Pervasive and Mobile Computing 42 (2017) 77–92

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

An approach for the discovery and validation of urban mobility patterns

Eugenio Cesario [a,*], Carmela Comito [a], Domenico Talia [b]

[a] ICAR-CNR, Italy
[b] DIMES, UNICAL, Italy

ARTICLE INFO

Article history:
Received 16 November 2016
Received in revised form 14 July 2017
Accepted 20 September 2017
Available online 7 October 2017

Keywords:
Trajectory pattern mining
Trajectory similarity
Urban computing

ABSTRACT

The increasing pervasiveness of mobile devices favors the collection of large amounts of movement data that can be analyzed to extract knowledge, i.e. patterns, rules and regularities, from user trajectories. In this paper we present TPM, an integrated algorithm which supports the overall trajectory pattern discovery process for detecting user's mobility behaviors. Specifically, the algorithm includes two main phases: (i) finding dense regions, more densely passed through ones; (ii) extracting trajectory patterns from those regions. Another contribution of the paper is a validation methodology for assessing the effectiveness of the TPM algorithm, e.g., evaluating how the discovered knowledge model fits to the input data it is discovered from. Such methodology represents a general solution that can be used to evaluate the accuracy of any algorithm aiming at extracting dense regions and trajectory patterns from GPS data. Furthermore, we propose novel trajectory similarity measures to evaluate the quality of the extracted patterns. A detailed experimental evaluation, performed by exploiting the proposed validation process, proves the efficiency and effectiveness of TPM.

1. Introduction

The widespread presence of wireless technologies, as well as the rapid development of satellite-enabled Global Positioning System (GPS) and mobile computing technologies, are providing large amounts of data pertaining to the mobility of people, cars, buses, trucks, etc. in urban areas. In particular, these networking infrastructures allow for sensing and collecting massive amount of spatio-temporal data. For instance, mobile phones leave positioning logs, which specify the cell they are connected in a cellular network. Analogously, GPS-equipped portable devices can record the latitude–longitude position everytime they are exposed to GPS satellites. These scenarios lead to the generation of a large number of trajectories drawn by mobile users during their daily activities allowing, thus, the collections of large amounts of movement data. The use of this data opens the opportunity of making discoveries about movement habits expressed as patterns, rules and regularities.

Analysis of people trajectories is a typical application of *urban computing* [1], an emerging multi-disciplinary research field that focuses on computing and digital networks in urban landscapes (e.g., cities, parks suburbs, etc.). The final goal consists in having smarter cities, by the application of ubiquitous and pervasive computing paradigms to urban spaces focusing on developing innovative services for citizens. In particular, traffic management is considered as one of the most challenging issues in urban areas and a great effort is being made to create more robust and intelligent systems that can be used to meet traffic flow needs.

* Corresponding author.
E-mail address: eugenio.cesario@icar.cnr.it (E. Cesario).

https://doi.org/10.1016/j.pmcj.2017.09.006
1574-1192/© 2017 Elsevier B.V. All rights reserved.

1.1. Motivations and contributions

Several approaches for trajectory pattern mining have been proposed in literature. Most of them are based on a combination of density-based clustering and sequential pattern discovery concepts, which are extensively used for detecting patterns from mobility data [2–7]. The general approach used to retrieve trajectory patterns consists of (i) discovering hotspots (e.g., single points or bounded areas) of interest and (ii) extracting mobility patterns among them. We refer to such an approach as *Density-based Sequential Pattern Mining* (DSPM).

The main contributions of the work proposed in this paper are summarized as follows:

- The *Trajectory Pattern Miner* algorithm, TPM. TPM implements the DSPM approach through an integrated algorithm which supports the overall trajectory patterns discovery process. The two main steps of the approach are: (i) finding frequent regions, and (ii) extracting sequential trajectory patterns among those regions.
- *A comprehensive validation methodology* for assessing accuracy and quality of detected dense regions and trajectory patterns, which consists of three steps: (i) generating synthetic trajectory data, adhering to some pre-defined mobility patterns and tuned by several input parameters, (ii) running a DSPM algorithm on such data to discover mobility models and (iii) finally computing how much the discovered models (i.e., dense regions and trajectory patterns) adhere to the characteristics of the input dataset they have been generated from.
- *Novel trajectory similarity measures* to evaluate the quality of the extracted patterns. Differently from the approaches in literature, such metrics are specifically tailored to trajectories among regions and not just GPS points. Thus, they are effective in providing the real geographic overlap among the regions involved in the mined trajectory patterns.

The proposed validation methodology represents a general solution that can be used to evaluate the accuracy of any of the algorithms implementing the DSPM approach introduced above, also including other kind of geo-referenced data like geo-tagged data from social networks (e.g., Twitter and Facebook). To the best of our knowledge, there are no other works in the literature proposing a comprehensive approach to validate both trajectory patterns and dense regions extracted from a GPS dataset.

For the sake of clarity, this paper largely extends the work presented in [8] and it provides several original contributions with respect to the previous one, as summarized in the following. First, the description of the trajectory pattern mining algorithm has been extended by enhancing it with formal definitions and a meta-coding of the algorithm (Section 3). Moreover, it describes a real-case study and shows the results of the algorithm on a real-world dataset. Then, the validation methodology (Section 4) has been extended by introducing further trajectory pattern similarity measures and metrics. Finally, several original contribution concerning the experimental evaluation (Section 6) have been reported in this paper.

1.2. Plan of the paper

The paper is structured as follows. Section 2 reviews related work on sequential pattern mining algorithms. Section 3 describes TPM, the proposed algorithm to discover trajectory patterns from large collections of moving object data, and the results of the analysis carried on T-Drive which we exploit as case study. The methodology introduced to validate the quality of the discovered patterns is detailed in Section 4. Section 5 presents novel similarity metrics designed to measure trajectory pattern similarity. Section 6 provides a detailed evaluation of the accuracy and effectiveness of the TPM algorithm, performed on several synthetic datasets. Finally, Section 7 concludes the paper and proposes some further research issues.

2. Related work: Density-based sequential pattern mining

Discovering patterns from historical object movements is a very challenging task and several algorithms tackle the problem by adopting sequential pattern mining approaches. Among all, particularly relevant are those based on a common inspiring idea that first detects geographic dense regions and then extracts sequential mobility patterns among such regions. As aforementioned in Section 1, we refer to such an approach as *Density-based Sequential Pattern Mining* (DSPM). In the following we review some representative algorithms of the DSPM approach, also sketching the main differences and similarities with the proposed TPM algorithm.

Mamoulis et al. [5] address the problem of mining sequential patterns from spatio-temporal data by considering the patterns in the form of trajectory segments. They first decompose the original trajectories into segments, then group them according to their shape and closeness. In particular, authors define the spatio-temporal periodic pattern mining problem and propose an algorithm for retrieving maximal periodic patterns based on a tree structure and an Apriori paradigm. In further extensions [6] and [9] of this work, authors study how to mine frequent sequences of trajectory segments and search approximate instances in the data representing approximate object movements over time.

In [2] authors extend the sequential pattern mining methodology to analyze moving objects, by proposing a method for extracting patterns containing both spatial and temporal information. In particular, a trajectory pattern describes movement trends in both spatial and temporal contexts, based on RoI (Region of Interest) detections. The approach first identifies RoIs, which are mostly visited regions, then, find frequent patterns from sequences of regions of interest. The sequences of RoIs not only represent the spatial movements but also the travel time of the moving objects. A similar approach has been presented in [3,4], where a trajectory pattern mining task is modeled as an extension of the association rules concept. In this case

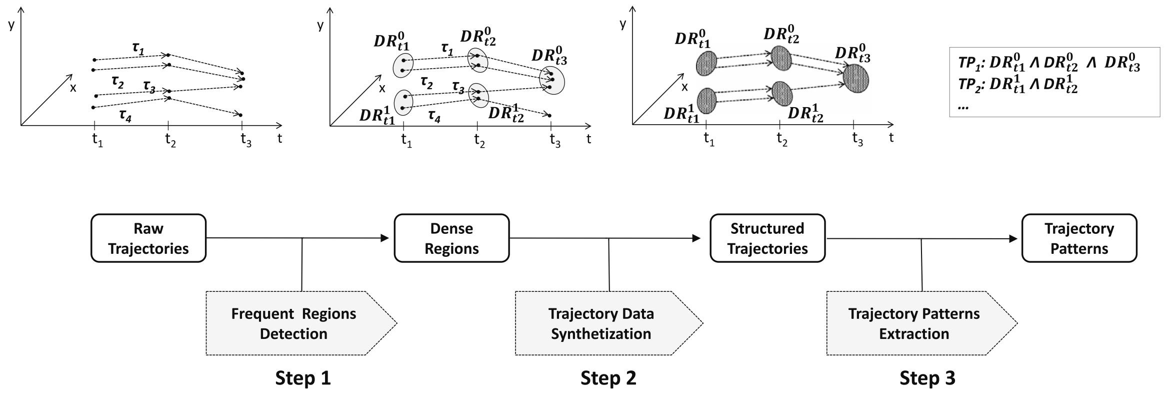


Fig. 1. Trajectory pattern detection steps.

retrieving trajectory patterns is essentially similar to that of mining association rules where it is necessary to bring the form of trajectory patterns from that of the rules.

The work in [10] provides a clustering-based perspective, and considers patterns in the form of moving regions within time intervals, such as spatio-temporal cylinders (or tubes) and counts as occurrences all trajectory segments partially contained in the moving regions.

In the rest of the section we summarize the differences among TPM and the main DSPM approaches reported above. The algorithms described in [5,6,9] are different from our approach in that they focus on extracting periodic patterns during a continuous sub-interval of the whole history. Moreover, their patterns are quite different: we consider patterns among regions while they use the objects coordinates. They mine recurring patterns of the same object whereas we mine patterns supported by many objects. Hence, the research problems addressed are quite different. The main difference between [2,11] and our work is that the trajectory patterns discovered in [2,11] represent sets of individual trajectories that visit the same sequences of places (ROIs) with similar travel times, and the typical travel time is included in the pattern. Conversely, our trajectory patterns represent group behavior, which is a key motivation in our work.

3. The TPM algorithm: Trajectory pattern miner

This section describes the Trajectory Pattern Miner algorithm, TPM, that we have designed following the DSPM approach. Section 3.1 depicts the main steps of TPM and the meta-code of the algorithm, whereas Section 3.2 describes the analysis performed by TPM on taxi trajectories in the urban area of Beijing and reports the mobility knowledge discovered in this real-world scenario.

3.1. The TPM algorithm

Fig. 1 sketches the general idea of the algorithm through a graphic representation of the whole process as a sequence of three main steps. The input data of the analysis is a set of raw trajectories that have been obtained by sampling real trajectories traced by users during their daily activities. The first step of the algorithm consists in the *detection of dense regions* from the original raw trajectory dataset. The goal of this step is detecting spatial areas more densely passed through, in order to conduct the further analysis as movements through areas rather than single points. This task can be modeled as a geo-spatial clustering instance and can be solved, as described in the following of the paper, by a density-based clustering algorithm. In figure, we represent with $DR^j_{t_i}$ the jth dense region at time t_i. The second step consists in the *synthesization of the trajectories*, by changing their representation from movements between points (raw data) into movements between dense regions (structured data). Precisely, each point of the original dataset is substituted by the region it belongs to. The third step is aimed at *extracting trajectory patterns*, in the form of sequential patterns, analyzing the trajectories of dense regions obtained at the previous step.

The meta-code of the algorithm is reported in Fig. 2. Input of the algorithm are: $\mathcal{D}$, i.e. the trajectory dataset, $param_{DRD}$ and $param_{TPE}$, i.e. parameters of the dense regions detection procedure and of the trajectory pattern extraction method, respectively. The output is the discovered knowledge, i.e., the dense region set $\mathcal{DR} = \{DR_{t_1}, \ldots, DR_{t_H}\}$ partitioned for timestamp, as well as the discovered trajectory patterns $\mathcal{TP} = \{TP_1, \ldots, TP_p\}$. The algorithm begins by partitioning the original trajectory dataset in a vertical way, with respect to the timestamp value (line L1). This is performed by the VerticalDataSplitting() method, whose goal is to split the original dataset $\mathcal{D}$ in $\mathcal{D}^{\perp} = \{D^{\perp}_{t_1}, \ldots, D^{\perp}_{t_H}\}$, where $D^{\perp}_{t_i}$ contains the points of the trajectories in $\mathcal{D}$ tagged with the timestamp t_i. At the end of this step, H different datasets are available. Then, each dataset is processed by the DenseRegionsDiscovery() method, aimed at discovering H clustering models, whereas the clusters of the t_h-model represent the detected dense regions of the t_h-timestamp (each cluster corresponds to a dense

```
TPM($\mathcal{D}$,$param_{DRD}$,$param_{TPE}$)
 Input:
     $\mathcal{D}$: trajectory dataset;
     $param_{DRD}$: parameters of the dense region discovery algorithm;
     $param_{TPE}$: parameters of the trajectory pattern extraction algorithm;
 Output:
     $\mathcal{DR} = \{DR_{t_1}, \ldots, DR_{t_H}\}$: dense regions, partitioned for timestamp;
     $\mathcal{TP} = \{TP_1, \ldots, TP_p\}$: a set of trajectory patterns;


 L1:  $(D^{\perp}_{t_1}, \ldots, D^{\perp}_{t_H}) \leftarrow$ VerticalDataSplitting($D$)
 L2:  for each timestamp $t_i = t_1, \ldots, t_H$ do
 L3:      $DR_{t_i} \leftarrow$ DenseRegionsDiscovery($D^{\perp}_{t_i}$, $param_{DRD}$)
 L4:      $\mathcal{DR} \leftarrow \mathcal{DR} \cup DR_{t_i}$
 L5:  end for
 L6:  $\mathcal{D}_{ST} \leftarrow$ TrajectoryDataSynthetization($\mathcal{DR}$);
 L7:  $\mathcal{TP} \leftarrow$ TrajectoryPatternDiscovery($\mathcal{D}_{ST}$, $param_{TPE}$);
      return ($\mathcal{DR}$, $\mathcal{TP}$)
```

Fig. 2. Trajectory Pattern Miner (TPM) algorithm.

region) (lines L2–L5). In our implementation, such step has been performed by applying DBSCAN [12], a density-based clustering algorithm that finds clusters with respect to the notion of density reachability among points. As soon as this step is completed, the raw trajectory dataset is converted into a structured trajectory dataset. This task is performed by executing the TrajectoryDataSynthetization() method, whose goal is to create a dataset where each point of the original trajectories is substituted by the dense region it belongs to. The final dataset, the *Structured Trajectory Dataset* $\mathcal{D}_{ST}$, results populated by trajectories between dense regions (not between single points). Finally, the TrajectoryPatternDiscovery() method discovers trajectory patterns. We implemented this step by the T-Apriori algorithm, an our ad-hoc modified version of the well known Apriori algorithm [13]. Differently from Apriori, T-Apriori extracts rules whose elements respect a monotonically increasing time order. That is, the timestamps of the antecedents are chronologically previous of those appearing in the consequent. The final mining model is a set of associative patterns describing spatio-temporal relations between the movement of the users under investigation.

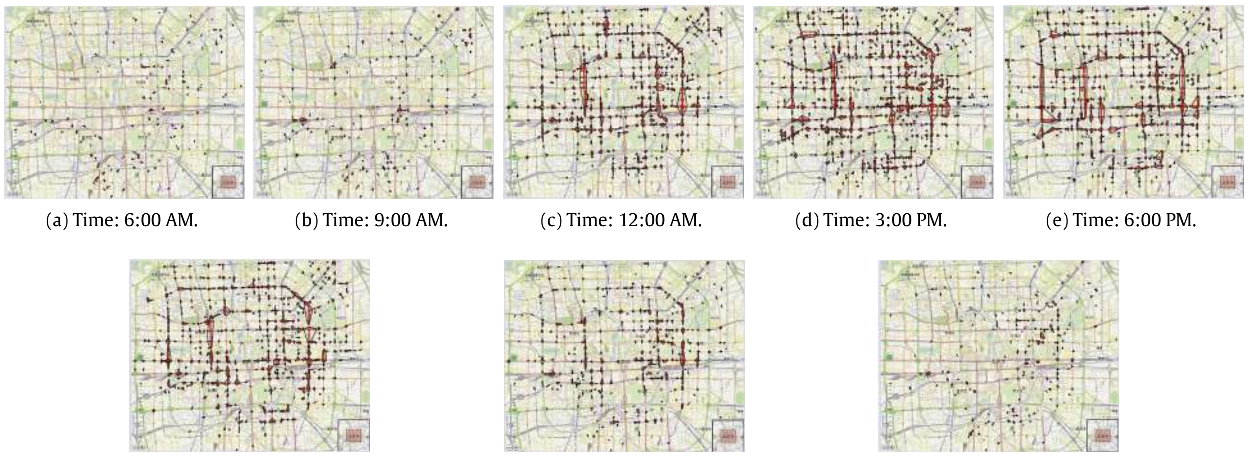
3.2. A real-case study: mobility analysis of taxies in Beijing

We consider as case study the analysis performed by TPM on T-Drive [14,15], a real-life GPS dataset tracing the movement of taxies in the urban area of Beijing. This dataset contains the GPS trajectories of 10,357 taxis during the period from February 2 to February 8, 2008 (one week) in Beijing. The total number of points is about 15 millions and the cumulative distance covered by the trajectories reaches almost 9 million kilometers.

Before the analysis of the trajectories, a pre-processing step has been performed to clean, select and transform data to make them suitable for analysis. The final dataset contains about 61,500 daily trajectories, each one containing the set of points traced by a single taxi during a day. The total data size is about 882 MB. We report in the following the results of the analysis performed on such a dataset, by showing dense regions (i.e., the most congested areas) of the city and mobility patterns with respect to different time periods.

Discovered Dense Regions. Dense regions discovered in T-Drive are shown in Fig. 3, for different time windows (3 h) of the day. Interestingly, such images show how the taxi mobility and traffic congestion change over the day. For example, we can observe that, during the early morning (around 6:00 AM and 9:00 AM), a low number of dense regions have been discovered, the most of them localized in South and West areas of the city (Figs. 3(a) and 3(b)). During the day, the distribution of vehicles increases and the traffic becomes more chaotic in several areas. In particular, we can observe (Figs. 3(c), 3(d), 3(e) and 3(f)) that from late morning to evening many regions in the city have an high concentration of taxies. We can clearly recognize the main streets that are used during these times: several highways crossing the central area of the city, as well as a circular highway around the city center (clearly observable in Fig. 3(d)) and an highway toward the airport. Finally, we can observe that during the night (Figs. 3(c) and 3(d)) the density of driving taxies strongly decreases, even remaining some areas more frequented with respect to others.

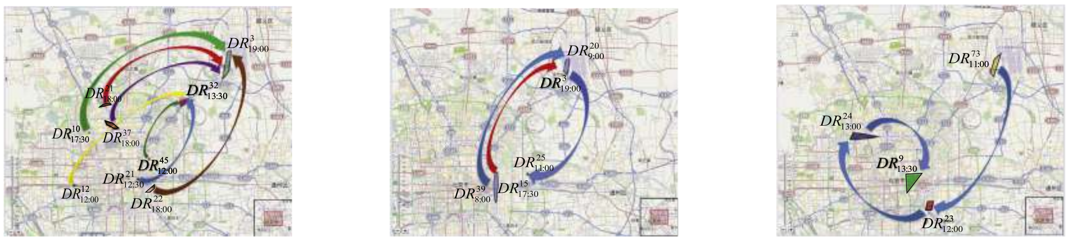
Discovered Mobility Patterns. Fig. 4 shows some examples of the most popular routes discovered in T-Drive by the TPM algorithm. To detect the most popular itineraries, we concentrate our analysis on routes around the city center and those going from the city center to strategic venues like the airport and the train stations. Fig. 4(a) highlights the main routes used by taxi drivers to leave the center toward the airport. It is evident that vehicles start from a broad diversity of locations in the city center, but converge toward two regions. Specifically, two mobility behaviors are evident: one leaving the city and



(a) Time: 6:00 AM. (b) Time: 9:00 AM. (c) Time: 12:00 AM. (d) Time: 3:00 PM. (e) Time: 6:00 PM.

(f) Time: 9:00 PM. (g) Time: 12:00 PM. (h) Time: 3:00 AM.

Fig. 3. Dense regions discovered in T-Drive, w.r.t. several time windows of the day.



(a) From the city center to the airport. (b) From the city center to the airport/sub-urban east area of the city. (c) From the airport to the train station.

Fig. 4. Travel patterns discovered in T-Drive by the TPM algorithm.

approaching the airport and another one leaving the city and approaching a venue outside the city center. In this case, a parking lot emerges as the top destination. One can note that the patterns approaching the airport from North are higher than those from South. The other behavior shown in Fig. 4(a) highlights a flow of people going from sub-urban/South-Center areas to region $\text{DR}^{3}_{19:00}$ that is a parking lot. For example, the route starting from the train station on the west (region $\text{DR}^{12}_{12:00}$ in Fig. 4(a)) or the ones originating from the South-Center. This flow could refer to people living outside the city that parked the cars in the parking lot to go to the city center and then coming back home in the suburbs after work. Fig. 4(b) shows a pattern starting from the city center, going through the airport and ending to a sub-urban east area of Beijing. In particular, the first flow represents the movement of people from the city center to the airport, e.g., going to work outside the city. The other flow goes from the airport to a suburban area, it could refer to people arriving to the airport and going back home in the residential sub-urban area. The pattern in Fig. 4(c) goes from the airport to a train station in the city center. The pattern is composed of 3 trips. The first goes from the airport to a popular venue in the sub-urban South area of the city. The second trips goes from this sub-urban area to the city center and the last one from the city center to the train station. This pattern could simply represent the most crowded route to get the central train station from the airport.

4. A validation methodology of the DSPM approach

This section presents a comprehensive validation methodology for assessing accuracy and quality of the results obtained by any density-based sequential trajectory pattern mining algorithm. In fact, the proposed methodology represents a general solution that can be used to evaluate the accuracy of any of the algorithms implementing the DSPM approach.

Model validation is a critical step of a knowledge discovery process that must be performed to assess how well the discovered model performs against new data. Since the final result of a DSPM algorithm is composed of two main models, i.e. dense regions and trajectory patterns, both must be evaluated to measure the adequacy of the discovered knowledge so that it can be interpreted objectively.

```
EVALUATE-QUALITY($N, H, P, \theta, \rho$)
 Input:
     $N$: number of trajectories;
     $H$: average length of trajectories;
     $P$: number of generative patterns to force within the data;
     $\theta$,$\rho$: outlier region percentage and noise degree;
 Output:
     $\mathcal{I_R}$: set of similarity indices for dense regions;
     $\mathcal{I_P}$: set of similarity indices for trajectory patterns.

 1: $(\mathcal{BRS}, \mathcal{ORS}) \leftarrow$ GENERATEREGIONS($H, \theta, P$)
 2: $(\mathcal{ST}, \mathcal{T}) \leftarrow$ GENERATETRAJECTORIES($N, H, P, \rho, \mathcal{BRS}, \mathcal{ORS}$);
 3: $\mathcal{TTP} \leftarrow$ TARGETTRAJECTORYPATTERNS($\mathcal{ST}$);
 4: $(\mathcal{DRS}, \mathcal{TP}) \leftarrow$ DSPM-ALGO($\mathcal{T}$);
 5: $\mathcal{I_R} \leftarrow$ COMPUTE_R-SIMILARITY($\mathcal{DRS}, \mathcal{BRS}$);
 6: $\mathcal{I_P} \leftarrow$ COMPUTE_P-SIMILARITY($\mathcal{TP}, \mathcal{TTP}$);
    return $(\mathcal{I_R}, \mathcal{I_P})$
```

Fig. 5. Validation algorithm.

To this purpose, we designed a *validation methodology* to assess how the discovered knowledge model fits to the input data it is extracted from. The general idea of the validation approach consists in (i) generating a synthetic dataset of trajectories, adhering to some pre-defined mobility patterns and tuned by several input parameters (number of dense regions, noise degree, average length of trajectories, etc.), (ii) running a DSPM algorithm (e.g., TPM) on such data to discover mobility models and (iii) finally computing how much the discovered models (i.e., dense regions and trajectory patterns) adhere to the characteristics of the input dataset they have been generated from. In this way, we are able to evaluate the accuracy of the discovered knowledge model, i.e. the similarity between the patterns generating the dataset and the models discovered from such data.

The overall validation algorithm is illustrated in Fig. 5. The input of the algorithm is represented by several parameters used to tune the synthetic dense regions (number of timestamps, percentage of outliers, etc.) and the trajectory data generation (number of trajectories, number of patterns, noise degree, etc.). The output is composed of several evaluation indices, grouped in the $\mathcal{I_R}$ and $\mathcal{I_P}$ sets, where the indices in $\mathcal{I_R}$ are targeted to evaluate the quality of the discovered dense regions and those in $\mathcal{I_P}$ are useful to assess the quality of the discovered patterns. According to the algorithm in Fig. 5, the validation process is composed of six steps, as detailed in the following.

Step 1. Target Dense Regions Definition. The first step, performed by the GENERATEREGIONS() method, is aimed at generating both a set of pre-defined dense regions, $\mathcal{BRS}$, and a set of outlier regions, $\mathcal{ORS}$. The elements in the first group, named also *target regions*, are considered as ground truth areas for all the evaluation process. In particular, the synthetic mobility patterns will be modeled as transitions among such regions. To evaluate the accuracy of the discovered mobility patterns, the base regions will be compared with the discovered dense regions. In addition, a set $\mathcal{ORS}$ of outlier regions will be generated and used for adding noise and perturbations inside data. The $\mathcal{BRS}$ set will be partitioned among patterns, while the $\mathcal{ORS}$ will be used to create noise and perturbations. Set sizes are fixed such that outlier regions amount to almost θ percent of the total number of regions: i.e., $|OR_{t_i}| = \theta \cdot (|OR_{t_i}| + |BR_{t_i}|)$, for each timestamp t_i.

Step 2. Synthetic Trajectory Data Generation. After the definition of base dense regions and outlier regions, the second step is aimed at building the trajectory dataset. This is performed by the GENERATETRAJECTORIES() method that creates a trajectory dataset satisfying input requirements (i.e., number of trajectories, overlap degree, etc.). The logic of the generation algorithm is summarized here. Starting from the set of base dense regions, $\mathcal{BRS}$, that have been yet partitioned among patterns, the procedure creates for each pattern *p* a certain number of structured and raw trajectories adhering to *p*. To introduce a certain level of noise and perturbations inside data, according to a specified noise degree, some regions are selected from the outlier set $\mathcal{ORS}$ (that is common for all the patterns). In this way, all the trajectories that should represent the same pattern will have similar routes, with some outliers whose occurrence rate can be tuned in input. Raw trajectories are generated by randomly selecting points in the dense/outlier regions occurring in the structured trajectories. The procedure returns the structured trajectory dataset $\mathcal{ST}$ and the raw trajectory dataset $\mathcal{T}$.

Step 3. Target Trajectory Patterns Detection. This step is aimed at mining the frequent mobility patterns through the TARGETTRAJECTORYPATTERNS() procedure that detects frequent patterns from the structured trajectory data $\mathcal{ST}$, by running T-Apriori on ST. In other words, it detects patterns whose items respect a chronological order. Those patterns will be then considered as target trajectory patterns (ground truth) for the evaluation, i.e., they will be compared with those discovered from the raw trajectories by applying any DSPM algorithm.

Step 4. A DSPM algorithm execution and Dense Regions/Frequent Patterns Extraction. This step consists in the execution of any DSPM algorithm (e.g., the proposed TPM) to discover frequent regions $\mathcal{DRS}$ and trajectory patterns $\mathcal{TP}$.

Steps 5 and 6. Similarity Computation. These two steps represent the final phase of the validation process and they actually perform the validation task. The procedure COMPUTE_R-SIMILARITY() computes the similarity between the target dense regions $\mathcal{BRS}$ and the discovered ones $\mathcal{DRS}$. Since such a process actually consists of the execution of a dense clustering algorithm, here we aim at the quality assessment of the clustering results. Accordingly, a set of external clustering validity indices and criteria available in the literature have been used to measure how much the discovered dense regions adhere to the target dense regions. As external indices we refer to the following ones: F-measure, Jaccard, Rand, Fowlkes and Γ. The procedure COMPUTE_P-SIMILARITY() compares the target trajectory patterns $\mathcal{TTP}$ and the discovered ones $\mathcal{TP}$ to evaluate the quality of the trajectory pattern discovery approach. The pattern similarity indices adopted in this step are described in detail in Section 5.

5. Trajectory pattern similarity measures

To validate the efficacy and accuracy of the trajectory pattern discovery step of a DSPM algorithm a feasible way is to assess how much the *discovered* trajectory patterns adhere to the *target* patterns obtained from the synthetic trajectory data. To reach this goal, the validation process actually consists of measuring the *similarity* of the target trajectory patterns to the extracted ones. As trajectory patterns are actually trajectories, the problem can be formulated as a trajectory similarity problem, that is: *given the set of target trajectory patterns $\mathcal{TTP}$ and the set of discovered trajectory patterns $\mathcal{TP}$, we formulate the validation of the trajectory patterns extracted by a DSPM algorithm as a similarity problem among each pair of trajectory patterns, $P = (R_1, \dots, R_n)$ and $Q = (R_1^\diamond, \dots, R_m^\diamond)$, with $P \in \mathcal{TP}$ and $Q \in \mathcal{TTP}$.*

In the following of the section we first provide some insights on the main trajectory similarity approaches and then describe the proposed similarity measures.

5.1. Trajectory similarity: main approaches

Several trajectories distance/similarity measures have been proposed in literature, e.g., Euclidean distance (ED) [16], Dynamic Time Warping (DTW) [17], distance based on Longest Common Subsequence (LCS) [18–20], Edit Distance with Real Penalty (ERP) [21], Edit Distance on Real sequence (EDR) [22].

Among such approaches, the LCS-based ones are the most widely adopted for trajectories similarities. Given two patterns, the more similar they are, the longer common part they share. The LCS problem is the problem of finding the longest subsequence common to all sequences in a dataset and then define the distance using the length of this subsequence. In [23] is proposed the modified longest common subsequence (MLCS) (see Eq. (1)) metrics that elaborates on the LCS for better robustness to noise. The main idea here is that the match of two trajectory patterns allows some elements to be unmatched within a minimum bounding envelope, but the order of elements in the match must remain. Two patterns P and Q are deemed to be similar, if there exists a long subsequence P_1 of P that can be mapped to a long subsequence Q_1 of Q. The similarity of P and Q is then defined as follows:

$$MLCS - Sim(P, Q) = \frac{|LCS(P, Q)|}{MAX(|P|, |Q|)} \tag{1}$$

Another common approach to compare trajectories is based on the *maximal trajectory pattern* (MTP) similarity that is a pattern that is not contained in any other pattern. Specifically, we consider for our evaluation (see Section 6.2) the *Maximal Semantic Trajectory Pattern similarity* (*MSTP*) introduced in [24].

The MSTP measure computes the similarity between two maximal semantic trajectory patterns by averaging the participation ratios of their common part which is as follows:

$$ratio(LCS(P, Q), P) = \frac{\sum_{i=1}^{|P|} \sum_{j=1}^{|LCS(P,Q)|} M(P_i, LCS_j)}{|P|} \tag{2}$$

where $M(P_i, LCS_j) = \frac{|P_i \cap LCS_j|}{|P_i|}$ *if* LCS_j *matches* P_i , 0 *otherwise*.

Two different MSTP metrics are proposed in [24], The first index, referred to as Equal Average (EA), given two patterns P and Q, computes the average of the two ratios to P and Q, as shown in Eq. (3). The other index computes the Weighted Average (WA) (see Equation (4)) proportionally to the lengths of the two patterns. This choice is motivated by the fact that a longer pattern provides more information than a shorter one.

$$MSTP - Sim_{EA}(P, Q) = \frac{ratio(LCS(P, Q), P) + ratio(LCS(P, Q), Q)}{2} \tag{3}$$

$$MSTP - Sim_{WA}(P, Q) = \frac{|P| \times ratio(LCS(P, Q), P) + |Q| \times ratio(LCS(P, Q), Q)}{|P| + |Q|} \tag{4}$$

5.2. Proposed similarity measures

In this section, we describe two novel trajectory patterns similarity measures specifically tailored to trajectories.

The first measure proposed is a distance-based similarity. A generic distance-based similarity measure can be expressed by the following equation:

$$Dist - Sim(P,Q) = 1 - \frac{\sum_{i=1}^{MIN(|P|,|Q|)} d(P_i, Q_i)}{MIN(|P|,|Q|)} \tag{5}$$

As in our scenario trajectory patterns are actually trajectories among regions, the distance $d(P_i, Q_i)$ in Eq. (5) expresses the distance on spatial space among the centroids of the respective regions crossed at the same timestamp *i*. In turn, a centroid of a geographic area is characterized by the pair of latitude and longitude coordinates. Given the two patterns $P = (R_1, \dots, R_n)$ and $Q = (R_1^\diamond, \dots, R_m^\diamond)$, we denote the centroid of a region R_i of P as c_{R_i} and in the same way we denote as $c_{R_i^\diamond}$ the centroid of a region $R_i^\diamond$ of Q. Accordingly, $d(P_i, Q_i) = d(c_{R_i}, c_{R_i^\diamond})$ in Eq. (5).

Among the distance-based similarity approaches, the well-known Euclidean distance is widely adopted for trajectory similarities [16,25,26]. According to the standard definition of the euclidean distance, Eq. (5) becomes the euclidean similarity (ED-Sim) as expressed by the following formula:

$$ED - Sim(P,Q) = 1 - \frac{\sum_{i=1}^{MIN(|P|,|Q|)} \sqrt{(lat_{c_{R_i^\diamond}} - lat_{c_{R_i}})^2 + (lon_{c_{R_i^\diamond}} - lon_{c_{R_i}})^2}}{MIN(|P|,|Q|)} \tag{6}$$

We elaborate on the euclidean distance to propose a new similarity metrics that is specifically tailored to trajectories; this metrics is based on the Haversine distance [27]. Accordingly, we determine the distance among two GPS points, expressed as latitude and longitude coordinates, by using the Haversine formula [27]. The Haversine formula is an equation giving great-circle distances (that is, the shortest distance over the earth's surface) between two points on a sphere from their longitudes and latitudes. Accordingly to the Haversine formula, the distance $d(c_{R_i}, c_{R_i^\diamond})$ in Eq. (5) is expressed by the following equation:

$$d_h(c_{R_i}, c_{R_i^\diamond}) = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{lat_{c_{R_i^\diamond}} - lat_{c_{R_i}}}{2}\right) + \cos(lat_{c_{R_i}})\cos(lat_{c_{R_i^\diamond}})\sin^2\left(\frac{lon_{c_{R_i^\diamond}} - long_{c_{R_i}}}{2}\right)}\right) \tag{7}$$

where $r = 6,371$ Km is the earth's mean radius. The Haversine distance-based similarity (HD-Sim) is, thus, represented as follows:

$$HD - Sim(P,Q) = 1 - \frac{\sum_{i=1}^{MIN(|P|,|Q|)} d_h(c_{R_i}, c_{R_i^\diamond})}{MIN(|P|,|Q|)} \tag{8}$$

Although many similarity measures treat in a proper way spatio-temporal data, they are not effective in the faced setting where we deal with trajectories among regions (set of GPS points) and not among simple GPS points. In fact, the current approaches to trajectory similarity, including the LCS-based and the distance-based ones, fail to catch the real similarity among two trajectories as they do not take into consideration the geographic extension of the regions, resulting this way to be quite imprecise in the computed similarity measures. Specifically, distance-based metrics just compute the distance among pairs of points (precisely among the centroids of the regions crossed in the same timestamp). On the other hand, the LCS approach may lead to some inaccuracy since it does not consider various unmatched sequences in trajectories. Nevertheless, it approximates to 1 the similarity among each pairs of matched subsequences, regardless of the actual geographic overlap among the matching regions.

Due to these reasons, we introduce a new index specifically tailored for trajectories among regions: the *overlap similarity* index that determines the exact geographic overlap between each pair of regions traversed in the same timestamp in the compared patterns. Therefore, the basic concept here is the geographic overlap among two regions at a given timestamp.

Definition 1. Geographic overlap among two regions.

Given two regions R_i and $R_i^\diamond$, we define the geographic overlap among them as the overlapping of the areas of the polygons accounting those regions:

$$AreaOverlap(R_i, R_i^\diamond) = \frac{area(R_i) \cap area(R_i^\diamond)}{area(R_i) \cup area(R_i^\diamond)} \tag{9}$$

According to the definition in (9) we introduce the overlap similarity metrics, defined as follows:

Definition 2. Overlap similarity.

The overlap similarity among two trajectory patterns P and Q is defined by accounting the geographic area overlap of each pair of regions crossed in the same timestamp:

$$Overlap - Sim(P,Q) = \frac{\sum_{i=1}^{MIN(|P|,|Q|)} AreaOverlap(R_i, R_i^\diamond)}{MAX(|P|,|Q|)} \tag{10}$$

In order to assess how good is the similarity that the state of the art approaches compute, whether they are able to reflect the true underlying similarity among the compared patterns in terms of spatial coverage, we calculate how much they deviate from the overlap-similarity value that we assume the real geographic intersection among two patterns, assumption that we have to make since we do not have ground truth knowledge about it. Nevertheless, this assumption it is rather realistic since the overlap-similarity is able to compute the geographic overlap among regions.

Specifically, to account how much the similarity metrics deviates from the actual geographic overlap among two patterns we introduced the following metrics.

Definition 3. Overlap Degree of a similarity metrics.

The overlap degree represents how much a given similarity metrics is able to account the geographic coverage overlap among two trajectory patterns, as expressed by the following equation:

$$Overlap - Degree(Sim(P, Q)) = \frac{Overlap - Sim(P, Q)}{Sim(P, Q)} \qquad (11)$$

where $Sim(P, Q)$ is one of the above similarity metrics. The higher the overlap degree value, the smaller the metrics deviates from the overlap similarity, the better the metrics accounts the geographic overlap among two patterns.

As explained above, distance-based metrics are not so effective in this direction as they can give just information about the distance among the centroid points of the regions involved in the compared patterns. Consequently, we cannot consider them for this evaluation.

6. Experimental evaluation

In this section we provide a detailed evaluation of the accuracy and effectiveness of TPM, our implementation of the DSPM approach, by exploiting the validation methodology proposed in Section 4. To validate TPM, we carried out a performance analysis by executing different experiments in various scenarios. We used synthetic data in order to assess the quality of the dense regions and trajectory patterns discovered by TPM and, in particular, to assess its capability to behave properly in extreme data contexts (Sections 6.1 and 6.2). Finally, we report a comparative experimental evaluation of TPM with the main DSPM approaches proposed in literature (Section 6.3).

6.1. Evaluation of the discovered dense regions

The aim of validation is to compare the results of the cluster analysis, i.e. the *discovered dense regions*, to externally known results, i.e. the *target dense regions*. Accordingly, a set of external clustering validity indices and criteria available in the literature have been used to measure how much the discovered dense regions confirm to the target dense regions. As external indices we refer to the following ones: (i) *Accuracy*, (ii) $F - measure$, (iii) *Jaccard*, (iv) *Rand*, (v) *Fowlkes* and (vi) Γ. Large values of such indices imply close agreement between discovered and target dense regions. In particular, the *Accuracy* is the proportion of points correctly assigned to the clusters in accordance with the ground truth model; $F-measure$ and *Jaccard* represent the compactness of the discovered structure; *Rand* is an evaluation measure of separability of the discovered dense regions related to the hypothesized ones (target dense regions); *Fowlkes* and Γ compute similarity and correlation between the discovered clusters and the ground truth.

The dense regions discovery step has been performed by the DBSCAN algorithm, which requires two input parameters, i.e. ϵ (radius of the neighborhood) and *minPoints* (minimum number of objects for a cluster). In particular, the ϵ value strongly leads the accuracy of the frequent region detection phase. In addition, the data generation is influenced by two parameters in particular, i.e. ρ and θ, which tune noise and perturbations inside data. Thus, in order to make a complete picture of the performance, we generated several datasets with respect to different combinations of ρ and θ values. Then, we run several DBSCAN instances by varying ϵ. The values of the aforementioned parameters have a direct influence on the quality of the results, as highlighted in the following charts.

A first set of experimental results were obtained by generating data with fixed values $\theta = 0.3$ and $\rho = 0.1$ (30% of outliers and 10% of noise), with ϵ varying from 0.03 to 0.4. Fig. 6 shows how external indices vary with respect to ϵ values (X-axis is ticked in log scale). It is worth noting that the trend is strongly affected by the values of ϵ, whose best value in this case can be clearly estimated equal to 0.01. To better show the variability with respect to ϵ values around 0.01, Table 1 reports the values of the indices, for ϵ ranging in a tight around of 0.01 (i.e., from 0.008 to 0.012). It is clear that the highest quality clusters are obtained when the indices value is 1.0, i.e. for $\epsilon = 0.01$. It is worth nothing that the clustering quality is in general higher than 0.8 for all ϵ values ranging between 0.006 and 0.06.

Now, to better assess the quality of results as well as the robustness of the algorithm, it is useful to perform an analysis for several values of outlier (θ) and noise (ρ) degrees. First, let us consider the case with a low noise degree and a variable number of outlier regions degree. Fig. 7 shows four selected indices, i.e. Accuracy, F-measure, Jaccard and Gamma, with respect to ϵ, fixing $\rho = 0.1$ and for θ varying from 0.3 to 0.7. As we can see, the algorithm is highly tolerant to high values of outlier region degree (θ). Interestingly, inside the range $\epsilon = [0.06, \dots, 0.1]$, all the indices show good clustering quality for all values of θ (with the peak for $\epsilon = 0.01$, as discussed above). It is worth nothing that for high values of θ (i.e., 0.6 or 0.7) the quality has a small reduction with respect to the best case. This means that the algorithm discovers suitable dense regions even when trajectories are traced by a high number of outlier points.

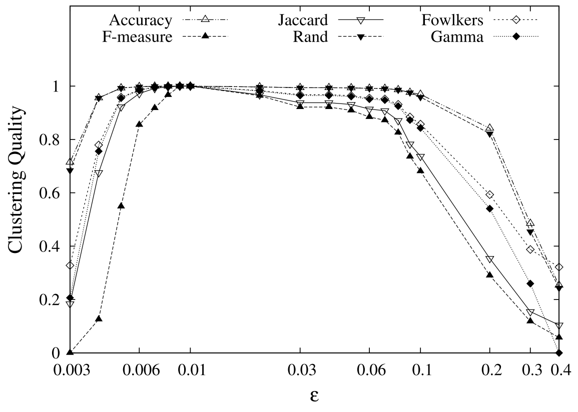


Fig. 6. Clustering quality w.r.t. ϵ (in log scale), for $\theta = 0.3$ and $\rho = 0.1$.

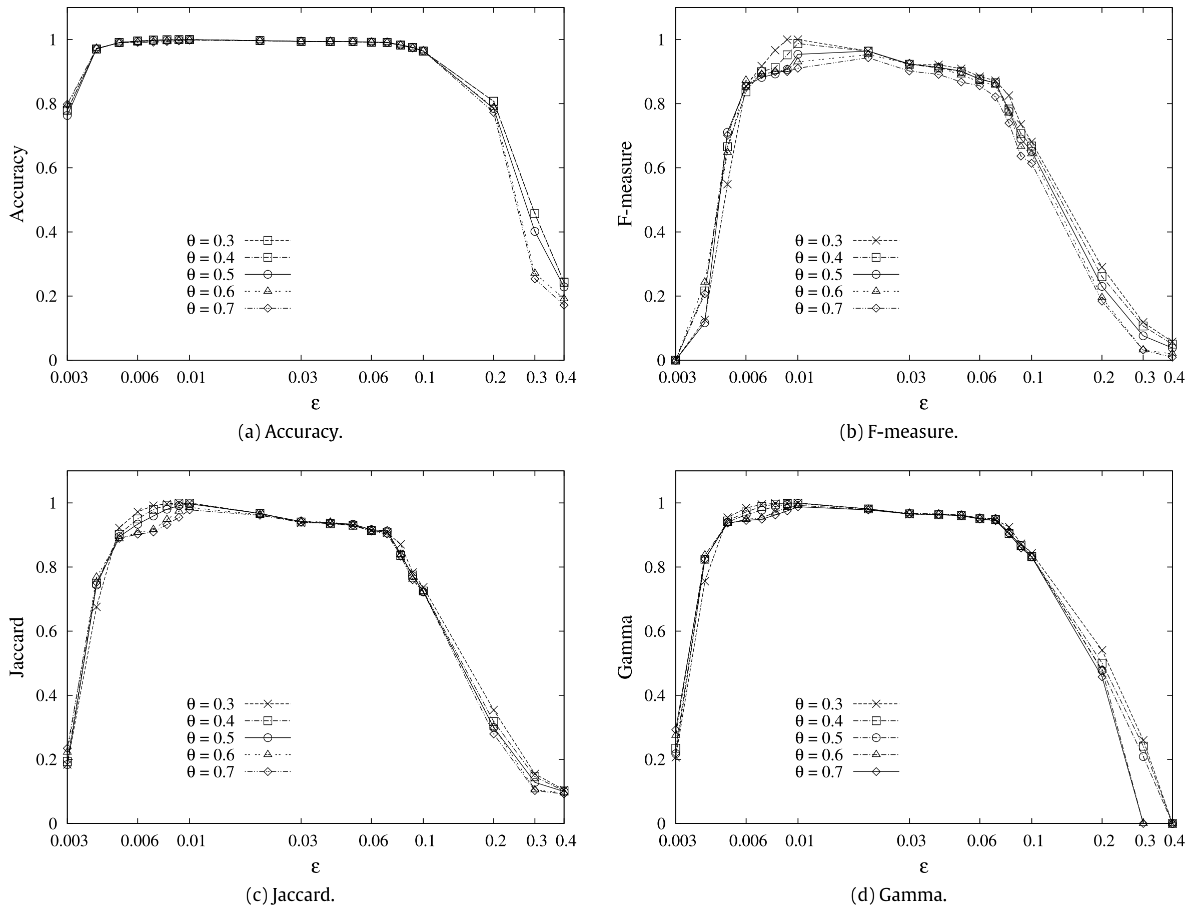
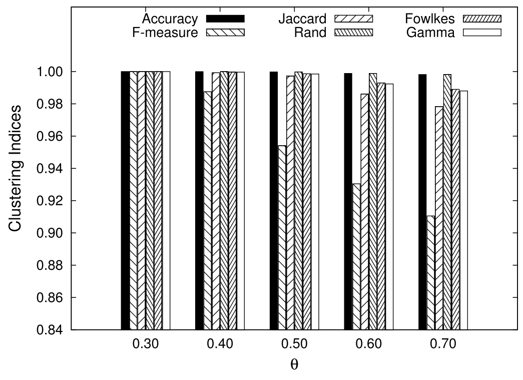


Fig. 7. Some quality indices, versus ϵ, for several values of θ and $\rho = 0.1$.

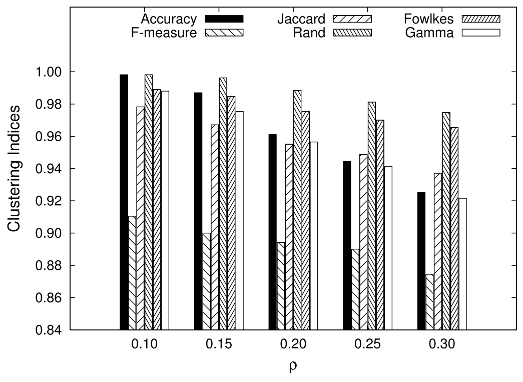
Fig. 8 shows some quality indices, for $\epsilon = 0.01$ and different combinations of θ and ρ values. In particular, Fig. 8(a) reports the results for $\rho = 0.1$ and θ varying from 0.3 to 0.7. For a relatively low degree of outlier regions (i.e., $\theta \leq 0.5$) the quality assumes values higher than 0.94. Moreover, even with a high number of outlier items (i.e., $\theta > 0.5$) the quality maintains values over 0.9. On the other side, Fig. 8(b) reports the results for the worst case of outlier degree, i.e. $\theta = 0.7$, and ρ varying from 0.1 to 0.3. Also in this case, we can observe that the clustering quality get values higher than 0.9. We can conclude that,

Table 1
Clustering quality indices vs ϵ values, for $\theta = 0.3$ and $\rho = 0.1$.

	ϵ				
	0.008	0.009	0.010	0.011	0.012
Accuracy	0.9981	0.9997	**1.0**	0.9988	0.9985
F-measure	0.9668	0.9989	**1.0**	0.9967	0.9854
Jaccard	0.9971	0.9983	**1.0**	0.9984	0.9978
Rand	0.9978	0.9995	**1.0**	0.9989	0.9981
Fowlkes	0.9975	0.9977	**1.0**	0.9963	0.9877
Gamma	0.9984	0.9992	**1.0**	0.9976	0.9867



(a) Vs θ, fixed $\epsilon = 0.01$ and $\rho = 0.1$.



(b) Vs ρ, fixed $\epsilon = 0.01$ and $\theta = 0.7$.

Fig. 8. Some quality indices, with $\epsilon = 0.01$, for several values of θ and ρ.

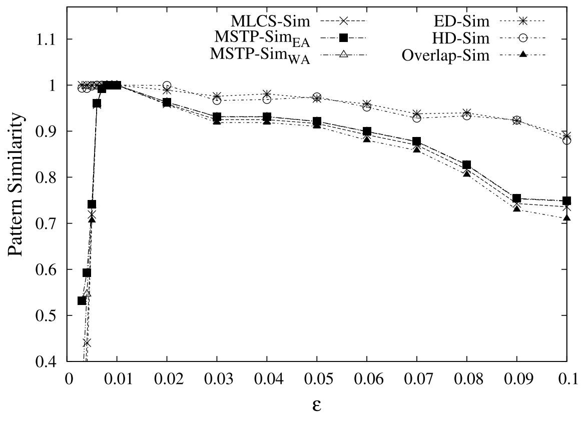
even combining a high degree of overlap and a high number of outlier items, the algorithm is able to discover good-quality clusters, that is, it separates well the estimated clusters.

6.2. *Evaluation of the mined trajectory patterns*

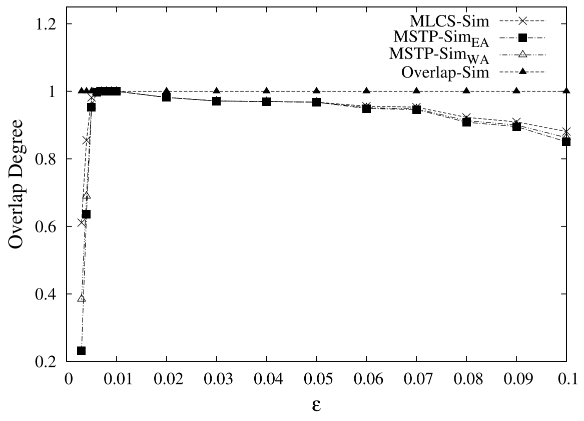
We validate the trajectory patterns extracted by TPM evaluating their similarity to the target trajectory patterns obtained from the synthetic dataset. To this aim we exploited the trajectory similarity metrics introduced in Section 5. Specifically, we use the proposed metrics *Haversine* distance ($HD - Sim$) and *Overlap Similarity* ($Overlap - Sim$) and comparing their efficacy against some representative approaches in literature: the Euclidean distance ($ED - Sim$) and three LCS-based approaches that are the Maximal Semantic Trajectory Pattern similarities ($MSTP - Sim_{EA}$ and $MSTP - Sim_{WA}$ [24], respectively) and the Modified Longest Common Subsequence ($MLCS - Sim$ [23]).

Fig. 9(a) shows the different trajectory pattern similarity metrics with respect to different values of ϵ. As one can notice, all the considered similarity metrics exhibit a notable trend, meaning that our trajectory pattern detection algorithm extracts patterns with very high accuracy, as the extracted patterns are highly similar to the target ones. In particular, according to previous experiments on dense region evaluation (see Section 6.1), the highest similarity value for all the metrics is obtained in correspondence of the ideal value of ϵ equals to 0.01. For that value we also obtain the highest values for all the dense region validation indices. From the graph is evident that both the distance-based metrics present the highest similarity that remains in the range of 1 with ϵ. The similarity computed by the LCS-based metrics slightly decreases for $\epsilon > 0.05$ but still remaining quite high around 1. Whereas, for the overlap similarity index the similarity decreases after the ideal value of ϵ, reaching the value of about 0.82 with $\epsilon = 0.1$.

The difference in the performance of the different metrics can be explained as follows. The distance-based metrics ($ED{-}Sim$ and $HD{-}Sim$) evaluate the similarity by accounting the distance of the centroid points in the regions. Consequently, even if the centroid points of each pair of regions are very close nothing can be said about the overlap of the two regions. Therefore, even if the distance-based metrics are higher, using them we cannot catch the real similarity among detected and target patterns that instead the other metrics do. For this reason, the introduced *overlap degree* index cannot be applied on the distance-based metrics. Differently, we use the overlap degree to compare the overlap similarity to the LCS-based ones. More precisely, since the LCS-based metrics do not consider the real geographic overlap of the regions involved in the trajectory patterns, approximating the regions similarity to 1 or 0 according to the label match, we need to look at the overlap degree metric to fully evaluate the similarity index. Fig. 9(b) shows the overlap degree for different values of ϵ for all the LCS-based similarity and the overlap similarity metrics. The chart shows that the overlap degree of the overlap similarity is always 1 regardless of the value of ϵ, while for the other metrics the overlap degree decreases for values of $\epsilon > 0.01$. This

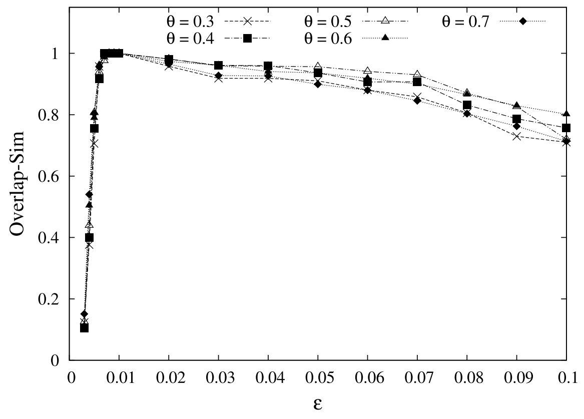


(a) Vs ϵ, fixed $\theta = 0.3$ and $\rho = 0.1$.

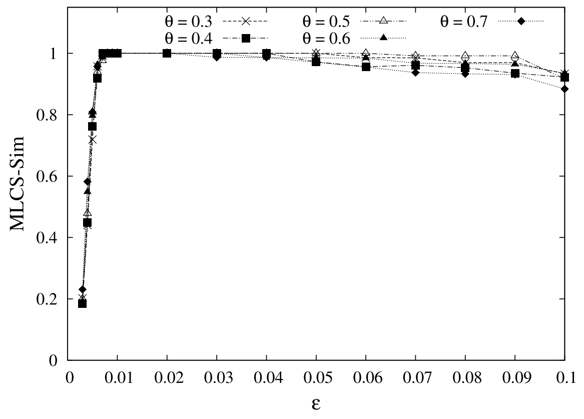


(b) Vs ϵ, fixed $\theta = 0.3$ and $\rho = 0.1$.

Fig. 9. Pattern similarity (a) and Overlap degree of similarity indexes (b) vs ϵ.



(a) Overlap similarity.



(b) MLCS similarity.

Fig. 10. Overlap similarity (a) and MLCS similarity (b) w.r.t. ϵ and θ, fixed $\rho = 0.1$.

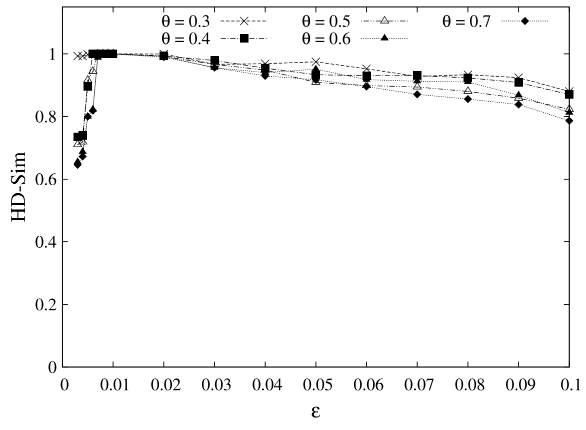
result shows that the only metrics able to account the real geographical overlap among two patterns is the overlap similarity and, thus, it is the metrics that gives the most reliable similarity measure among two trajectory patterns.

We also evaluated the trajectory pattern extraction technique when noise is introduced in the data. For the LCS-based metrics we show only the MLCS similarity, as results are similar. Fig. 10(a) shows how the overlap similarity index varies with ϵ for different outlier percentage values (θ), while the noise degree (ρ) is fixed and set to 0.1. The graph points out that the similarity is 1 in correspondence of the optimal value of ϵ (0.01), then remains quite high, around 1, up to values of ϵ equal to 0.03, regardless of the outlier percentage. For higher values of ϵ the similarity still kept high (around 0.9) except for high values of θ (i.e, $\theta = 0.6$ and $\theta = 0.7$) where it decreases. This decrease becomes significant, around 0.8, for very high values of θ and ϵ. Overall, this outcome highlights that the trajectory pattern extraction technique exhibits robustness while noise is introduced in the data and also when ϵ takes values that lead to perturbations in the construction of regions.

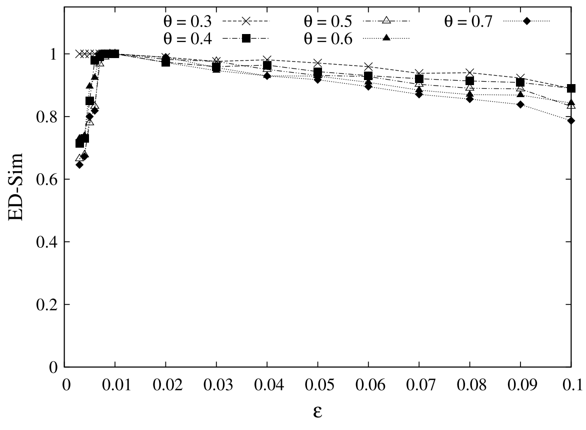
Also the other similarity metrics exhibit robustness while noise is introduced in the generated data. Figs. 10(b), 11(a) and 11(b) show that the similarity is always very high, despite the introduced noise. However, the MLCS metrics does not capture the real geographic overlap among the patterns and, thus, it results to be a little bit imprecise.

6.3. Comparative analysis with related approaches

In this section we compare the proposed TPM algorithm against some of the most representative DSPM approaches. Specifically, we first summarize the main features of the different approaches and then compare their performance by exploiting the validation approach presented in the paper.



(a) Haversine distance similarity.



(b) Euclidean Distance similarity.

Fig. 11. Haversine distance similarity (a) and Euclidean Distance similarity (b) w.r.t. ϵ and θ, fixed $\rho = 0.1$.

We identified the main features characterizing DSPM algorithms (i.e., dense region detection approach, dense region shapes, sequential pattern mining approach, etc.) and compared them on the basis of such one, as summarized in Table 2. The comparison takes into account the following seven features:

1. *Dense region detection*. This feature describes whether the approach implements a method to automatically detect dense regions from raw trajectory data. This is a crucial issue for the effectiveness of a DSPM algorithm since the whole process is influenced by the quality of the regions considered for the analysis. The proposed TPM algorithm and the approaches presented in [10,2] implement methods to detect dense regions from raw trajectory data, whereas the rest of the related works rely on pre-defined regions, like grids and point of interests (POIs) [3,4,6]. The limit of the latter approaches is that they rely on a static subdivision of regions and categorization of them as region/point of interests. Such predefined classification of regions could generate two main drawbacks: including regions not interesting in terms of mobility analysis and excluding others that are not classified as POI in the reference sources (e.g., tourist guides, city districts) and from where many mobility data are generated. Differently, TPM and the works in [10,2] are able to identify relevant locations since regions are automatically derived from actual data, instead of being statically defined a priori.
2. *Dense region detection approach*. We also classify the compared systems on the basis of the approach used to detect dense regions, when applicable. In fact, as previously highlighted, the systems presented in [3,4,6] do not use any dense region detection approach as they rely on pre-defined regions. All the other systems implement a density-based clustering approach to identify dense regions, each one with specific peculiarities. The TPM approach extends the well-known density-based clustering algorithm DBSCAN to deal with spatio-temporal data. Giannotti et al. [2] exploits an ad-hoc density-based clustering algorithm proposed in [28] opportunely adapted to deal with trajectory data. Kalnis et al. [10] proposed an ad-hoc density based clustering algorithm that exploits DBSCAN to identify moving clusters.
3. *Dense region shapes*. Another important feature for classification purpose is the shape of the dense regions. In fact, this feature allows to assess the ability of the detection approach in identifying any possible dense spatial area, regardless of the shape. The more shapes the algorithm is able to catch, the better the accuracy and effectiveness of the detected dense regions. TPM and the work described in [10] are able to detect regions of any shape (e.g., circular, rectangular, linear) while the works in [2,4,6] deal with only rectangular regions within grids. Finally, the work presented in [3] considers hexagonal cell network area.
4. *Dense region dimensions*. A relevant distinctive feature among the considered approaches is related to how regions' density is computed with respect to space and time dimensions. More specifically, the algorithms described in [2–4,6,10] deal with only spatially-labeled dense regions (e.g., an object passes through the *j*th region), while TPM discovers spatio-temporal dense regions (e.g., an object passes through the *j*-th dense region at the time (t).
5. *Transition Time*. This feature specifies whether a system includes the transition time from a dense region to the consecutive one in the patterns. The only two approaches addressing this feature are TPM and the one proposed by Giannotti et al. [2]. Specifically, in [2] the patterns are labeled with typical transition times between regions but no information about the time when the movements take place is given. Differently, TPM identifies patterns among regions that are dense both in the space and time, therefore, it specifies exactly at what time a person is in a given region. Moreover, the time of movement can be computed as the difference between the timestamps of two consecutive regions in a pattern.

Table 2
Comparison of TPM with other approaches proposed in literature.

	Dense region detection	Dense region detection approach	Dense region shapes	Dense region dimensions	Transition time	Sequential spatio temporal patterns	Sequential pattern mining approach
TPM	*Yes*	*Density-based clustering*	*Anyshape*	*Spatial Temporal*	*Yes*	*Yes*	*A-Priori like*
Giannotti et al. [2]	Yes	Density-based clustering	Rectangular	Spatial	Yes	Yes	PrefixSpan like
Yang et al. [4]	No	N.A.	Rectangular	Spatial	No	No	Min–Max Pattern Mining
Kalnis et al. [10]	Yes	Density-based clustering	Any shape	Spatial	No	Yes	Clustering
Cao et al. [6]	No	No	Rectangular	Spatial	No	Yes	Substring tree
Yavas et al. [3]	No	N.A.	Hexagonal	Spatial	No	No	Graph-based algorithm

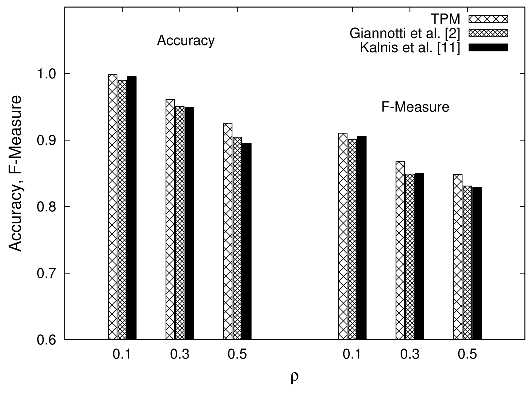
6. *Sequential Spatio-Temporal Patterns*. This feature considers whether patterns are discovered with respect to spatial and temporal dimensions. In particular, TPM and the approaches discussed in [2,10,6] discover spatio-temporal annotated sequences (i.e., an extension of classic sequential frequent patterns enriched with temporal annotations), while the systems presented in [4,3] discover only sequential patterns among spatial regions (without time details).
7. *Sequential Pattern Mining Approach*. This feature classifies the systems on the basis of the sequential pattern mining approach used to mine patterns. Specifically, TPM exploits T-Apriori (an our ad-hoc modified version of the well-known Apriori algorithm [13]), while a projection-based approach (PrefixSpan-like) is adopted in [2]. In [4] is presented the TrajPattern algorithm that mines patterns exploiting the normalized match measure. In the approach described in [6] patterns are detected by performing substring counting exploiting a substring tree structure. Finally, the work described in [3] models trajectories as a graph and discovers mobility patterns by a graph traversal algorithm.

From the above comparative evaluation, we can conclude that the proposed TPM algorithm exhibits several distinguished benefits. First, it detects actual dense regions as they emerge from real data without relying on static subdivisions of the spatial area. In doing so, it achieves high accuracy since it is able to catch regions of any shape. This is also confirmed by Fig. 12(a), showing that TPM achieves the highest accuracy with respect to other approaches (further details about the experiments concerning this results are provided in the following). Moreover, TPM is the only approach that computes the density of regions on the basis of both spatial and temporal dimensions. This allows to obtain spatio-temporal patterns, and deduce transition times among regions in the patterns. Finally, another novelty of our work is the validation approach assessing accuracy and quality of dense regions and trajectory patterns. Specifically, we defined a new trajectory similarity measure to evaluate the quality of the extracted patterns that, differently from the approaches in literature, is specifically tailored for trajectory among regions and not just GPS points. Thus, it is effective in providing the real geographic overlap among the regions involved in the mined trajectory patterns.

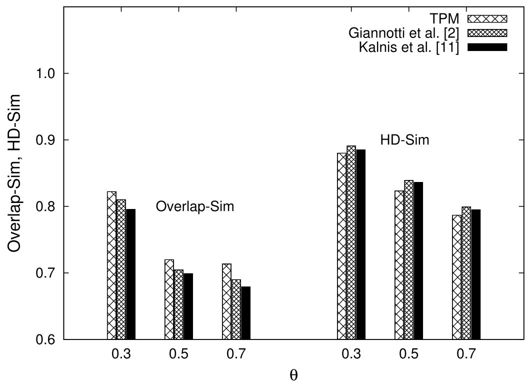
Based on the result of this comparative analysis we selected the two approaches most similar to TPM, that is Giannotti et al. [2] and Kalnis et al. [10], to evaluate their performances, assessing accuracy and quality of dense regions and trajectory patterns, by using the validation methodology proposed in Section 4. The performance indices are evaluated for $\epsilon = 0.01$ and different combinations of θ (outlier regions) and ρ (noise degree) values.

For what concerns the evaluation of the detected dense regions we used the *Accuracy* and *F-measure* metrics. In particular, according to the results shown in Section 6 (see Figs. 8(a) and 8(b)), since the Accuracy and F-measure metrics remain pretty constant with θ (while they vary with ρ), we set $\theta = 0.7$ and varied ρ from 0.1 to 0.5. Accuracy and F-measure values of the three algorithms are plotted in Fig. 12(a), on the left and right side of the chart respectively. In particular, we can notice that TPM achieves a slightly higher accuracy than contestant algorithms, and the trend is more evident for larger noise degree. This result is in accordance to the analysis presented in Table 2: since TPM detects actual dense regions as they emerge from real data without relying on static subdivisions of the spatial area, it achieves high accuracy since it is able to catch regions of any shape and this also makes it more robust to noise. For the same reason, TPM outperforms the other approaches also in terms of F-measure, as shown in the right side of Fig. 12(a). Again, this improvement is more evident when the noise degree increases. In fact, the algorithm correctly clusters the GPS points belonging to the same area, identifying correctly the target dense regions and missing, thus, a very few percentage of them.

We also compared the different algorithms in terms of the quality of the patterns extracted. We validate the trajectory patterns extracted by evaluating their similarity to the target trajectory patterns obtained from the synthetic dataset. To this purpose we used the *Overlap Similarity* and *Haversine Similarity* metrics, as shown in Fig. 12(b). Specifically, TPM is more accurate than contestant approaches in terms of overlap similarity, as the extracted patterns are highly similar to the target ones, even with a high number of outlier regions (i.e., $\theta \geq 0.5$) the quality maintains values over 0.7. However, the chart shows that in terms of haversine similarity TPM performs slightly worse than related approaches since the metric evaluates the similarity by accounting only the distances among centroid points in the regions. Consequently, even if the centroid points of each pair of regions are very close, nothing can be said about the overlap of the two regions. Therefore, we cannot catch the real similarity among detected and target patterns that instead the overlap similarity metrics does.



(a) Dense regions quality.



(b) Trajectory patterns quality.

Fig. 12. Comparative analysis of dense regions quality ($\epsilon = 0.01, \theta = 0.7$, vs ρ) and trajectory patterns quality ($\epsilon = 0.1, \rho = 0.1$, vs θ).

7. Conclusion

In this paper we presented a novel algorithm, TPM, that implements the DSPM approach to detect movement patterns from large-scale socio-geographic data, obtained through the trajectories followed by mobile devices. The approach consists of two key phases: (i) finding the more densely passed through regions in a given geographical area, and (ii) then extracting trajectory patterns from those regions in the form of sequential patterns.

A second contribution of our work is a validation methodology for assessing accuracy and quality of dense regions and trajectory patterns. Precisely, we proposed new trajectory similarity measures to evaluate the quality of the extracted patterns that, differently from the approaches in literature, are specifically tailored for trajectories among regions and not just GPS points. Thus, they are effective in providing the real geographic overlap among the regions involved in the mined trajectory patterns. A detailed evaluation performed by exploiting such a validation process proves the accuracy and quality of the TPM in terms of both dense regions and trajectory patterns discovered, even if data are affected by a significant noise degree.

As future work, we will extend TPM by adopting automatic parameter setting criteria, to improve the dense region discovery step of the algorithm. Furthermore, we plan to apply it on new urban computing applications, i.e., prediction of future locations of vehicles driving in urban areas, route recommendations for drivers that have to reach a given place, prediction of traffic congestion patterns.

References

[1] Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: Concepts, methodologies, and applications, ACM TIST 5 (3) (2014) 1–55.
[2] F. Giannotti, M. Nanni, F. Pinelli, D. Pedreschi, Trajectory pattern mining, in: 13th ACM SIGKDD, 2007.
[3] G. Yavas, D. Katsaros, O. Ulusoy, Y. Manolopoulos, A data mining approach for location prediction in mobile environments, Data Knowl. Eng. 54 (2) (2005) 121–146.
[4] J. Yang, M. Hu, TrajPattern: Mining sequential patterns from imprecise trajectories of mobile objects, in: 10th EDBT, 2006, pp. 664–681.
[5] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, D.W. Cheung, Mining, indexing, and querying historical spatiotemporal data, in: 10th ACM SIGKDD, 2004, pp. 236–245.
[6] H. Cao, N. Mamoulis, D.W. Cheung, Mining frequent spatio-temporal sequential patterns, in: 5th IEEE ICDM, 2005, pp. 82–89.
[7] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti, WhereNext: a location predictor on trajectory pattern mining, in: 15th ACM SIGKDD, 2009, pp. 637–646.
[8] E. Cesario, C. Comito, D. Talia, A comprehensive validation methodology for trajectory pattern mining of GPS Data, in: 2nd IEEE DataCom, 2016, pp. 819–826.
[9] H. Cao, N. Mamoulis, D.W. Cheung, Discovery of periodic patterns in spatiotemporal sequences, IEEE TKDE 19 (4) (2007) 453–467.
[10] P. Kalnis, N. Mamoulis, S. Bakiras, On discovering moving clusters in spatio-temporal data, in: 9th SSTD, 2005, pp. 364–381.
[11] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, R. Trasarti, Unveiling the complexity of human mobility by querying and mining massive trajectory data, VLDB J. 20 (5) (2011) 695–719.
[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, 1996, pp. 226–231.
[13] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: VLDB, 1994, pp. 487–499.
[14] J. Yuan, Y. Zheng, X. Xie, G. Sun, Driving with knowledge from the physical world, in: 17th ACM SIGKDD, 2011, pp. 316–324.
[15] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, Y. Huang, T-drive: Driving directions based on taxi trajectories, in: GIS'10, pp. 99–108.
[16] R. Jonkery, G. De Leve, J. Van Der Velde, A. Volgenant, Rounding symmetric travelingsalesman problems with an asymmetric assignment problem, in: Operations Research, 1980, pp. 623–627.
[17] F. Soong, A. Rosenberg, On the use of instantaneous and transitional spectral information in speaker recognition, IEEE Trans. Acoust. 36 (1988) 871–879.
[18] R. Agrawal, K. Lin, H.S. Sawhney, K. Shim, Fast similarity search in the presence of noise, scaling and translation in time-series databases, in: VLDB, 1995, pp. 490–501.
[19] B. Bollobas, G. Das, D. Gunopulos, H. Mannila, TimeSeries similarity problems and well-separated geometric sets, in: 13th ACM SCG, 1997, pp. 454–456.

[20] T. Bozkaya, N. Yazdani, M. Ozsoyoglu, O. Glu, Matching and indexing sequences of different lengths, in: CIKM, 1997, pp. 128–135.
[21] L. Chen, R. Ng, On the marriage of lp-norms and edit distance, in: VLDB, 2004, pp. 792–803.
[22] L. Chen, M.T. Özsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: ACM SIGMOD, 2005, pp. 491–502.
[23] Y.-T. Zheng, Z.-J. Zha, T.-S. Chua, Mining travel patterns from geotagged photos, ACM TIST 3 (3) (2012) 1–18.
[24] J.J.-C. Ying, E.H.-C. Lu, W.-C. Lee, T.-C. Weng, V.S. Tseng, Mining user similarity from semantic trajectories, in: 2nd ACM SIGSPATIAL LBSN Workshop, 2010, pp. 19–26.
[25] E.J. Keogh, M.J. Pazzani, Scaling up dynamic time warping for datamining applications, in: ACM SIGKDD, 2000, pp. 285–289.
[26] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, ACM SIGMOD Rec. 23 (2) (1994) 419–429.
[27] http://en.wikipedia.org/wiki/Haversine_formula.
[28] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, Mining sequences with temporal annotations, in: ACM SAC, 2006, pp. 593–597.