is expected as the level of detail gets finer.

The second set of experiments is about the execution time of the various steps in our approach. Fig. 10(b) presents the accumulative processing time scaling the dimensionality *p*, and setting the cell size to 2.5%, and top*k* to 200 (i.e., we choose the maximum values used in the first experiment). In detail, we compute the processing time for each of the main steps presented in Fig. 4, namely the voting method applied in *ApTra* (line 2), the computation of *SyTra* (line 3), the re-application of the voting scheme to *SyTra* (line 4) and the actual sampling (line 6). The conclusions that can be drawn are: a) the processing time of every step is linear with dimensionality *p*; b) the computation of *SyTra* is extremely fast, negligible to the overall time, though extremely important as it relaxes the initially static dimensionality constraint discovering maximal mobility patterns; c) the processing time of *SyTra* voting is significantly lower than that of *ApTra* (due to merging), while both appear to be affordable (a few seconds are required even without index support). Fig. 10(c) presents

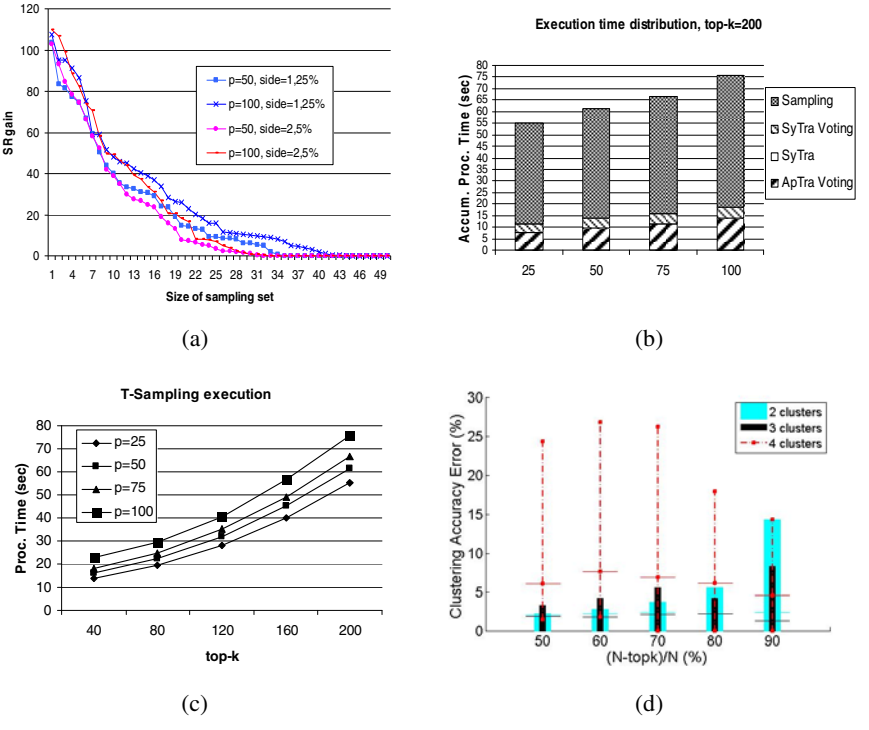


Fig. 10. (a) SRgain scaling the sampling set size, (b) and (c) T-Sampling processing time, (d) Clustering accuracy error vs. top*k*

the overall execution time for *T-Sampling*, where a smooth superlinear behavior of the algorithm appears, resulting in a few tens of seconds even for very large dimensionality and top*k* values.

5.2 Experimenting with Synthetic Data

The purpose of experimenting with synthetic data is to assess the ability of the proposed methodology to preserve patterns extracted by trajectory data mining algorithms. In our first experiment, we use a state-of-the-art trajectory clustering algorithm [5], and we measure the effect of sampling in clustering. We sampled the $C_1C_2^{40}$, $C_1C_2^{40}C_3^{30}$, $C_1C_2^{40}C_3^{30}C_4^{20}$ datasets including 2, 3, and 4 clusters respectively, scaling the top*k* from 50% to 10% of the cardinality of each dataset (i.e. the reduction after sampling was up to 90%). We measure the clustering accuracy error (i.e. the difference in the clustering accuracy before and after the sampling) for various granularity levels, ranging *p* from 10 to 50 (step 10), and *cell size* equal to 0.15%, 0.25%, 0.5%, 0.75% and 1% of total space. As such, for each dataset and for each top*k* we performed 5×5=25 experiments. Fig. 10(d) illustrates the range of values, the minimum, maximum, and average (i.e., the horizontal in-between crossing segment)

clustering accuracy error for each set of experiments, where it is clear that the error introduced in the clustering patterns is low (around 5%). We would like to note that the maximum error is introduced only in one out of 25 experiments, when $p = 50$ and *cell size* = 0.15%.

In the second experiment, we employ two algorithms that extract the so-called representative or centroid of a trajectory cluster, namely TRACLUS [12] and CenTra [19], respectively. TRACLUS identifies local, more or less linear clusters of segments of trajectories without using the temporal information, while CenTra discovers clusters of whole symbolic trajectories. As such, these two approaches are typical examples of two different clustering techniques having as output different mobility patterns. The idea of the experiment is to evaluate whether the two techniques capture more or less similar mobility patterns when applied before and after sampling. For this purpose, we use the C_1C_2 dataset from which we sample the top$k$ = 50% of the cardinality of each dataset. The visualization of the clusters and their mobility patterns in Fig. 11 clearly shows the resemblance of the resulted patterns, before and after sampling. We repeated the experiment several times tuning the parameters of these algorithms and with various sampling sizes and the conclusion remained the same.

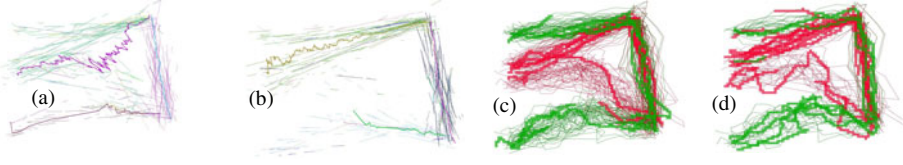


Fig. 11. TRACLUS [12] (a-b) and CenTra [19] (c-d) results before and after sampling

6 Conclusion and Future Work

In this work, we proposed a novel solution to the challenging problem of trajectory sampling, where the challenge is to build a sample by selecting *representatives* among a large set of trajectories in an unsupervised way for general purpose. To the best of our knowledge, there is no related work that addresses this problem apart from explorative, supervised by the user, approaches.

In the future, we plan to add a least enlargement criterion of the merged directed regions to the process of *SyTra* algorithm (as in [1]), so as to tight our implementation with an R-tree like access method for efficiency purposes. Moreover, we plan to evaluate our approach on other types of mobility patterns, such as T-patterns [7] and flock patterns [9], and we will try to adapt it for the purpose of outlier detection [13].

Acknowledgements. Research partially supported by the FP7 ICT/FET Project MODAP (Mobility, Data Mining, and Privacy) funded by the European Union. URL: www.modap.org.

References

1. Anagnostopoulos, A., Vlachos, M., Hadjieleftheriou, M., Keogh, E., Yu, P.S.: Global distance-based segmentation of trajectories. In: Proc. of KDD (2006)
2. Andrienko, G., Andrienko, N., Wrobel, S.: Visual Analytics Tools for Analysis of Movement Data. ACM SIGKDD Explorations 9(2) (2007)
3. Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D.: A visual analytics toolkit for cluster-based classification of mobility data. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) Advances in Spatial and Temporal Databases. LNCS, vol. 5644, pp. 432–435. Springer, Heidelberg (2009)
4. Andrienko, G., Andrienko, N., Rinzivillo, S., Nanni, M., Pedreschi, D., Giannotti, F.: Interactive visual clustering of large collections of trajectories. In: Proc of VAST (2009)
5. Gaffney, S., Smyth, P.: Trajectory Clustering with Mixtures of Regression Models. In: Proc. of SIGKDD (1999)
6. Frentzos, E., Gratsias, K., Pelekis, N., Theodoridis, Y.: Algorithms for Nearest Neighbor Search on Moving Object Trajectories. Geoinformatica 11, 159–193 (2007)
7. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F.: Trajectory Pattern Mining. In: Proc. of SIGKDD (2007)
8. Giannotti, F., Pedreschi, D.: Mobility, Data Mining and Privacy, Geographic Knowledge Discovery. Springer, Heidelberg (2008)
9. Gudmundsson, J., Kreveld, M.J., Speckmann, B.: Efficient detection of patterns in 2d trajectories of moving points. GeoInformatica 11(2), 195–215 (2007)
10. Kalnis, P., Mamoulis, N., Bakiras, S.: On discovering moving clusters in spatio-temporal data. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 364–381. Springer, Heidelberg (2005)
11. Kollios, G., Gunopulos, D., Koudas, N., Berchtold, S.: Efficient biased sampling for approximate clustering and outlier detection in large datasets. TKDE 15(5) (2003)
12. Lee, J.-G., Han, J., Whang, K.-Y.: Trajectory clustering: a partition-and-group framework. In: Proc. of SIGMOD (2007)
13. Lee, J.-G., Han, J., Li, X.: Trajectory Outlier Detection: A Partition-and-Detect Framework. In: Proc. of ICDE (2008)
14. Li, Y., Han, J., Yang, J.: Clustering moving objects. In: Proc. of KDD (2004)
15. Nanni, M., Pedreschi, D.: Time-focused clustering of trajectories of moving objects. Journal of Intelligent Information Systems 27(3) (2006)
16. Nanopoulos, A., Theodoridis, Y., Manolopoulos, Y.: Indexed-based density biased sampling for clustering applications. Data and Knowledge Engineering 57(1), 37–63 (2006)
17. Palmer, R., Faloutsos, C.: Density biased sampling: An improved method for data mining and clustering. In: Proc of SIGMOD (2000)
18. Panagiotakis, C., Pelekis, N., Kopanakis, I.: Trajectory voting and classification based on spatiotemporal similarity in moving object databases. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 131–142. Springer, Heidelberg (2009)
19. Pelekis, N., Kopanakis, I., Kotsifakos, E., Frentzos, E., Theodoridis, Y.: Clustering Trajectories of Moving Objects in an Uncertain World. In: Proc. of ICDM (2009)
20. Theodoridis, Y., Silva, J.R.O., Nascimento, M.A.: On the Generation of Spatiotemporal Datasets. In: Güting, R.H., Papadias, D., Lochovsky, F.H. (eds.) SSD 1999. LNCS, vol. 1651, p. 147. Springer, Heidelberg (1999)
21. Yuan, J., Bo, L., Wang, K., Yu, T.: Adaptive spherical gaussian kernel in sparse bayesian learning framework for nonlinear regression. Expert Syst. Appl. 36(2) (2009)