A Sliding Window Approach for Discovering Dense Areas in Trajectory Streams

Corrado Loglisci and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro"
via Orabona, 4 - 70126 Bari - Italy
{corrado.loglisci,donato.malerba}@uniba.it

Abstract. The task of discovering dense areas aims at detecting regions with high concentration of moving objects from trajectory data and it plays an important role in the development of systems related to traffic and transport management. The problem has been investigated under two main assumptions: the representation of the trajectories in the euclidean space and the characterization of the dense areas as pre-defined cells. However, the adoption of mobile devices enables the generation of trajectories in a streaming style, which adds another degree of complexity to the problem. This is an issue not yet addressed in the literature. We propose a computational solution aiming at detecting dense areas from trajectory streams in a network. Our proposal adopts a sliding window strategy which enables the discovery of two types of dense areas, one based on spatial closeness, the other one based on temporal proximity. Experiments are conducted on trajectory streams generated by vehicular objects in a real-world network.

1 Introduction

The recent adoption of the mobile and ubiquitous technologies has enabled the generation of massive data about moving objects. The development of advanced applications has greatly benefited from this new source of information. For instance, applications for monitoring fleets of vehicles have been conceived thanks to the possibility to collect positions recorded by GPS-equipped devices installed on the moving vehicles. Typically, the generated data are trajectories corresponding to sequences of latitude and longitude positions each associated with the time-stamp when the object had that position. Analysing trajectories remains a widely investigated research line due to its impact on prominent fields such as intelligent traffic management and urban transportation design.

A challenging problem related to the analysis of the trajectories is the identification of areas (or regions) with high density. An area is dense if the number of moving objects it contains is above some threshold or if it exhibits a concentration of objects greater than expected. Usually, it is characterized by its spatial location and the associated time-interval.

Most of the existing works (e.g.,[3, 4]) follow a database-inspired approach based on density queries. There, the main problem is to effectively compute answers and it is solved through the simplification of the queries or the integration of human-specified criteria. Common to these works there is the representation of the space in form of a grid which forces the dense areas to be fixed-size cells. In real-world applications, dense areas do not have a pre-defined form and each area can have a size and shape different from the others. This is typical in spaces structured as road networks where the areas have a so irregular form that it would be difficult to discover dense roads in pre-defined cells.

In this paper, we propose an approach to discover dense areas from trajectories of objects moving in a road network. The network-based representation has a two-fold purpose. First, the consideration of some intrinsic features of space (such as, the presence of buildings and infrastructures) which constraint the movements of the objects, while, in the usual euclidean representation the objects can move freely without particular limitation. Second, the possibility of dealing with the problem at a lower level of granularity through the discovery of dense segments (of the roads) which permit to built areas with any shape and size. Strictly connected with the spatial dimension, we have to take into account the temporal component. Indeed, ignoring it can make the spatial location alone not very useful. We propose to consider time both as dimension of analysis, since the movement and speed of the objects are function of time, and as information associated to a dense area, since some roads can be particularly dense in some hours of day only.

In the real-world applications, we cannot forget the streaming activity of the GPS devices to record the repeated movements of the objects which result in trajectories generated as unbounded sequences of positions. This raises new challenges about the storage, acquisition and analysis that make most of the existing works on dense areas difficult to apply and ineffective. Based on these considerations, we argue that handling trajectory streams becomes necessary, especially in urban contexts where technologies enabled to process trajectories and detect dense areas in real-time can be of support in several practical activities.

To face the streaming activity there are basically two alternatives. First, collecting the trajectories according to a *snapshot* setting which models the positions of the objects recorded in a time-stamp. Second, adopting *sliding windows* which collect positions recorded in periods of time and enable the analysis of the data comprised in a window in the meanwhile another window being created. In this work we follow the second approach since the first one requires that the positions are regularly recorded over time which is an assumption that could not be guaranteed in the moving objects. The second approach instead appears appropriate to model a trajectory in its natural conception of sequence of positions in a time-interval. Coherently to what above illustrated, windows turn out to be particularly suitable to take into account time both as physical measure (with velocity and space) in the process of dense area discovery and as annotation in the representation of those areas. In this work, a sliding window approach is adopted to discover two types of dense areas, one based on the spatial closeness, the other one based on the temporal proximity. In particular, overlapping windows are generated to continuously monitor road segments while a technique, which combines an ad-hoc querying mechanism and rule matching, detects those dense. The rest of the paper is organized as follows. In the Section 2 we overview the related literature. The scientific problem is formalized in the Section 3, while the Section 4 describes the computational solution. Experiments are described in the section 5. Conclusions in the Section 6 close the paper.

2 Related Work

The problem of discovering dense areas was originally faced through a purely spatial dimension and later under a spatio-temporal perspective where moving objects are considered. Two main approaches can be identified: *i)* density-based clustering methods, and *ii)* density querying on grid-based space.

Although dense areas have been investigated with clustering algorithms, the original problem of the clustering is quite different. In these algorithms, dense areas are determined as those geographic regions which have particular spatial properties and which exhibit a particular concentration of objects. Also, they often work on some assumptions about the data distribution, which is anyway difficult to estimate a-priori. Wang et al. [8] study the problem through spatial and multidimensional domains, and the proposed solution permits to generate hierarchical statistical information from spatial data. A static graph representation of the road network has been considered in [9] where three different clustering techniques (partitioning, density-based and hierarchical) work on a network-based distance notion. The dynamic features of the road network have been instead studied in the research line on the moving objects. Chen et al. [1] define an effective clustering approach which works in road networks: the technique first identifies cluster units and then creates different kinds of clusters based on the units by means of a split-and-merge technique. The network features are exploited to reduce the search space and avoid unnecessary computation of network distance. A quite similar method is implemented in [5] through an algorithm which discovers dense areas from cluster units. The latter are generated as groups of moving objects on the basis of the locations and moving patterns. A different perspective is provided in [6] where clustering is performed on sub-trajectories. The idea is that movements can be similar only in segment of the whose trajectories. The approach first partitions the data into line segments, then group them with a density-based clustering.

The grid-based representation for dense areas is not new and it was originally used in [3] in association with the density queries. The original space is modelled in the euclidean framework and partitioned into fixed-size cells to generate cells efficiently. The accuracy loss, introduced by the pre-defined partitioning, is mitigated in the work of Jensen [4].

In the literature of trajectory streams, most of the works are based on clustering. Costa et al [2] propose an effective and accurate solution to find similarities in trajectories modelled with the Fast Fourier Transform. The approach adopts a window-based mechanism finalized to merge newly created clusters with those oldest. A slightly different problem is the grouping of objects moving together [7]. The method first performs a grouping step at each snapshot of the streamed trajectories, and then completes an intersection operation among the groups previously created. The integration of micro-groups of objects and smart intersection operations make that method efficient. However, in both these research lines no paper focused on the discovery of dense areas can be enumerated, so this work can be considered as the first attempt.

3 Basics and Problem Definition

Before formally defining the problem we intend to solve, some definitions are necessary. Consider $T : \{t_1, t_2, \dots, t_k, \dots,\}$ be the discrete time axis, where $t_1, t_2, \dots, t_k$ are time-points equally spaced, $M : \{o_1, o_2, \dots, o_i, \dots, o_m\}$ be the finite set of unique identifiers of the moving objects, $p^i_k \in \mathbb{R}^2$ is the latitude and longitude position of the i^{th} object at the time-point k^{th}. A trajectory stream S_i is the unbounded sequence of positions associated to the object i^{th}, $S_i = \{p^i_1, p^i_2, \dots, p^i_k, \dots\}$. The time-points of recording of the positions are not necessarily equally spaced.

A road network is modelled as a directed graph $G(V, E)$, where V is the set of vertices representing road intersections and terminal points, and E is the set of edges representing road segments each connecting two vertices. A road segment is defined by a unique identifier and the latitude and longitude positions of the terminal points p_s, p_e. For simplicity, we assume that the speed of an object in a road segment is constant. Given the positions p^i_h, p^i_k $(t_k > t_h)$, we can compute the speed of an object in a segment. While, given the speed, the position p^i_h and the length of a segment, we can compute the time-point t_k $(t_k > t_h)$ when the object is expected to be out of the segment (afterwards, t_{out}), where p^i_k in t_k denotes the position of the object in correspondence to the terminal point of the segment.

In this work, dense segments and areas are detected from the trajectory streams.

Definition 1 (Dense Segment). *A segment $\sigma \in E$ is dense if the following conditions hold:*

- *there exists a set of objects $D \subseteq M$ s.t. $\forall o_i \in D$: $t^i_{out} < t^i_k$, t^i_{out} is the time in which we expected the object o_i is out from the segment. This condition is true when o_i is still in the segment at the time t^i_k and has position p^i_k;*
- *$|D| > \delta_{min}$, δ_{min} a user-defined minimum density threshold*

A dense segment σ is identified by the tuple: $\langle p_s, p_e, D, t_0, t_{max} \rangle$, where $t_{max} = \arg\max_{o_i \in D} t^i_{out}$ $(t_0 < t_{max})$, so $[t_0, t_{max}]$ establishes the interval in which σ is dense.

Intuitively, a segment is dense when there are objects which have not come out from the segment within the expected time which is specific for each object. This notion is coherent with the queueing process.

A collection of dense segments can form a dense area on the basis of the spatial closeness. It would depict a traffic jam where neighbouring segments exhibit density at the same time (the objects are fully stopped). More formally,

Definition 2 (Spatial Closeness-based Dense Area). *A collection of dense segments Σ is a dense area of type SC if the following conditions hold:*

- $\forall \sigma', \sigma'' \in \Sigma : d(\sigma', \sigma'') \leq L$, *$L$ is a user-defined parameter, d is a network-based distance;*
- $\forall \sigma', \sigma'' \in \Sigma : t_0' = t_0''$ *(t_0', t_0'' are the lower bounds the time-intervals of the tuple of σ' and σ'' respectively);*
- $\forall \sigma', \sigma'' \in \Sigma : D' \cap D'' = \oslash$

A collection of dense segments can form a dense area on the basis of the spatial closeness and temporal proximity. It would depict a traffic congestion where near segments exhibit density at consecutive times (the objects flow slowly). More formally,

Definition 3 (Spatial Closeness and Temporal Proximity-based Dense Area). *A collection of dense segments Σ is a dense area of type TP if the following conditions hold:*

- $\forall \sigma', \sigma'' \in \Sigma : t_0'' > t_0'$;
- $\forall \sigma', \sigma'' \in \Sigma : t_0'' - t_0' < \Omega$, *$\Omega$ user-defined parameter;*
- $\forall \sigma', \sigma'' \in \Sigma : D' \cap D'' \neq \oslash$.

The threshold Ω defines the temporal proximity within which the dense areas of type TP can be seek.

Given the set of moving objects M, the trajectory streams in the form of S_i, the road network G, the problem at the hand is *To Detect* dense areas as defined in the Definitions 3 and 4. The parameters δ_{min}, L and Ω are used to filter out meaningless segments and areas.

4 Dense Areas from Trajectory Streams

4.1 Overview

To illustrate the computational solution we need the notion of trajectory window. Let $S_i, S_{i+1}, \ldots S_n$ be a stream of trajectories of objects $o_i, o_{i+1}, \ldots, o_n$. The trajectory window $[W]_w^u$ of width ω is a sub-sequence of the streams $S_i, S_{i+1}, \ldots S_n$ and comprises the positions observed in the time-interval $[t_u, t_w]$, where ω is a user-defined parameter.

The streaming setting requires that the phase of analysis is not postponed to that of acquisition of data. It often precludes the possibility of performing pre-processing operations on the whole dataset which are instead adopted by techniques which do not work on data stream (e.g.[5]). We propose a solution which performs simultaneously a step of data acquisition from trajectories stream and a step of analysis to discover dense areas.

More precisely, the first step collects data (about the positions and segments) continuously coming from moving objects and fills partially overlapping windows.

Each window is overlapped with (some of) the windows that precede and follow it, so it contains part of the positions which are contained in the windows created before and after. For instance, the windows $[W]_{w}^{u}$, $[W]_{s}^{r}$, $[W]_{n}^{m}$ are created in this order so that the orders $t_r < t_w$ and $t_m < t_s$ hold, while the order between t_w and t_m cannot be established a-priori since it depends from the rate (defined by the user) with which the windows are generated.

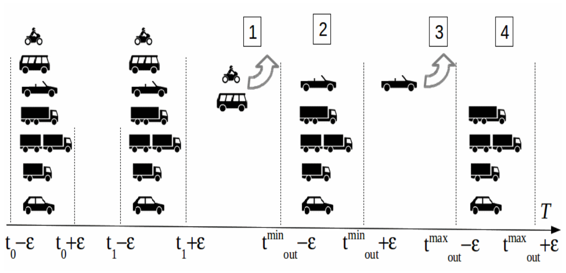
The organization of the windows by partial overlapping allows to share information of the moving objects among different windows and therefore it reduces the possibility of information loss when analyzing the movements.

The second step proceeds at the level of single windows and at the level of sequences of windows. More precisely, it processes one window at time and finds dense areas of type SC from that window: for each segment, the positions buffered in one window are analyzed in order to determine whether that segment is dense. The segments identified as dense in a window will be used to create a dense area of type SC. In the meanwhile one window is processed, the next window is acquired. When the algorithm turns to process the next window, it stores *i)* the dense segments discovered in the past window(s) and *ii)* the expected times t_{out} computed in the past window(s). So, when a new window is processed, the algorithm has the results of the sequence of the windows since there analyzed. The expected times (computed before) are used to complete, in the current window, the search (started in the past window(s)) of dense segments. The discovery of dense areas of type TP is performed on the dense segments obtained from the sequences of windows. Only sequences of windows whose overall width does not exceed Ω are considered.

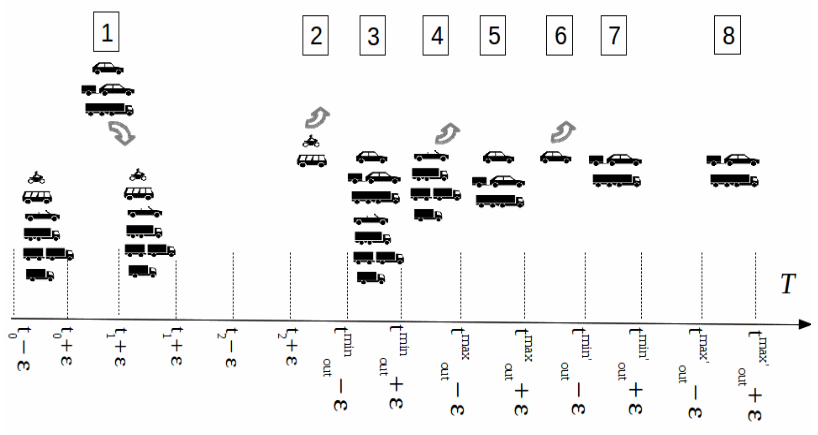
4.2 Detection of Dense Segments

The method for discovering dense segments resorts to the queueing process according to which the delay of an object (with respect to the expected time) to leave the queue can be attributed to the increase of the number of the objects present in the queue or to a demand for resources greater than expected. We argue that the increase of the moving objects is the most reasonable motivation of the delay in the road networks.

The algorithm combines two activities. The first one performs the monitoring of the segments by means of queries submitted in correspondence of the time-points in which it is expected that the objects will leave the segment. Once a query has been completed, the second activity checks whether the monitored objects are still in the segment: if it is so, they will contribute to make that segment dense. More precisely, the check mechanism adopts heuristics (modelled in the form of *if-then* rules) and performs a matching operation between (the information associated to) each segment and the antecedent parts of the rules. The consequent parts indicate whether the segment is dense or not. The possibility to query the trajectory streams only in correspondence of the expected time-points makes our approach different from the methods in which the queries have to be periodically (and often more frequently) submitted ([4]).


1
2
3
4
T
$t_0-\varepsilon$
$t_0+\varepsilon$
$t_1-\varepsilon$
$t_1+\varepsilon$
$t^{min}_{out}-\varepsilon$
$t^{min}_{out}+\varepsilon$
$t^{max}_{out}-\varepsilon$
$t^{max}_{out}+\varepsilon$

(a)


1
2
3
4
5
6
7
8
T
$t_0-\varepsilon$
$t_0+\varepsilon$
$t_1-\varepsilon$
$t_1+\varepsilon$
$t_2-\varepsilon$
$t_2+\varepsilon$
$t^{min}_{out}-\varepsilon$
$t^{min}_{out}+\varepsilon$
$t^{max}_{out}-\varepsilon$
$t^{max}_{out}+\varepsilon$
$t^{min'}_{out}-\varepsilon$
$t^{min'}_{out}+\varepsilon$
$t^{max'}_{out}-\varepsilon$
$t^{max'}_{out}+\varepsilon$

(b)

Fig. 1. Estimation of the times t_{out} and application of the check mechanism.

As anticipated in the section 3, the estimation of times requires the speed of each object, its position inside the segment and length of the segment. In particular, the speed of the object is determined when starting to process the current window by submitting two queries (in correspondence of two time-points t_0, t_1) that retrieve the objects to be monitored and their positions, while, the road network G provides the length of the segment. However, the objects can move with different velocities and can leave the segment at different times. This may be determinant in the formation of dense segments and, in order to take into account it, we consider both fast objects and those slow when discovering dense segments. In this sense, we estimate two kinds of expected times, one, denoted as t_{out}^{min}, which indicates the time within which the fast objects should be out, the other one, denoted as t_{out}^{max}, which indicates the time within which the slow objects should be out. More precisely, t_{out}^{min} refers to the slowest object out of the fastest objects, while t_{out}^{max} is the speed of the slowest object in the set of monitored objects of that segment. Intuitively, we expect that the those fastest will be out before (within t_{out}^{min}), while the slowest ones will do it after (but within t_{out}^{max}). If this does not happen, we could have an high concentration of objects in that segment. We encode this idea into *if-then* rules which work as follows. Once computed t_{out}^{min} and t_{out}^{max}, two queries are submitted (in the same order of t_{out}^{min},t_{out}^{max}) in correspondence of these time-points to check whether the fastest objects (among those monitored) have left before t_{out}^{min} and the slowest objects (among those monitored) have left before t_{out}^{max}: if the number of the objects in common ($|D|$) to the time t_{out}^{min} and to the time t_{out}^{max} is higher than the minimum threshold δ_{min}, then the segment is dense.

Notice that, since it would be unrealistic that the positions of all objects are recorded at the same times, the queries submitted with the value of the time-point could not generate results, so we consider time-intervals with fixed radius ϵ centred on t_{out}^{min} and t_{out}^{max}, and on the first two time-points t_0, t_1. As illustrated in the Figure 1a, the fast objects go out before $t_{out}^{min} + \epsilon$ (square 1), while other fast objects remain, although we expect their absence (square 2). Next, in correspondence of $t_{out}^{max} + \epsilon$ (square 3), we expect that the remaining fast objects and those slow go out: if, even after $t_{out}^{max} + \epsilon$, there are still objects (whose size exceeds δ_{min}), then the segment is dense (square 4).

However, in while identifying the objects to be monitored (queries on t_0 and t_1), new objects enter the segment. The check mechanism could provide wrong results since those objects are not comprised in the initially retrieved set. The solution consists of i) integrating additional *if-then* rules able to recognize the insertion of new objects and ii) adapting the monitoring process since we expect to retrieve the new objects in the next queries. More specifically, an additional query on t_2 ($t_1 < t_2$) is submitted in order to compute the speed and expected time-points (afterwards, $t_{out}^{min'}$ and $t_{out}^{max'}$) for the new objects. The check mechanism is updated in order to monitor the new objects in correspondence of *i)* the expected time-points $t_{out}^{min'}$ and $t_{out}^{max'}$, *ii)* the expected time-points t_{out}^{min} and t_{out}^{max} initially estimated. The values of the new time-points establish the new order of the queries to be submitted and, dependently on this order, different rules will be matched. The time-point t_0 and the longest estimated time-point define (t_{out}^{max} or $t_{out}^{max'}$) the time-interval of the tuple of the segment.

For brevity, here we detail (Figure 1b) only the case in which the time-points $t_{out}^{min'}$, $t_{out}^{max'}$ come after t_{out}^{min} and t_{out}^{max}. As illustrated in the Figure 1b, the presence of new objects in t_1, with respect to t_0, forces to run another query in order to complete the calculation of the speed for the new objects (square 1). In the meanwhile, the algorithm estimates t_{out}^{min} and t_{out}^{max} which will be the time-points that the check mechanism will use for the next queries. After the query t_2, also $t_{out}^{min'}$ and $t_{out}^{max'}$ are estimated, so we have two additional queries to be included in the check mechanism: the new order of the queries will comply with the order of the expected times $t_{out}^{min} < t_{out}^{max} < t_{out}^{min'} < t_{out}^{max'}$. Then, the algorithm checks that the new objects are present in t_{out}^{min} and t_{out}^{max} (squares 2-5), and that all have left before $t_{out}^{max'}$ (square 8). In t_{out}^{min} and t_{out}^{max} we observe that the initially monitored objects leave the segment (squares 2,4) and that other objects remain there (squares 3,5). The latter should leave before $t_{out}^{max'}$, while instead the check mechanism detects that from $t_{out}^{min'}$ to $t_{out}^{max'}$ no object has left (squares 6-8): if these numerically exceed δ_{min}, the segment is dense.

4.3 Detection of Dense Areas

Dense areas are generated with at least two dense segments which meet the Definitions 2 and 3. This provides some hints on the technique to use. The algorithm of the areas of type SC is performed once a window is processed and operates on the segments already detected and dense areas which are being generated during the process. Each segment can be associated to only one area if *i)* its distance from the segments (already added to that area) is lower than the parameter L and *ii)* it has no object in common with those segments. If the examined segment is added to none of the areas, it will be considered as seed for a new area. Finally, we will have a collection of segments which begin to be dense at the time t_0, which is common to all segments of that area.

The dense areas of type TP involve the dense segments obtained from a sequence of overlapping windows. Considering sequences of windows has a two-fold advantage: *i)* mitigating the effect that pre-defined windows (as in the case of areas SC) can have on the final results, and *ii)* enabling the discovery of dense segments on consecutive windows. In particular, when the values of t_{out}^{min} and t_{out}^{max} exceed the last time-point t_w of the window $[W]_w^u$, we need to adapt the check mechanism in order to submit the corresponding queries later, specifically when the window which contains t_{out}^{min} and t_{out}^{max} will be processed. The algorithm acts once a number of windows having a total width less than Ω has been analyzed and it executes two selection operations. The first one takes one segment from each window provided that they are temporally ordered: for instance, given σ',σ'', σ''' detected in the windows $[W]_w^u$,$[W]_s^r$, $[W]_n^m$ respectively, σ',σ'', σ''' are taken if the order $t_0' < t_0'' < t_0'''$ holds. The second one refines the previous operation by selecting the segments which have moving objects in common.

Searching dense segments which share objects over a finite sequence of windows introduces implicitly a spatial neighbourhood in which those objects move around. This allows us to attribute the formation of the congestion areas to specific objects.

For both types of areas, the notion of the distance between two segments σ', σ'' is based on the given network structure and corresponds to the usual distance of the shortest path in the graphs. More precisely, it is the minimum summation of the single distances associated to the intermediate segments and to the segments σ', σ''. Each single distance is the spatial distance between the terminal points of the segment.

5 Experiments

The computational solution was tested on a real-world dataset which comprised trajectories produced by taxies moving in the city of Bejing for one week (02/02/2008-08/02/2008) [1] The original dataset was modified by removing the positions which could not be associated to segments of the network and by adding new data in order to maintain the order of magnitude of the size. Modifications

[1] http://research.microsoft.com/apps/pubs/default.aspx?id=138035

concern the insertions of i) positions in all segments for all objects, ii) positions in the segments of the centre of the city, iii) positions for the objects which have an higher (and lower) number of positions, iv) positions for all objects in peak hours, v) positions in the roads having many segments. Totally, we have almost 37000 roads, 89900 segments, 98600 moving objects. The dataset is poured in a traditional relational DBMS and converted into a data stream by taking the temporal order of the input data as the order of streaming. The road network of the city of Bejing [2] has 141,380 segments and 106,579 vertices from which we derive the lengths of the segments.

Experiments are performed to test the influence of the input threshold δ_{min} on the final dense areas. We report results on the dense segments and dense areas discovered in each day and in the peak hours, namely 7:00-9:00pm, 12:00-14:00pm, 18:00-20:00pm (Figures 2). The thresholds and parameters are set as follows: $\delta_{min} = 3, 5, 7$, $L = 1.5km$, $\Omega = 30mins$, $|t_1 - t_0| = 1min$, $|t_2 - t_1| = 1min$, $\epsilon = 10secs$, $\omega = 5mins$, the rate of generation of the windows equal to $2.5mins$. As to the dense segments, a quite expected behaviour is that the number of dense segments decreases as the minimal threshold increases, and this is common to all days for all hours. An analysis done on the basis of the hours may provide indications of social nature: on the week-end, a greater flow can be observed only in the time-slot 18:00-20:00, while on the weekdays the highest numbers of segments is produced from 12:00-14:00. As to the dense areas, the results basically follow the behaviour of the dense segments. In particular, by analyzing the time-slots 7:00-9:00 and 18:00-20:00, numerous sets of SC-areas are detected when we have more dense segments, while the greatest sets of SC-areas are discovered in correspondence of the highest number of dense segments (12:00-14:00). Also expected it is the result that associates great sets of SC-areas to great sets of segments when δ_{min}=3: in that case, the dense segments can be so numerous to be close to each other with the result to form more dense areas. Numerous collections of TP-areas are obtained when we have many dense segments in all configurations (δ_{min}=3,5,7). Indeed, particularly for the time-slots 12:00-14:00 (on the weekdays) and 18:00-20:00 (on the week-end), the expected high number of objects can justify a slow movement and therefore the raising of TP-areas in all configurations.

We evaluated the final results through a quantitative measure which estimates the capacity of the approach to detect segments which have an high concentration of objects with respect to the neighbourhood. More formally, $\Theta(\sigma) = \frac{\sum_{j=1\ldots m}(\sigma - \sigma_j)}{(m*1)+avg\sigma}$, where σ_j are non-dense segments close to σ in a diameter of 100m[3], m is the number of non-dense close segments, $avg\sigma$ is the average number of objects per segment. The value of $\Theta(\sigma)$ would tend towards 0 with an equal distribution of the objects among the segment σ and its neighbours, while when there is a strong concentration of the objects in the segment σ, the value of $\Theta(\sigma)$ tends towards 1. The Table 1 reports the mean of the values of Θ on all

[2] http://www.openstreetmap.org/

[3] This value has been specifically computed of the road network of the city of Bejing by considering also that two parallel roads can be close within 100m.

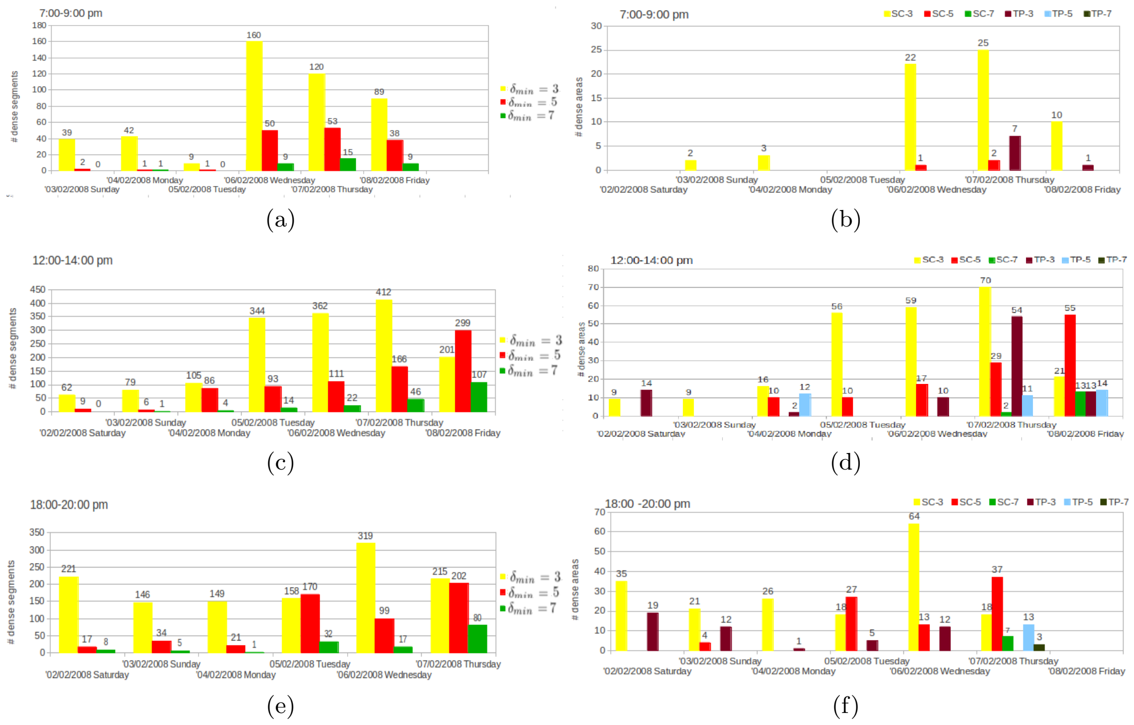


Fig. 2. Dense segments when σ=3,5,7 at peak hours 7:00-9:00pm, 12:00-14:00pm, 18:00-20:00pm (a,c,e). Dense areas of type SC and TP discovered when σ=3(SC-3,TP-3), σ=5(SC-5,TP-5), σ=7(SC-7,TP-7) (b,d,f) .

dense segments discovered in each day: we observe the better performances in the days from Tuesday to Friday. This is encouraging because it points out the ability of the method to recognize correctly high concentrations of objects when the dense segments grow and maintain the robustness with respect to the noise.

6 Conclusions

In this paper we investigated the task of discovering dense areas in a stream of trajectory defined in a road network. The proposed approach adopts a sliding window strategy to detect dense segments and use them to form dense area. It combines knowledge in the form of *if-then* rules and a querying mechanism to retrieve information about the segments from the stream. Experiments prove the applicability to a real-world road network As future direction, we plan to extend the rule base with additional conditions and validate the approach on road networks and trajectory data where ground truth is available.

Table 1. Evaluation on the dense segments by day.

	Θ
02/02/2008 Saturday	0.52
03/02/2008 Sunday	0.67
04/02/2008 Monday	0.41
05/02/2008 Tuesday	0.54
06/02/2008 Wednesday	0.72
07/02/2008 Thursday	0.68
08/02/2008 Friday	0.64

Acknowledgments. This work is in partial fulfilment of the PRIN 2009 Project "Learning Techniques in Relational Domains and Their Applications" funded by the Italian Ministry of University and Research (MIUR).

References

1. J. Chen, C. Lai, X. Meng, J. Xu, and H. Hu. Clustering moving objects in spatial networks. In K. Ramamohanarao, P. R. Krishna, M. K. Mohania, and E. Nantajeewarawat, editors, *DASFAA*, volume 4443 of *Lecture Notes in Computer Science*, pages 611–623. Springer, 2007.
2. G. Costa, G. Manco, and E. Masciari. Effectively grouping trajectory streams. In A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras, editors, *NFMCP*, volume 7765 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2012.
3. M. Hadjieleftheriou, G. Kollios, D. Gunopulos, and V. J. Tsotras. On-line discovery of dense areas in spatio-temporal databases. In T. Hadzilacos, Y. Manolopoulos, J. F. Roddick, and Y. Theodoridis, editors, *SSTD*, volume 2750 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2003.
4. C. S. Jensen, D. Lin, B. C. Ooi, and R. Zhang. Effective density queries on continuouslymoving objects. In L. Liu, A. Reuter, K.-Y. Whang, and J. Zhang, editors, *ICDE*, page 71. IEEE Computer Society, 2006.
5. C. Lai, L. Wang, J. Chen, X. Meng, and K. Zeitouni. Effective density queries for moving objects in road networks. In G. Dong, X. Lin, W. Wang, Y. Yang, and J. X. Yu, editors, *APWeb/WAIM*, volume 4505 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007.
6. J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In C. Y. Chan, B. C. Ooi, and A. Zhou, editors, *SIGMOD Conference*, pages 593–604. ACM, 2007.
7. L. A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng. On discovery of traveling companions from streaming trajectories. In A. Kementsietsidis and M. A. V. Salles, editors, *ICDE*, pages 186–197, 2012.
8. W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *VLDB*, pages 186–195. Morgan Kaufmann, 1997.
9. M. L. Yiu and N. Mamoulis. Clustering objects on a spatial network. In G. Weikum, A. C. König, and S. Deßloch, editors, *SIGMOD Conference*, pages 443–454. ACM, 2004.