Noname manuscript No.
(will be inserted by the editor)

Time-focused clustering of trajectories of moving objects

Mirco Nanni · **Dino Pedreschi**

the date of receipt and acceptance should be inserted later

Abstract Spatio-temporal, geo-referenced datasets are growing rapidly, and will be more in the near future, due to both technological and social/commercial reasons. From the data mining viewpoint, spatio-temporal trajectory data introduce new dimensions and, correspondingly, novel issues in performing the analysis tasks. In this paper, we consider the clustering problem applied to the trajectory data domain. In particular, we propose an adaptation of a density-based clustering algorithm to trajectory data based on a simple notion of distance between trajectories. Then, a set of experiments on synthesized data is performed in order to test the algorithm and to compare it with other standard clustering approaches. Finally, a new approach to the trajectory clustering problem, called *temporal focussing*, is sketched, having the aim of exploiting the intrinsic semantics of the temporal dimension to improve the quality of trajectory clustering.

Note: The authors are members of the Pisa KDD Laboratory, a joint research initiative of ISTI-CNR and the University of Pisa: `http://www-kdd.isti.cnr.it`.

1 Introduction

Spatio-temporal, geo-referenced datasets are growing rapidly, and will be more in the near future. This phenomenon is due to the daily collection of transaction data through database systems, network traffic controllers, web servers, sensor networks. In prospect, an important source is telecommunication data from mobile phones and other location-aware devices – data that arise from the necessity of tracking such wireless, portable devices in order to support their interaction with the network infrastructure. But other than ordinary communication operations, the large availability of these forms of geo-referenced information is expected to enable novel classes of applications, where the discovery of knowledge is the

Mirco Nanni
ISTI - Institute of CNR, Via Moruzzi 1 – Loc. S. Cataldo, 56124 Pisa
Phone: +39 050 3152934, Fax: +39 050 3138091
E-mail: mirco.nanni@isti.cnr.it

Dino Pedreschi
Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti 2, 56127 Pisa, Italy
Phone: +39 050 2212752, Fax: +39 050 2212726
E-mail: pedre@di.unipi.it

key step. As a distinguishing example, the presence of a large number of location-aware, wireless mobile devices presents a growing possibility to access their tracking logs and reconstruct space-time trajectories of these personal devices and their human companions: trajectories are indeed the traces of moving objects and individuals. These mobile trajectories contain detailed information about personal and vehicular mobile behaviour, and therefore offer interesting practical opportunities to find behavioural patterns, to be used for instance in traffic and sustainable mobility management.

However, spatio-temporal data mining is still in its infancy, and even the most basic questions in this field are still largely unanswered: what kinds of patterns can be extracted from trajectories? Which methods and algorithms should be applied to extract them? One basic data mining method that could be applied to trajectories is *clustering*, i.e., the discovery of groups of *similar* trajectories.

Spatio-temporal trajectory data introduce new dimensions and, correspondingly, novel issues in performing the clustering task. Clustering moving object trajectories, for example, requires finding out both a proper spatial granularity level and significant temporal subdomains. Moreover, it is not obvious to identify the most promising approach to the clustering task among the many in the literature of data mining and statistics research; neither it is obvious to choose among the various options to represent a trajectory of a moving objects and to formalize the notion of (dis)similarity (or distance) among trajectories. A brief account of the research in this area is reported in Section 2.

In this context, we precisely address the problem of trajectory clustering. Our basic assumption on source data is that a collection of individual trajectories of moving objects can be reconstructed in an approximated way on the basis of tracking log data left by the objects as they move within the network infrastructure. As an example, a mobile phone that moves among the various cells of the wireless network leaves, during its interactions with the network, a set of triples (id, loc, t), each specifying the localization at space *loc* and at time *t* of the phone *id*. Starting from the set of triples for a given object *id* is therefore possible, in principle, to approximate a function $f_{id} : time \rightarrow space$, which assigns a location to object *id* for each moment in a given time interval. We call such a function a *trajectory*, and we concentrate on the problem of clustering a given set of trajectories. This basic assumption is consistent with the form of tracking log data that are (or can be) collected in the wireless network infrastructure; for the sake of concreteness, we could count in our research on the availability of a synthesizer of trajectory data, which has been developed at our laboratory (Giannotti, Mazzoni, Puntoni & Renso 2005) and has been used to create the source trajectory datasets employed in the empirical evaluation of the achieved results.

We address two distinct questions: (i) what is the most adequate clustering method for trajectories, and (ii) how can we exploit the intrinsic semantics of the temporal dimension to improve the quality of trajectory clustering.

Concerning the first problem, we advocate that *density-based clustering*, in the forms originally proposed in (Ester et al. 1996) and (Ankerst, Breunig, Kriegel & Sander 1999), is particularly well-suited to the purpose of trajectory clustering, given its distinctive features:

- the ability to construct non-spherical clusters of arbitrary shape, unlike the classical *k*-means and hierarchical methods,
- the robustness with respect to noise in the data,
- the ability of discovering an arbitrary number of clusters to better fit the source data – like hierarchical methods, but with a considerably lower complexity ($O(n \log n)$ vs. $O(n^2)$).

All the above are key issues for trajectories: it is likely that trajectories of cars in the urban traffic tend to agglomerate in snake-like, non-convex clusters, that many outlier trajectories should not be included in meaningful clusters but rather considered as noise, and that the number of clusters is unpredictable. Density-based clustering algorithms deal with the above problems by agglomerating objects within clusters on the basis of *density*, i.e., the amount of population within a given region in the space. In our approach, we generalize the spatial notion of distance between objects to a spatio-temporal notion of distance between trajectories, and we thus obtain a natural extension of the density-based clustering technique to trajectories. To analyze the consequences of our approach, we consider a particular density-based clustering algorithm, OPTICS (Ankerst et al. 1999), and propose an empirical comparison with several traditional *k*-means and hierarchical algorithms; we show how, on a set of experiments, our density-based approach succeeds in finding the natural clusters that are present in the source data, while all the other methods either fail or obtain less accurate results. To some extent, this sort of empirical evidence points out that density-based trajectory clustering yields a better quality output, with respect to the other traditional methods.

The second contribution of this paper is *temporal focusing*. Here, we stress that the temporal dimension plays a key role in trajectory clustering: two trajectories, which are very different considering the whole time interval of their duration, may become very similar if restricted considering a smaller sub-interval – an obvious observation with reference to, e.g., the vehicle trajectories in the urban traffic. It is therefore interesting to generalize trajectory clustering with a focus on the temporal dimension – basically enlarging the search space of interesting clusters by considering the restrictions of the source trajectories onto sub-intervals of time. Our proposed algorithm for temporal focussing is therefore aimed at searching the most meaningful time intervals, which allow to isolate the (density-based) clusters of higher quality.

The plan of the paper follows. In the next Section we briefly discuss related work, while in Section 3 we define trajectories and their distances. In Section 4 we revise density-based clustering and the OPTICS algorithm, while in Section 5 we propose the extension of OPTICS to trajectories and empirically evaluate our proposal. In Section 6 we propose the temporal focusing methods, together with some preliminary empirical experiments. Finally, in Section 7 we report the results of a larger experimentation aimed at assessing both the output quality and the performances of the proposed algorithm.

2 Related Work

In recent years, the problem of clustering spatio-temporal data received the attention of several researchers. Most of the actual work is focused on two kinds of spatio-temporal data: moving objects trajectories (the topic of this paper), such as traffic data, and geographically referenced events, such as epidemiological and geophysical data collected along several years.

Trajectory clustering. In one of the first works related to the topic, Ketterlin (Ketterlin 1997) considers generic sequences (thus modelling trajectories as sequences of points) together with a conceptual hierarchy over the sequence elements, used to compute both the cluster representatives and the distance between two sequences. Nanni, one of the authors of the present paper adapted two classical distance-based clustering methods (k-means and hierarchical agglomerative clustering) to trajectories (Nanni 2002). In the first part of the present work, we perform a step in the same direction, by adapting instead the density-based

approach to trajectories. An alternative strategy is to apply to trajectories some multidimensional scaling technique for non-vectorial data, e.g., Fastmap (Faloutsos & Lin 1995), which maps a given data space to an Euclidean space preserving (approximatively) the distances between objects, so that any standard clustering algorithm for vectorial data can be applied. Other distances can be inherited from the time-series domain, with the implicit assumption that the temporal component of data can be safely ignored and replaced by an order in the collected data values: the most common distance is the Euclidean metric, where each value of the series becomes a coordinate of a fixed-size vector; other approaches, which try to solve problems such as noise and time warping, include the computation of the longest common subsequence of two series (e.g., see (Vlachos, Gunopulos & Kollios 2002)), the count of common subsequences of length two (Agrawal, Lin, Sawhney & Shim 1995), the domain-dependent extraction of a single representative value for the whole series (Kalpakis, Gada & Puttagunta 2001), and so on. The main drawback of this transformational approach is ad-hoc nature, bound to specific applications. A thoroughly different method, proposed by Gaffney and Smyth (Gaffney & Smyth 1999), is model-based clustering for continuous trajectories, which groups together objects which are likely to be generated from a common core trajectory by adding Gaussian noise. In a successive work (Chudova, Gaffney, Mjolsness & Smyth 2003) spatial and (discrete) temporal shifting of trajectories within clusters is also considered.

Spatio-temporal density. The problem of finding densely populated regions in space-time is conceptually closely related to clustering, and it has been undertaken along several different directions. In (Hadjieleftheriou, Kollios, Gunopulos & Tsotras 2003), a system is proposed to support density queries over a database of uniform speed rectilinear trajectories, able to efficiently discover the spatial locations where moving objects are – or will be – dense. In (Gudmundsson, van Kreveld & Speckmann 2004) the computational complexity and approximation strategies for a few motion patterns are studied. In particular, the authors study *flock patterns*, that are defined as groups of at least n moving objects (n being a parameter) such that there exists a time interval of width larger than a given threshold where all such objects always lay inside a circle of given radius. A much similar objective is pursued in (Hwang, Liu, Chiu & Lim 2005), with an emphasis on efficiency issues. Finally, in (Li, Han & Yang 2004) an extension of *micro-clustering* to moving objects is proposed, which groups together rectilinear segments of trajectories that lay within a rectangle of given size in some time interval. The objectives of such work, however, are a bit different, being focused on the efficient computation of static clusters at variable time instants. The second contribution of this paper works in a direction similar to micro-clusters discovery. However, in addition to the more general concept of density adopted, in this paper we focus on the global clustering structure of the whole dataset, and not on small groups of objects.

Event clustering. A different view of the spatio-temporal clustering problem consists in considering spatially and temporally referenced events, instead of moving objects. In this case, the most basic approach consists in the application of spatial clustering algorithms where time becomes an additional spatial dimension. In addition to that, in literature different approaches have been proposed, mostly driven by specific application domains. Among them, we mention Kuldorff's spatial scan (Kulldorff 1997), a well known statistical method developed for epidemiological data which searches spatio-temporal cylinders (i.e., spatial circular shapes which remain still for some time interval) where the rate of disease cases is higher than outside the cylinder, and extensions for considering more flexible square pyramid shapes (Iyengar 2004).

3 A data model and distance for trajectories

In this paper we consider databases composed by a finite set of spatio-temporal objects. From an abstract point of view, a spatio-temporal object o is represented by a *trajectory* τ_o, i.e., a continuous function of time which, given a time instant t, returns the position at time t of the object in a d-dimensional space (typically $d \in \{2,3\}$). Formally: $\tau_o : \mathbb{R}^+ \rightarrow \mathbb{R}^d$.

In a real-world application, however, trajectories of objects are given by means of a finite set of observations, i.e. a finite subset of points taken from the real continuous trajectory. Moreover, it is reasonable to expect that observations are taken at irregular rates within each object, and that there is not any temporal alignment between the observations of different objects. This calls for an approximate reconstruction of the original trajectory. In this paper, we employ the model used in (Saltenis, Jensen, Leutenegger & Lopez 2000), where the objects are assumed to move in a piecewise linear manner. Namely, an object moves along a straight line with some constant speed till it changes the direction and/or speed. Such model essentially corresponds to the well known parametric 2-spaghetti approach (Chomicki & Revesz 1999).

In this work, we are interested in distances that describe the similarity of trajectories of objects along time and therefore are computed by analyzing the way the distance between the objects varies. More precisely, we restrict to consider only pairs of *contemporary* instantiations of objects, i.e., for each time instant we compare the positions of the objects at that moment, thus aggregating the set of distance values obtained this way. This implies, in particular, that we exclude subsequence matching and other similar operations usually adopted in the time series field, as well as solutions that try to align – in time and/or in space – shifted trajectories.

The distance between trajectories adopted in this paper is computed in a most natural way, as the average distance between objects, that is to say:

$$D(\tau_1,\tau_2)|_T = \frac{\int_T d(\tau_1(t),\tau_2(t))dt}{|T|},$$

where $d()$ is the Euclidean distance over $\mathcal{R}^2$, T is the temporal interval over which trajectories τ_1 and τ_2 exist, and $\tau_i(t)$ ($i \in \{1,2\}$) is the position of object τ_i at time t. We notice that such a definition requires a temporal domain common to all objects, which, in general, is not a hard requirement. From a conceptual viewpoint, moreover, in order to compute $D()$ we need to compute the infinite set of distances for each $t \in T$ (e.g., in and, afterward, to aggregate them. However, due to the (piece-wise) linearity of our trajectories, it can be shown (Nanni 2002) that $D()$ can be computed as a finite sum by means of $O(n_1+n_2)$ Euclidean distances, n_1 and n_2 being the number of observations respectively available for τ_1 and τ_2. Moreover, such distance is a metric, thus allowing the use of several indexing techniques that help to improve performances in several applications, including clustering.

4 Density-based clustering and OPTICS

In this section we briefly review the principles of density-based clustering, summarizing the motivations which led us to adopt this approach for trajectories, and describe the OPTICS algorithm.

4.1 Density-based Clustering

The key idea of density-based clustering algorithms is that for each object in some cluster the neighborhood of a given radius ε has to contain at least a minimum number n_{pts} of objects, i.e., the cardinality of the neighborhood has to exceed a given threshold. For that reason, such algorithms are naturally robust to problems such as noise and outliers, since they usually do not significantly affect the overall density distribution of data. This is an important feature for several real world applications, such as all those that work with data sources having some underlying random (unpredictable) component – e.g., data obtained by observing human behaviour – or that collect data by means of not-completely reliable methods – e.g., low-resolution sensors, sampled measures, etc. Trajectory data usually suffer of both the mentioned problems, so noise tolerance is a major requisite.

Moreover, in typical applications dealing with moving objects, such as traffic analysis or the study of PDAs usage and mobility, any strict constraint to the shape of clusters would be a strong limitation, so *k*-means and other spherical-shape clustering algorithms could not be generally applied (e.g., large groups of cars moving along the same road would form a "snake"-shaped cluster, not a spherical one).

The algorithms in the DBSCAN family (Ester et al. 1996) appear to be a good candidate choice for density-based clustering, since they satisfy the above mentioned requirements and, additionally, can be easily applied also to complex data at reasonable computational costs. In particular, in this paper we will focus on the OPTICS algorithm (Ankerst et al. 1999), a widely known evolution of the basic DBSCAN that solves many of its technical issues, first of all the sensitivity to input parameters. Moreover, it is worth noting that the output of OPTICS, the *reachability plot*, described in the following, is an intuitive, data-independent visualization of the cluster structure of data, that yields valuable information for a better comprehension of the data and that is (also) used to assign each object to its corresponding cluster or to noise, respectively.

4.2 OPTICS

In the following, we will shortly introduce the definitions underlying OPTICS, i.e., *core objects* and the *reachability-distance* of an object p w.r.t. a predecessor object o, and briefly describe how the algorithm works by means of a small example.

An object p is called a *core object* if the neighborhood around it is a dense region, and therefore p should definitely belong to some cluster and not to the noise. More formally:

Definition 1 (Core object) Let $p \in D$ be an object in dataset D, ε a distance threshold and $N_\varepsilon(p)$ the ε-neighborhood of p, i.e., the set of points $\{x \in D | d(p,x) \leq \varepsilon\}$. Then, given a parameter $n_{pts} \in \mathcal{N}$, p is a *core object* if: $|N_\varepsilon(p)| \geq n_{pts}$.

Based on core objects, we have the following:

Definition 2 (Reachability-distance) Let $p \in D$ be an object, $o \in D$ a core object, ε a distance threshold and $N_\varepsilon(o)$ the ε-neighborhood of o. Denoting with n-distance(p) the distance from p to its n-th neighbor in order of proximity ($n \in \mathcal{N}$), and given a parameter $n_{pts} \in \mathcal{N}$, the *reachability-distance of p with respect to o* is defined as

$$reach\text{–}d_{\varepsilon,n_{pts}}(p,o) = \max\{\ n_{pts}\text{–distance}(o)\ ,\ d(o,p)\ \}$$

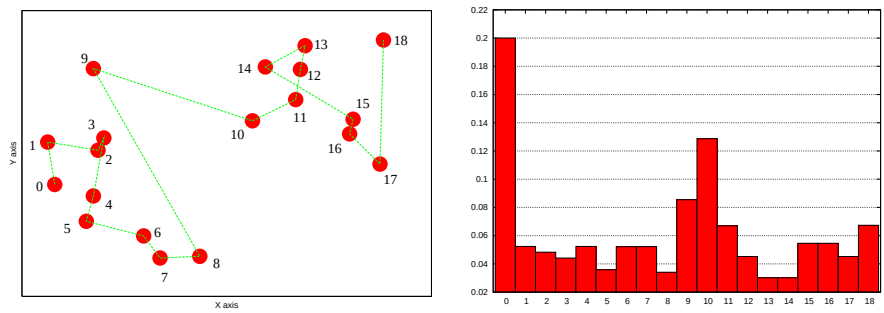


Fig. 1 Sample run of OPTICS with the resulting reachability plot

The reachability-distance of p w.r.t. o is essentially their distance, excepted when p is *too close*, in which case such distance is *normalized* to a suitable value. OPTICS works as follows: initially a random object p_0 is chosen; then, at each iteration i, the next object p_i chosen from D is that with the smallest reachability-distance w.r.t. all the already visited core objects; the process is repeated until all objects in D have been considered.

The whole process is summarized by means of a *reachability plot*: on the horizontal axis are represented the objects in their visit ordering $0,\ldots,|D|-1$, and on the vertical axis, for each i the reachability-distance corresponding to p_i is plotted. The sequence $\langle p_0,\ldots,p_{|D|-1}\rangle$ is also called a *cluster-reordering* of the objects in D.

Intuitively, the reachability-distance of a point p_i corresponds to the minimum distance from the set of its predecessors $p_j, 0 \leq j < i$. As a consequence, a high value of reachability-distance approximatively means a high distance from all other objects, and therefore indicates a *rarefied* area. Then, clusters, i.e., dense areas, are represented as valleys in the reachability plot. In particular, only core objects are considered among the predecessors, which results in not considering noise in the creation of clusters. Figure 1 shows a sample execution of OPTICS on a toy dataset with two clusters, and the corresponding values on the reachability plot. We observe that the jump from point 9 to point 10, belonging to different clusters, corresponds to a peak in the reachability plot.

From the reachability plot we can easily obtain a partitioning of the data into a set of clusters, plus noise: we simply need to choose a threshold to separate clusters, expressed in terms of the maximum value of the reachability distance allowed within clusters. Such value, denoted ε' and set by the user, is used to separate the objects into peaks and valleys: the former will be considered noise, the latter as clustered objects. In the example in Figure 1, setting $\varepsilon' = 0.1$, we would obtain two clusters: objects 1-9 and 11-18.

5 Extending OPTICS to trajectories

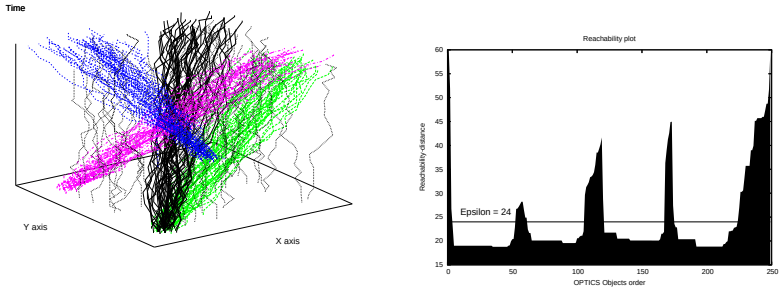
In order to define similarity measures for OPTICS over complex domains, its inventors proposed and tested a few solutions, classified into two classes: feature-based models and direct geometric models (Kriegel, Brecheisen, Kröger, Pfeifle & Schubert 2003). In the first case, the basic idea is to use a feature extraction function that maps the objects onto vectors in an appropriate multidimensional feature space. The similarity of two objects is then defined as their proximity in the feature space, and the closer their feature vectors are located, the more similar the objects are considered. In the second class of models, the distance is defined by directly using the geometry of objects. Examples include the computation of the

volume of the geometric intersection of the compared objects. In this paper, we followed an approach similar to the latter, since trajectories are compared and clustered by means of the spatio-temporal distance described in Section 3. In particular, our prototypes (both the Trajectory-OPTICS described in this section and the TF-OPTICS introduced in Section 6) have been implemented by integrating M-tree (Ciaccia, Patella & Zezula 1997), a index for generic metric spaces aimed at supporting efficient range queries, which are the core operation of the OPTICS algorithm. The results obtained in terms of performances will be discussed in detail in Section 7. However, it is less obvious to verify that OPTICS deliver high quality trajectory clusters. In the next section, we provide some very intuitive and preliminary evidence that OPTICS finds dense and well defined trajectory clusters, with a good tolerance to noise and outliers. A larger experimental assessment will presented later in the paper, in Section 7.

5.1 OPTICS vs. Hierarchical and K-means algorithms

We considered a synthetic test dataset randomly generated with the C4C trajectory generator – described in Section 7.1 – composed of 250 trajectories organized into four natural clusters plus noise. The dataset is depicted in Figure 2(a), where horizontal coordinates represent spatial positions and the vertical one represents time. Each cluster contains objects that move towards some defined direction. The objective of these experiments is to evaluate the behaviour of OPTICS on the trajectory data domain, as compared to other classical, general purpose clustering algorithms. We applied to the dataset the *k*-means algorithm and a standard hierarchical agglomerative algorithm in three versions: single-linkage, complete-linkage and average-linkage. The same dataset was processed by the trajectory version of OPTICS. All algorithms were configured to find four clusters (i.e., the expected number of natural clusters in the data). For the output of each algorithm, the average *purity* of the clusters (i.e., percentage of objects in the cluster that belonged to the corresponding real cluster) and the average of their *coverage* (i.e., percentage of objects of the real cluster that appear in the cluster found). The results are discussed below:

- K-means yields a 100% coverage but only a 53.7% purity. That is due to the fact that it merges together two clearly distinguished clusters, since it is sensible to noise and outliers. In fact, a whole resulting cluster is composed only of noisy objects.
- The hierarchical single-linkage algorithm yields a 100% coverage but only a 52.0% purity. Indeed, it is very sensitive to the chaining effect – i.e., the fact of collapsing far clusters due to a thin, yet continuous, line of objects linking them – and therefore it merges two dense and well separated clusters because of noise and the closeness of their borders.
- The complete-linkage algorithm yields a 99% coverage but only a 50.0% purity: it tends to form clusters with equal diameter, and so, due to the presence of noise that biases such size, two of the natural clusters are again merged together.
- The average-linkage algorithm yields a 100% coverage but only a 50.5% purity: it keeps clusters with balanced average intra-cluster distance, which usually results in a behavior similar to the complete-linkage case. Again, a pair of natural clusters is merged.
- Finally, Trajectory-OPTICS yields a 93.5% coverage and also a 99.0% purity, i.e., the best trade-off between the two measures. As we can see from the resulting reachability plot (Figure 2(b)), Trajectory-OPTICS correctly finds the four natural clusters, which can be easily isolated by selecting a proper value for the ε parameter ($\varepsilon = 24$ in our example, but any value that crosses the three central protrusions yield the same result).



(a) Input dataset (b) OPTICS Reach. plot

Fig. 2 A synthesized dataset (a) and the corresponding reachibility plot for OPTICS

6 Temporal Focusing

The approach to trajectory clustering presented above treats trajectories as unique, indivisible elements, and tries to group together those moving objects that globally move in a similar way, ”smoothing” the effect of any sporadic divergence in their movement. However, such global trajectory clustering may sometime be misleading and yield counter-intuitive results. In particular, it might keep separated objects that moved together for a significant amount of time, while gathering objects that constantly keep a significant distance between them.

From real world experience, we learnt that not all time intervals have the same importance. A meaningful example is urban traffic: in rush hours a large quantity of people move from home to work and viceversa, or, more generally, from/to largely shared targets. Therefore, we can expect that the sheer size of the population sample will make it possible for groups of individuals having similar destinations to clearly emerge from traffic data to form compact clusters. In quiet periods of the day, on the contrary, we expect to mainly observe static individuals, whose distribution on the territory is more driven by the geographical population density than by collective motion behaviors. This is a general problem not limited to urban traffic, and, while in this sample context some interesting hours of the day for cluster analysis can be guessed – e.g., typical morning rush hours –, in other, less understood cases and domains it might be not possible to fix a priori criteria for choosing the right period of time. In these cases, some automatic mechanism to discover the most interesting intervals of time should be applied. In what follows we formalize the problem mentioned above, and suggest a solution. We anticipate that we will consider only single intervals of time in our search problem, thus not taking into account more complex patterns of time, such as periodical patterns or irregular sets of disjoint time intervals.

6.1 Problem setting

As discussed above, there may exist time segments where the clustering structure of our moving objects dataset is clearer than just considering the whole trajectories. In order to discover such clustering structure, then, we should provide a method for locating the right time interval, and focus the clustering process on the segments of trajectories that lay in that interval, ignoring the remaining parts.

Our general approach consists in the following: the Trajectory-OPTICS algorithm is modified to compute distances between trajectories focusing on any time interval specified by the user; therefore, we (hypothetically) apply such algorithm to each possible time interval, evaluate each result obtained and determine the best one. This entails solving the following optimization problem:

$$\arg\max_{\theta} Q(D,\theta)$$

where D is the input dataset and Q is a quality function that measures the goodness of the clustering results obtained with the parameters θ. The set of parameters θ can contain, in a general setting, several different components, e.g.: basic parameters for OPTICS, a temporal window, a spatial granularity, etc. In this work we will focus on time windows, so $\theta = \langle\ \varepsilon, \varepsilon', n_{pts}, I\ \rangle$, ε, ε' and n_{pts} being the already mentioned general parameters for OPTICS and I being the time window we are focusing on. Moreover, since OPTICS is not very sensible to its input parameters n_{pts} and ε, we can assume that they are set before approaching the optimization problem, so that the only variables of the problem remain I and ε'.

We observe that the problem introduced above can be seen as a subspace clustering problem. In particular, if we assume that time is discrete, and therefore the time interval I can be reduced to a finite sequence of N time points, trajectories can be seen as $2N$-dimensional objects, and our objective is to discover $2N'$ contiguous dimensions ($N' \leq N$) that optimize our quality function. However, in all works of the subspace clustering literature, the dimensions are not related to each other by any specific semantics, differently from the temporal semantics that underlies trajectories. In particular, distances that are additive w.r.t. dimensions[1] are usually applied, making density of regions a monotonic function w.r.t. the dimensions (i.e., dense regions on some N-dimensional space always correspond to dense regions on any of its $N-1$-dimensional subspaces). On the contrary, when dealing with trajectories we have to deal with a semantics of time, that significantly modifies the problem. The most direct and relevant effects are the following: (i) a notion of contiguity is defined, that has to be preserved when selecting subspaces; and (ii) a distance between objects is given, that is not additive w.r.t. dimensions. As a consequence, actual subspace clustering techniques are not applicable in our case. In what follows, we propose an heuristic solution to this new variant of the subspace clustering problem.

6.2 Quality measure

The first issue to solve, now, is the definition of a quality function Q, which can provide a good criterion for deciding whether a clustering result is better than another. Any suitable definition should take into account the nature of clusters we obtain. In particular, since we are working with density-based tools, typical dispersion measures cannot be applied, because we can have good non-spherical clusters, which, in general, are not compact.

In the density-based setting, the standard *high intra-cluster vs. low inter-cluster similarity* principle could be naturally translated into a *high-density clusters vs. low-density noise* rule. Highly dense clusters can be considered interesting per se, while having a rarefied noise means that clusters are clearly separated. Put together, these two qualities seem to reasonably qualify a *good* (density-based) clustering result.

[1] I.e., distances between N-dimensional objects can be written as a sum of N contributions independently computed on each dimension. E.g., it holds for Euclidean distances, since $d^2(a,b) = \sum_{i=1}^{N}(a_i - b_1)^2$.

The reachability plot returned by the OPTICS algorithm contains a summary of the information on data density we need. Therefore, we can simplify the computation of the Q measure by deriving it from the corresponding reachability plot, since density at each point can be estimated by the corresponding reachability-distance.

Definition 3 Let D be an input dataset of trajectories, I a time interval and ε' a density threshold parameter. Then, the *average reachability*, $R(D,I,\varepsilon')$, is defined as the average reachability-distance of non-noise objects: $R(D,I,\varepsilon') = avg\{r | \langle p_0,\ldots,p_{|D|-1}\rangle$ is OPTICS cluster reordering $\wedge\, r = reach\text{–}d_{\varepsilon,n_{pts}}(p_i) \wedge r \leq \varepsilon'\}$. When clear from the context, average reachability will be denoted as R_C (reachability of clustered objects). When no ε'-cut is specified (i.e., $\varepsilon' = \infty$), average reachability will also be denoted as R_G (global reachability).

In this work, in particular, we will give the greatest importance to high-density clusters, which seem to be the most essential property between the two listed above, and will not take into consideration noise. For the latter, we will consider as sufficient the minimum density requirement specified by the user through the ε' parameter – noise, by definition, cannot be denser than such threshold. As mentioned above, density within clusters can be easily estimated by means of in-cluster reachabilities, leading to the following simple formulation:

Definition 4 Let D be an input dataset of trajectories, I a time interval and ε' a density threshold parameter. Then, we define the Q_1 quality measure as follows:

$$Q_1(D,I,\varepsilon') = -R(D,I,\varepsilon').$$

Dispersed clusters yield high reachability distances, and therefore highly negative values of Q_1, while compact clusters yield values of Q_1 closer to zero. Moreover, notice that Q_1, being an average value and not a simple sum, does not force any preference for small or large clusters.

6.3 Self-tuning of parameters

The search for an optimal time interval requires to iteratively analyze different views of the input trajectory dataset. However, each view can result in a quite different reachability plot, with peaks and valleys of highly variable size. As a consequence, it is not possible to determine a single value of the ε' that is valid for all time intervals. Since it is not reasonable to ask the user to choose a suitable value for each interval analyzed, we should design an automated method for the purpose. We describe a solution to the problem that requires the user to specify an initial ε'_0 value for the largest time interval, and then automatically adapts such value to the smaller intervals.

In general, the optimal ε' value for an interval I depends on several variables. We consider a simple solution and assume that densities follow some general trend, so that a global rescaling factor for density values can be computed. Such rescaling can be applied to ε'_0 to obtain the actual ε'. The overall density for a given time interval can be estimated in several ways, the simplest being the average reachability-distance R_G. Adopting such measure and a linear rescaling, we have the following value for ε':

$$\varepsilon'(D,I,\varepsilon'_0) = \frac{R_G}{R_G^0}\varepsilon'_0$$

where R_G^0 denotes the value of R_G on the largest time interval. In order to test the reliability of the ε' re-scaling method proposed above, we performed an experiment on the

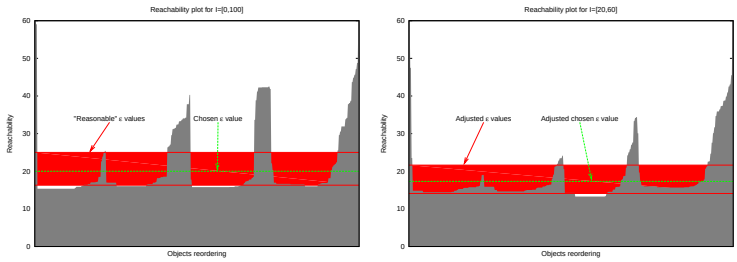


Fig. 3 Example of ε adaptation from $I = [0, 100]$ to $I = [20, 60]$

trajectory dataset considered in Section 5.1. We assumed that for each reachability plot it is possible to define an interval for the acceptable values of the ε' parameter, and that the specific ε' value that a generic user would choose follows a random uniform distribution over the above mentioned interval. Figure 3 (left) depicts an example of reasonable values over the largest time interval $(0-100)$, represented by a dark band, with an example of chosen value for ε. The same Figure (right) shows also their mapping to a smaller interval $(20-60)$.

In our experiment, We considered 50 samples that include time intervals of all sizes and of variable position, i.e., not biased towards initial or final intervals. Then, for each time interval I considered, we compute the probability that a value chosen on the $0-100$ time interval is *adjusted* to a reasonable value on I, i.e., a value that lays within corresponding interval of reasonable values. Then, we compute the average probability of success, over all cases. The result is the following: given a random interval I and an ε value randomly chosen on the reachability plot of the $0-100$ time interval, in the 80.5% of the cases such ε is mapped on I to a reasonable value. Such estimate shows that, in spite of its simplicity, the proposed rescaling method is empirically reliable.

6.4 Evaluating and improving Q_1

In order to accept Q_1 as a suitable quality measure, an important property to verify is its *non-shrinking* behavior w.r.t. interval size, i.e., the fact that not all time intervals I contain at least a subinterval I' having a higher quality measure. Formally, we would like to refute the following property: $\forall I.\exists I' \subset I : Q_1(I') > Q_1(I)$. In fact, such property would imply that the optimal time interval for Q_1 is always a minimal-size interval, thus trivializing the problem and yielding uninteresting results. Fortunately, it is easy to find a counterexample:

Example 1 Let consider a set of trajectories over a time interval that is formed by exactly two time units: in the first unit trajectories are very dense, while in the second one they become more rarefied, although not beyond the given density threshold. Now add a trajectory that is always distant from all the others, but in the first time interval is close enough to them to be part of the same cluster, while in the second one immediately gets extremely far from them. Then, focusing on the first interval, all trajectories belong to the same cluster but, due to the single outlier trajectory described above, it has only a medium density. In the second interval, the outlier trajectory becomes noise, but the remaining ones are a bit rarefied, so density is not high here. On the overall interval, we get all the positive aspects of both the sub-intervals: the outlier trajectory is considered as noise, and so the first sub-interval yields a very high density that is able to lift the overall density beyond the limits of the two sub-intervals. As a consequence, the larger interval has a better quality than its sub-intervals.

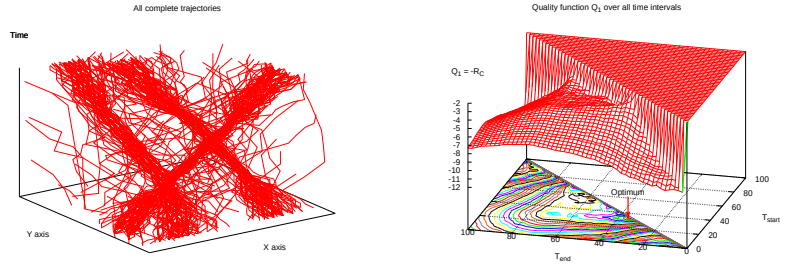


Fig. 4 Sample dataset and corresponding Q_1 plot

The second step in validating the quality measure is a test against a dataset. In Figure 4, on the left, a set of 320 trajectories is shown. Such dataset was generated by using the already mentioned C4C system, and contains (in addition to noise) three groups of trajectories that tend to move together within a central time interval, and tend to spread across a wider area in the other intervals. Trajectories are defined over the $[0,100]$ time interval, that is discretized into 50 time units of 2 seconds each. In figure 4, on the right, a plot of the Q_1 measure over all time intervals is depicted: the two horizontal axes represent the boundaries of time intervals, so that only the lower-left part of the plane – where the lower bounds of intervals are smaller than upper bounds – is significant. Below the 3D plot, the corresponding (plane) contour plot gives a bi-dimensional view of the same value, with the optimal interval pointed by an arrow. The plot shows that the optimum is located along the diagonal, which represents the intervals with the smallest size. In general, there is a strong bias towards small intervals, even though we can notice the presence of several central regions in the plot – corresponding to larger intervals – with Q_1 values close to the optimum.

When only very small variations of Q_1 are observed, it is reasonable to prefer larger intervals, since they are more informative. We can do that by slightly promoting larger intervals directly in the quality measure. A simple solution consists in adding a factor that increases the quality function as the interval size grows, but only very slowly, in order to avoid excessive bias. To this purpose, we adopt the following variation of the Q_1 measure:

$$Q_2(D,I,\varepsilon') = Q_1(D,I,\varepsilon')/\log_{10}(10+|I|)$$

In Figure 5 we can see the plot corresponding to the new Q_2 measure. As we can notice, the optimum value (pointed by the arrow) is now in the central region, on the $[30,72]$ interval, which means that the small correction introduced is sufficient to discover significantly large dense intervals. On the righthand plot we see the resulting clusters, where the segments of the clustered trajectories contained in the optimal time interval are emphasized: the densest segments of the natural clusters present in the data are clearly discovered, leaving out the portions of trajectory where the objects start/end dispersing.

6.5 Searching strategies

The basic method for finding the time interval that maximizes Q_2 is an exhaustive search over all possible intervals. Such approach is obviously very expensive, since it adds to the complexity of OPTICS a quadratic factor w.r.t. the the maximum size of time intervals, expressed in time units. More precisely, given a granularity of time τ, the global interval

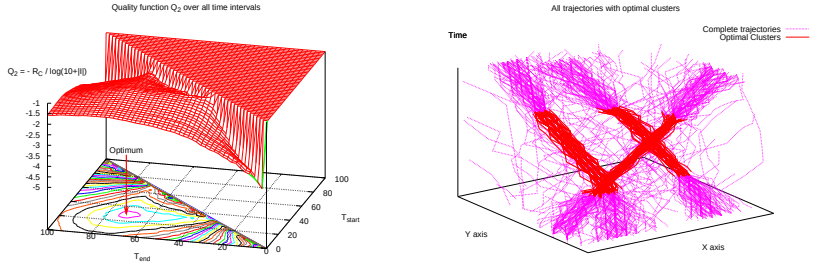


Fig. 5 Q_2 plot on sample dataset and corresponding output clusters

I_0 is divided into $N_t = |I_0|/\tau$ time units. Then, the cost of an exhaustive search is $O(N_t^2 \cdot OPTICS(n)) = O(N_t^2 n \log n)$.

A natural alternative to the exhaustive search of a global optimum is the search of local one, adopting some greedy search paradigm. We considered a standard hill climbing approach, where a randomly chosen solution (i.e., time interval) is iteratively modified, trying to improve the quality function at each step. We follow the procedure described below, where τ is the chosen temporal granularity and I_0 the largest interval:

1. Choose an initial random time interval $I \subseteq I_0$;
2. Let $I' = \arg\max_{T \in Neigh_I} Q_2(D, T, \varepsilon')$, where $Neigh_I = \{T \in \{[T_s \pm \tau, T_e], [T_s, T_e \pm \tau]\} | T \subseteq I_0\}$ and $I = [T_s, T_e]$;
3. If $Q_2(D, I', \varepsilon') > Q_2(D, I, \varepsilon')$ then let $I := I'$ and return to step 2; otherwise stop.

The total cost of the search procedure is $O(n_{iter} n \log n)$, where the number of iterations n_{iter} has a worst-case value of $O(N_t^2)$, but usually assumes much smaller values, linear in N_t. Executing the greedy search over our sample dataset from all possible starting points, we obtained the exact global optimum in the 70.7% of cases, which provides the success probability when seed points in the search strategy are chosen in a uniformly random way. Therefore, on this sample data we reach a high success probability with a small number of trials (with 5 runs we have over the 99.8% probability of finding the global optimum on this dataset). On the average, each run required around 17 steps and 49 OPTICS invocations – corresponding to less than 4% of the invocations required by the exhaustive search – thus keeping computational costs within reasonable limits.

7 Experiments

In this section we present a set of experiments aimed at assessing efficiency and effectiveness of the Trajectory-OPTICS system (T-OPTICS) and its time-focused version (TF-OPTICS). The datasets used in experiments have been synthetically generated by the C4C algorithm, an *ad hoc* data generator for trajectories with clusters briefly described below.

7.1 The C4C synthetic data generator

The C4C generator is a new, revised version of CENTRE (Giannotti et al. 2005). CENTRE is a general purpose trajectory generator able to simulate a large variety of movements be-

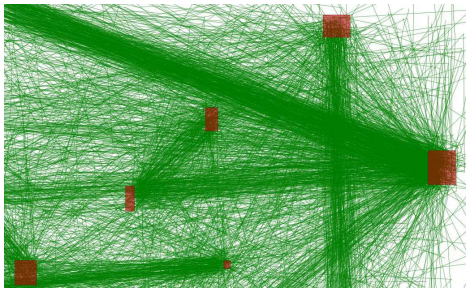


Fig. 6 Sample C4C dataset (spatial projection)

haviors, and the generation process is driven by several user defined parameters that act on speed, direction, agility and so on.

C4C (CENTRE for Clustering) is a generator of spatio-temporal objects that evolve in space and time producing a sequence of samples (i.e., spatial locations and their corresponding observation times) called trajectories. The main idea is to generate trajectories that follow some pre-defined clusters in order to stress and probe the limits of a clustering algorithm.

The core concept of C4C is to model a cluster with a sequence of rectangles (zones of attraction) placed within the workspace of the generator: one object assigned to one cluster must reach and cross the first zone of the sequence and subsequently all the others, respecting some temporal constraints defined by the user. On one hand, the temporal constraints define *when* an object belonging to a cluster starts to follow its corresponding rectangles, i.e., the typical behavior that characterizes the objects in the cluster. On the other hand, the density of clusters directly depends on the area of the rectangles in the sequence: objects forced to pass through small rectangles will produce denser groups of trajectories than others that can pass through larger rectangles. More in detail, each cluster is defined as a set of m sub-clusters $\{SubC_1, \ldots, SubC_m\}$ ($m > 0$). A sub-cluster $SubC$ is defined as a single step of the generation, and is characterized by four values: $SubC = \langle StartPos, EndPos, StartTime, EndTime \rangle$, where $StartPos$ represents the rectangle where the object starts evolving and $EndPos$ the one where it stops. $StartTime$ and $EndTime$ describe the time restrictions, so that the object has to start from a point in $StartPos$ at time $StartTime$ and reach a point in $EndPos$ at time $EndTime$. Finally, the sequence of sub-clusters that defines a cluster must be sorted and continuous, i.e., times between adjacent sub-clusters has to be non decreasing ($EndTime(i) < StartTime(i+1)$, for $0 < i < m$) and a sub-cluster has to start from the destination rectangle of the previous one ($StartPos(i+1) = EndPos(i)$, for $0 < i < m$).

Figure 6 shows a (spatial projection view of a) sample dataset generated by C4C, with the corresponding rectangles used in the generation. Trajectories that follow a cluster have to move from a randomly chosen point in a rectangle to another one in the destination rectangle. As we can see in the figure, there are rectangles of different sizes that we can use to define clusters of different densities: small rectangles will produce close trajectories, and therefore dense clusters. Moreover, a subset of trajectories do not follow any cluster, generating simple noise. In particular, the figure shows an instance of 1000 trajectories with 5 clusters with slightly different densities.

The generation of trajectories and clusters is driven by two sets of input parameters. The first set contains numerical parameters with some general information like random dis-

Dataset	n.cluster	density	type	noise
A	5	high	simple	no
B	5	high	complex	yes
C	3	mixed	simple	no
D	8	medium	complex	no
E	4	mixed	simple	no
F	3	mixed	simple	yes

Table 1 Datasets in brief

tributions settings used to control random movements of the objects, number of objects to generate, definition of clusters (times restrictions and rectangles they have to visit), etc. The second sets of inputs provides spatial information about the rectangles used to define clusters, i.e., their position on workspace and their size. In particular, C4C can import such information through GML documents (an XML format with geographical extensions), so these inputs can be generated by means of any GIS platform that supports it – e.g., the JUMP environment, used to generate the datasets for our experiments.

The generation process of a single spatio-temporal object takes the following steps:

- Initial positioning: the object is randomly positioned within the workspace.
- Move to first sub-cluster *StartPos*: once the object has been placed in the workspace, the generator assigns it to a cluster, and initializes its direction and speed in such a way that the object is able to reach a point of rectangle *StartPos* of the first sub-cluster on time (i.e., not later than *StartTime*).
- Evolution of a sub-cluster: the object moves towards its current destination *EndPos*, possibly performing some deviations that introduce noise in process, and adjusting direction and speed along the generation in order to reach the destination rectangle in time.
- End of cluster-like movement: when the object completes all the sub-clusters it was assigned to, it chooses a final destination (inside the workspace, again) and moves towards it (with possible perturbation) until the generation process is stopped.

7.1.1 Datasets description

In order to test our clustering algorithms, we used C4C to generate 6 datasets containing around 10000 trajectories, each composed of a variable number of observations, ranging from 10 to 30. Each dataset has different characteristics, in order to stress the algorithm by simulating different scenarios, so for each one we chose a different number and positions of the rectangles, different sizes and different levels of noise. Table 1 summarizes the characteristics of the datasets: the number of clusters, their density (high for clusters formed by small rectangles, and so on, while mixed stands for a mix of clusters in the same dataset having different densities), their type (simple if clusters contain only one sub-cluster, complex otherwise) and presence of noise.

7.2 Performance evaluation

7.2.1 Trajectory-OPTICS.

In order to measure the scalability of the T-OPTICS algorithm (without time-focusing) we performed several runs of the system over input datasets of growing size. Such input data

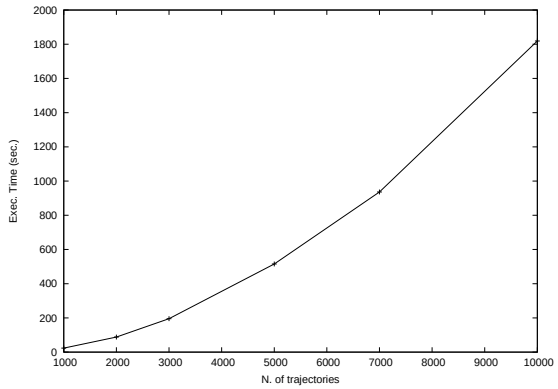


Fig. 7 Average performances of T-OPTICS

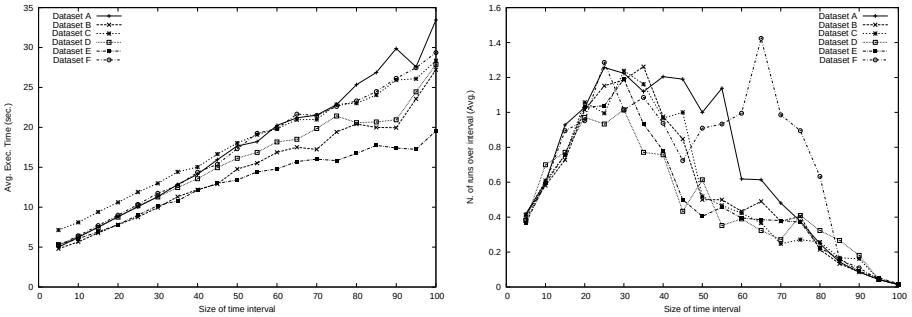
has been obtained as samples of the datasets generated through the C4C system, described in the previous section. A summary of the (average) execution times obtained is depicted in Figure 7.

As the graph shows, the curve apparently grows slightly more than linearly, which is the expected general behavior of OPTICS over any data type having an efficient indexing support. However, a detailed analysis of execution times reveals that (i) the greatest part of the computational cost of the algorithm is always due to the computation of distances; and (ii) the number of distances required to build the M-tree index and to answer the needed range queries apparently grows quadratically, even though with small constants (that make the phenomenon less visible in Figure 7), i.e., the M-tree seems to be not as efficient on trajectories with our (rather complex) distance function as it has been shown to be on different metric spaces (Ciaccia et al. 1997). However, as it will be shown in the rest of this section, working on time sub-intervals of trajectories makes the computational cost of the distance function smaller, making each step of the TF-OPTICS algorithm comparatively much cheaper than an execution of T-OPTICS.

7.2.2 Time-focused Trajectory-OPTICS.

As described in Section 3, the cost of computing distances between trajectories is linear w.r.t. their complexity, i.e., the number of observations they contain. On the other hand, the TF-OPTICS algorithm essentially consists in the reiterated execution of T-OPTICS on segments of trajectories, obtained by properly *clipping* the original ones. The clipping process in general reduces the complexity of trajectories, and therefore the cost of computing the distances between them. Figure 8(a) briefly reports the results of an empirical verification of such a behavior, showing the relation between the (average) execution time of a single step of TF-OPTICS (vertical axis) and the size of the time interval explored in that step (horizontal axis): on all datasets we obtain a linear or quasi-linear scale-up, confirming the intuition that, on average, the complexity of trajectories grows linearly w.r.t. their time extension. Notice: in this and in all the following experiments, time of trajectories ranges from 0 to 100, as in the examples already considered in this paper, and in the time-focusing processes it is discretized into 20 segments of length 5.

Based on the results shown above, a general (though empirical) estimation of the impact that the clipping process can produce on the execution times of TF-OPTICS can be obtained



(a) Cost of TF-OPTICS steps vs. interval size (b) Frequency of interval sizes

Fig. 8 Computational cost and frequency of interval sizes in TF-OPTICS

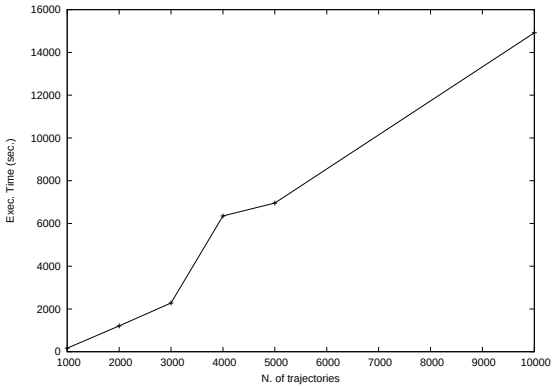


Fig. 9 Average performances of TF-OPTICS

by measuring how often large intervals are explored in a generic execution of TF-OPTICS, i.e., how many of the iterations of TF-OPTICS work on large intervals and how many work on small ones. That is done in Figure 8(b), that plots for each dataset the number of intervals of any given size that is analyzed by an execution of TF-OPTICS, averaged on all the possible paths followed by the search strategy – each path corresponding to a different random starting point. As we can see, the most frequently analyzed interval sizes range between 20 and 40, i.e., medium- and small-sized intervals, while large intervals are seldom visited, and therefore each step of TF-OPTICS is expected to be performed considerably faster than the basic T-OPTICS. Finally, Figure 9 reports the execution time of a whole execution of TF-OPTICS over different dataset sizes, averaged over the same datasets used for Figure 7: as expected, we obtain a scale-up similar to that of T-OPTICS, although with a less regular shape, where execution times are multiplied by the number of iterations of the algorithm but are also mitigated by the clipping phenomenon studied above.

7.3 Output evaluation

The output results given by T-OPTICS and TF-OPTICS can be evaluated, as done in Section 5.1 on a toy dataset, in terms of purity and coverage of the clusters found w.r.t. the real

	K-means		HC-avg		T-OPTICS		TF-OPTICS	
Dataset	*purity*	*coverage*	*p*	*c*	*p*	*c*	*p*	*c*
A	0.798	0.8	0.761	0.74	1	0.33	1	0.449
B	0.769	0.749	0.606	0.608	0.661	0.222	0.694	0.283
C	0.997	0.997	1	1	1	0.365	1	0.365
D	0.807	0.83	0.992	0.992	0.952	0.325	0.967	0.375
E	0.626	0.626	0.624	0.743	0.878	0.23	0.992	0.397
F	0.711	0.653	0.334	0.321	0.746	0.301	0.975	0.347

Table 2 Comparison of T-OPTICS and TF-OPTICS output results

Dataset	P(success)	OPTICS runs	Q_2 error
A	9.05%	5.40%	30.95%
B	2.86%	4.53%	48.46%
C	2.38%	4.69%	93.49%
D	3.81%	4.28%	31.03%
E	5.24%	5.40%	34.70%
F	4.29%	4.49%	22.00%

Table 3 Results of the search heuristic

clusters in the datasets. A summary of such measures is given in Table 2, where they are compared with analogous values obtained by applying a K-means and an average-based hierarchical approach (HC-avg, in short).

We notice in particular that: (i) in most of the cases T-OPTICS yields a better purity of K-means and HC-avg, and TF-OPTICS further improves such value; (ii) the coverage of T-OPTICS and TF-OPTICS is always much smaller than the other approaches. These results can be explained by noting that the clusters contained in the datasets are affected by a certain amount of dispersion (as opposed to the extremely compact trajectories we had in the toy example used in Section 5.1 and shown in Figure 2) that introduces a significant number of borderline trajectories, i.e., trajectories that belong to some cluster but not to its *core* group of members. Therefore, the density-based approach focuses on the core subset of elements of the clusters, identifying the ambiguous trajectories and labelling them as noise – a behavior that fits quite well with the context, objectives and general requirements discussed in the introduction – while the other approaches always assign them to some cluster – sometimes successfully (e.g., on dataset C) sometimes not (e.g., HC-avg on dataset F). T-OPTICS and TF-OPTICS consistently discover high-purity, non-ambiguous clusters, yielding compact sets of trajectories (i.e., clusters) that, where needed, can be more easily summarized through single/sets of representative trajectories or other concise forms of summarization.

Finally, we evaluate the effectiveness of the hill-climbing search strategy on our six datasets. In Table 3 we summarize the results in terms of success probability (probability of reaching global optimum), average number of T-OPTICS executions required (percentage w.r.t. the exhaustive search) and average error of the local optimum w.r.t. the global one.

As we can see, the probabilities of finding the global optima are not high, but, at the same time, on average the local optimum found has usually a quality value relatively close to the global optimum: we remark, in fact, that the Q_2 measure in our datasets has a high variability that leads to have a high average error w.r.t. the optimum and, therefore, the errors shown in Table 3 result to be comparatively small. Finally, we remark that the greedy search of TF-OPTICS required on average a small number of T-OPTICS executions, always between 4% and 6% of that required by an exhaustive search.

8 Conclusions

In this paper we developed a density-based clustering method for moving objects trajectories, aimed at properly exploiting the intrinsic temporal semantics to the purpose of discovering interesting time intervals, where (when...) the quality of the achieved clustering is optimal. Future research includes, on one hand, a refinement of the general method, the adoption of several different distance measures between trajectories (in addition to the one introduced in Section 3), and a vast empirical evaluation over various real-life datasets, mainly aimed at consolidating (and possibly correcting) the preliminary results shown in this work, and, on the other hand, a deeper integration between the underlying clustering engine and the search method.

Acknowledgements The authors are grateful to the members of the Pisa KDD Lab that helped to tune and test the prototypes, and in particular to Andrea Mazzoni and Simone Puntoni for their great help in setting up the synthetic data generator and integrating the M-tree index in TF-OPTICS.

References

Agrawal, R., Lin, K.-I., Sawhney, H. S. & Shim, K. (1995), Fast similarity search in the presence of noise, scaling, and translation in time-series databases., *in* 'VLDB', pp. 490–501.

Ankerst, M., Breunig, M., Kriegel, H.-P. & Sander, J. (1999), Optics: Ordering points to identify the clustering structure, *in* 'Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)', ACM Press.

Chomicki, J. & Revesz, P. (1999), 'Constraint-Based Interoperability of Spatiotemporal Databases', *GeoInformatica* **3**(3), 211–243.

Chudova, D., Gaffney, S., Mjolsness, E. & Smyth, P. (2003), Translation-invariant mixture models for curve clustering, *in* 'KDD '03: Procs. of ACM SIGKDD', ACM Press, pp. 79–88.

Ciaccia, P., Patella, M. & Zezula, P. (1997), M-tree: An efficient access method for similarity search in metric spaces, *in* 'VLDB'97', Morgan Kaufmann Publishers, Inc., pp. 426–435.

Ester, M. et al. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, *in* 'Second International Conference on Knowledge Discovery and Data Mining', AAAI Press, pp. 226–231.

Faloutsos, C. & Lin, K.-I. (1995), Fastmap: a fast algorithm for indexing of traditional and multimedia databases, *in* 'SIGMOD Conf.', ACM, pp. 163–174.

Gaffney, S. & Smyth, P. (1999), Trajectory clustering with mixture of regression models, *in* 'KDD Conf.', ACM, pp. 63–72.

Giannotti, F., Mazzoni, A., Puntoni, S. & Renso, C. (2005), Synthetic generation of cellular network positioning data, Technical report, ISTI-CNR.

Gudmundsson, J., van Kreveld, M. J. & Speckmann, B. (2004), Efficient detection of motion patterns in spatio-temporal data sets., *in* 'GIS', pp. 250–257.

Hadjieleftheriou, M., Kollios, G., Gunopulos, D. & Tsotras, V. J. (2003), On-line discovery of dense areas in spatio-temporal databases, *in* 'Proceedings of SSTD'03'.

Hwang, S.-Y., Liu, Y.-H., Chiu, J.-K. & Lim, E.-P. (2005), Mining mobile group patterns: A trajectory-based approach, *in* 'Proceedings of PAKDD'05'. To appear.

Iyengar, V. S. (2004), On detecting space-time clusters., *in* 'KDD', pp. 587–592.

Kalpakis, K., Gada, D. & Puttagunta, V. (2001), Distance measures for effective clustering of arima time-series., *in* 'ICDM', pp. 273–280.

Ketterlin, A. (1997), Clustering sequences of complex objects, *in* 'KDD Conf.', ACM, pp. 215–218.

Kriegel, H.-P., Brecheisen, S., Kröger, P., Pfeifle, M. & Schubert, M. (2003), Using sets of feature vectors for similarity search on voxelized cad objects., *in* 'SIGMOD Conference', pp. 587–598.

Kulldorff, M. (1997), 'A spatial scan statistic', *Comm. in Statistics: Theory and Methods* **26**(6), 1481–1496.

Li, Y., Han, J. & Yang, J. (2004), Clustering moving objects., *in* 'KDD', pp. 617–622.

Nanni, M. (2002), Clustering Methods for Spatio-Temporal Data, PhD thesis, CS Dept., Univ. of Pisa.

Saltenis, S., Jensen, C. S., Leutenegger, S. T. & Lopez, M. A. (2000), Indexing the positions of continuously moving objects, *in* 'Procs. of ACM SIGMOD', ACM, pp. 331–342.

Vlachos, M., Gunopulos, D. & Kollios, G. (2002), Discovering similar multidimensional trajectories., *in* 'ICDE', pp. 673–684.