Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 9 (2015) 164 – 184

Transportation Research Procedia

www.elsevier.com/locate/procedia

21st International Symposium on Transportation and Traffic Theory

Spatial and Temporal Characterization of Travel Patterns in a Traffic Network Using Vehicle Trajectories

Jiwon Kim[a], Hani S. Mahmassani[b]*

[a]*The University of Queensland, Brisbane St Lucia, QLD 4072, Australia*
[b]*Transportation Center Northwestern University, 600 Foster Street, Evanston, IL 60208, USA*

Abstract

This paper presents a trajectory clustering method to discover spatial and temporal travel patterns in a traffic network. The study focuses on identifying spatially distinct traffic flow groups using trajectory clustering and investigating temporal traffic patterns of each spatial group. The main contribution of this paper is the development of a systematic framework for clustering and classifying vehicle trajectory data, which does not require a pre-processing step known as *map-matching* and directly applies to trajectory data without requiring the information on the underlying road network. The framework consists of four steps: similarity measurement, trajectory clustering, generation of cluster representative subsequences, and trajectory classification. First, we propose the use of the Longest Common Subsequence (LCS) between two vehicle trajectories as their similarity measure, assuming that the extent to which vehicles' routes overlap indicates the level of closeness and relatedness as well as potential interactions between these vehicles. We then extend a density-based clustering algorithm, DBSCAN, to incorporate the LCS-based distance in our trajectory clustering problem. The output of the proposed clustering approach is a few spatially distinct traffic stream clusters, which together provide an informative and succinct representation of major network traffic streams. Next, we introduce the notion of cluster representative subsequence (CRS), which reflects dense road segments shared by trajectories belonging to a given traffic stream cluster, and present the procedure of generating a set of CRSs by merging the pairwise LCSs via hierarchical agglomerative clustering. The CRSs are then used in the trajectory classification step to measure the similarity between a new trajectory and a cluster. The proposed framework is demonstrated using actual vehicle trajectory data collected from New York City, USA. A simple experiment was performed to illustrate the use of the proposed spatial traffic stream clustering in application areas such as network-level traffic flow pattern analysis and travel time reliability analysis.

Peer-review under responsibility of the Scientific Committee of ISTTT21
Keywords: trajectory clustering; spatial and temporal traffic patterns; traffic stream clusters; trajectory similarity; longest common subsequence

* Corresponding author. Tel.: +1-847-491-2276; fax: +1-847-491-3090.
E-mail address: masmah@northwestern.edu

Peer-review under responsibility of the Scientific Committee of ISTTT21
doi:10.1016/j.trpro.2015.07.010

1. Introduction

Vehicle trajectories are traffic data obtained from tracking individual vehicles' movements over time, where each trajectory is a sequence of consecutive geo-referenced coordinates and the corresponding timestamps, typically recorded every few seconds. Compared to traditional traffic data obtained from fixed loop detectors, vehicle trajectories provide much richer information—not only information on user-centric travel experiences (e.g, speed profile of each vehicle along its journey and travel times experienced by individual vehicles) but also system-wide spatial and temporal trip and traffic patterns (e.g., origin-destination pairs, routes, and bottlenecks). These are considered as by far the most comprehensive traffic data available in transportation systems analysis. With the ubiquitous use of GPS devices (e.g., smartphones and in-vehicle navigation systems) and RFID tags (e.g., electronic fare cards and electronic toll collection systems), trajectory data of moving objects in traffic networks are becoming increasingly available and this opens up new opportunities for performing more sophisticated and comprehensive analyses for the planning, design, and management of transportation systems. Great challenges remain, however, due to the massive size of such data and the complexity of dealing with both the spatial and temporal dimensions. Data mining has thus become an important part of transportation research and practice, calling for the in-depth investigation and development of domain-specific analysis methods and tools. This paper addresses these needs by presenting a systematic framework and detailed methods for processing, clustering, and classifying vehicle trajectories, with a primary application in characterizing spatial and temporal travel patterns that can be used in network-level traffic flow analysis and travel time reliability analysis. In particular, this study considers the problem of clustering trajectories as a means to discover major traffic flow groups across the network, which may then be used in analyzing the temporal evolution of traffic states for each region more effectively. As the main contribution of this paper, we present the step-by-step procedure for clustering vehicle trajectories based on their spatial characteristics (i.e., route similarity) and show how network space can be partitioned into and represented by a few major traffic stream clusters. The proposed trajectory clustering method is demonstrated using actual vehicle trajectory data collected from New York City, USA.

1.1. Related Work

Trajectory clustering has been gaining increasing interest in recent years and several clustering methods tailored specifically for trajectories have been proposed. Clustering analysis in general requires two essential components, namely, similarity measure and clustering algorithm. Trajectory clustering approaches proposed in the literature also largely depend on the choice of the combination of these two. Nanni and Pedreschi (2006) describes an approach to clustering trajectories using the average distance between objects' positions and density-based clustering algorithm OPTICS (Ankerst et al., 1999). They further extend their OPTICS-based trajectory clustering approach to identify a specific time interval where trajectories are clustered in the most meaningful way and focus on those interesting time intervals to improve the quality of clustering results. Rinzivillo et al. (2008) proposed progressive clustering that allows successive application of several different distance functions and gradual refinement of clustering results obtained from the previous steps. They suggest a number of intuitive distance functions such as 'common source', 'common destination', 'route similarity', and 'time steps'. Lee at al. (2007) proposed a partition-and-group framework for clustering trajectories (TRACLUS), which partitions a trajectory into a set of line segments and discovers common sub-trajectories by grouping similar line segments.

Earlier studies in trajectory clustering mainly considered trajectories of general moving objects, which can move freely in Euclidean space. Recognizing that in the case of vehicles the movements are actually constrained by the underlying road networks, another line of studies have emerged focusing on the clustering of network-constrained trajectory data. Kharrat et al. (2008) proposed a two-step clustering algorithm (NETSCAN), where the first step clusters road segments into a set of dense paths and the second step groups trajectories according to their similarity to each dense path. Roh and Hwang (2010) proposed a distance measure that reflects road-network proximity, computed using the shortest path calculation, and applied in their trajectory clustering algorithm (NNCluster). Han et al.(2012) proposed a road network aware approach for clustering trajectories (NEAT) and Mahrsi and Rossi (2013) proposed a graph-based approach to clustering network-constrained trajectory data.

While it is attractive to consider the network constraint and take advantage of the underlying network representation, it requires a pre-processing step for mapping each point in a trajectory to a road network, known as *map-matching*, and that process itself requires non-trivial effort. Above all, it is highly desirable to work purely on the available vehicle trajectory data, without having to prepare the associated network topology. In this paper, we focus on the clustering approach that does not require the information on the underlying road network. Although we do not attempt to map trajectories to known road segments, we take into account the fact that vehicle movements are constrained by the road network in defining a similarity measure. We consider the fact that trajectories on the same route are completely overlapping (at a macroscopic scale) due to the network constraint and this allows us to use the algorithm for finding the Longest Common Subsequence (LCS) between two sequences. We measure how much two trajectories overlap each other to define their level of closeness, relatedness, and connectivity. Conceptually, this LCS-based similarity measure lies between the Euclidean distance that is used for free moving objects and the graph-based distance (e.g., shortest path) that is used for fully network-mapped objects. Below, we describe a number of characteristics for vehicle trajectory data that are assumed in this study.

1.2. Vehicle Trajectory Data

A vehicle trajectory, denoted by *Tr,* is a time-ordered sequence or time-series of 3-tuples (x, y, t) representing the x and y (longitude and latitude) coordinates of a vehicle at time t. Let $Tr_i = (p_1^i, p_2^i, \cdots, p_N^i)$ denote the trajectory of vehicle i, where $p_n^i = (x, y, t)_n^i$ is the n^{th} point of the sequence ($n = 1, \cdots, N$). Given two vehicle trajectories $Tr_i = (p_1^i, p_2^i, \cdots, p_N^i)$ and $Tr_j = (p_1^j, p_2^j, \cdots, p_M^j)$ with size N and M, respectively, the trajectory data addressed in this study have the following characteristics.

- *Different lengths*: trajectories may have different lengths in terms of the number of data points within each trajectory (e.g., N and M can be different for $Tr_i = (p_1^i, p_2^i, \cdots, p_N^i)$ and $Tr_j = (p_1^j, p_2^j, \cdots, p_M^j)$).
- *Different and uneven sampling rates*: data sampling rates are not necessarily identical across trajectories (e.g., $p_1^i, p_2^i, \cdots, p_N^i$ are recorded every 1 second while $p_1^j, p_2^j, \cdots, p_M^j$ are recorded every 2 seconds). Moreover, the time interval between two consecutive points within the same trajectory can also vary (e.g., 1 second between p_1^i and p_2^i while 2 seconds between p_2^i and p_3^i). In this paper, we assume that data points in a vehicle trajectory are recorded at sufficiently short time intervals (e.g., recorded at every 2 to 3 seconds) such that the distance between two consecutive points can be reasonably approximated by a straight line connecting the two points.
- *Different directions and regions*: In the context of vehicle movement, the direction and space region of each trajectory are important conditions for similarity of trajectories. For instance, two trajectories moving in opposite directions should be considered as different trajectories despite their close proximity to each other. Also, trajectories showing similar movements in different regions (e.g., left-turning vehicles at different intersections) should be considered as different objects despite their similar shapes.
- *Road network constraint*: Unlike trajectories of free moving objects in Euclidean space such as animals and hurricanes (Lee et al., 2007), vehicle trajectories are constrained by the underlying road network. This suggests that data points can appear only in the space along the roads and attempting to use similarity measures designed for general line segments or vectors (e.g., Euclidean distance and angle distance) may be inefficient.

2. Methodology

2.1. Framework

This study proposes a systemic framework and detailed implementation methods for clustering and classifying vehicle trajectory data. Fig. 1 shows the overall framework for analyzing trajectories addressed in this paper.

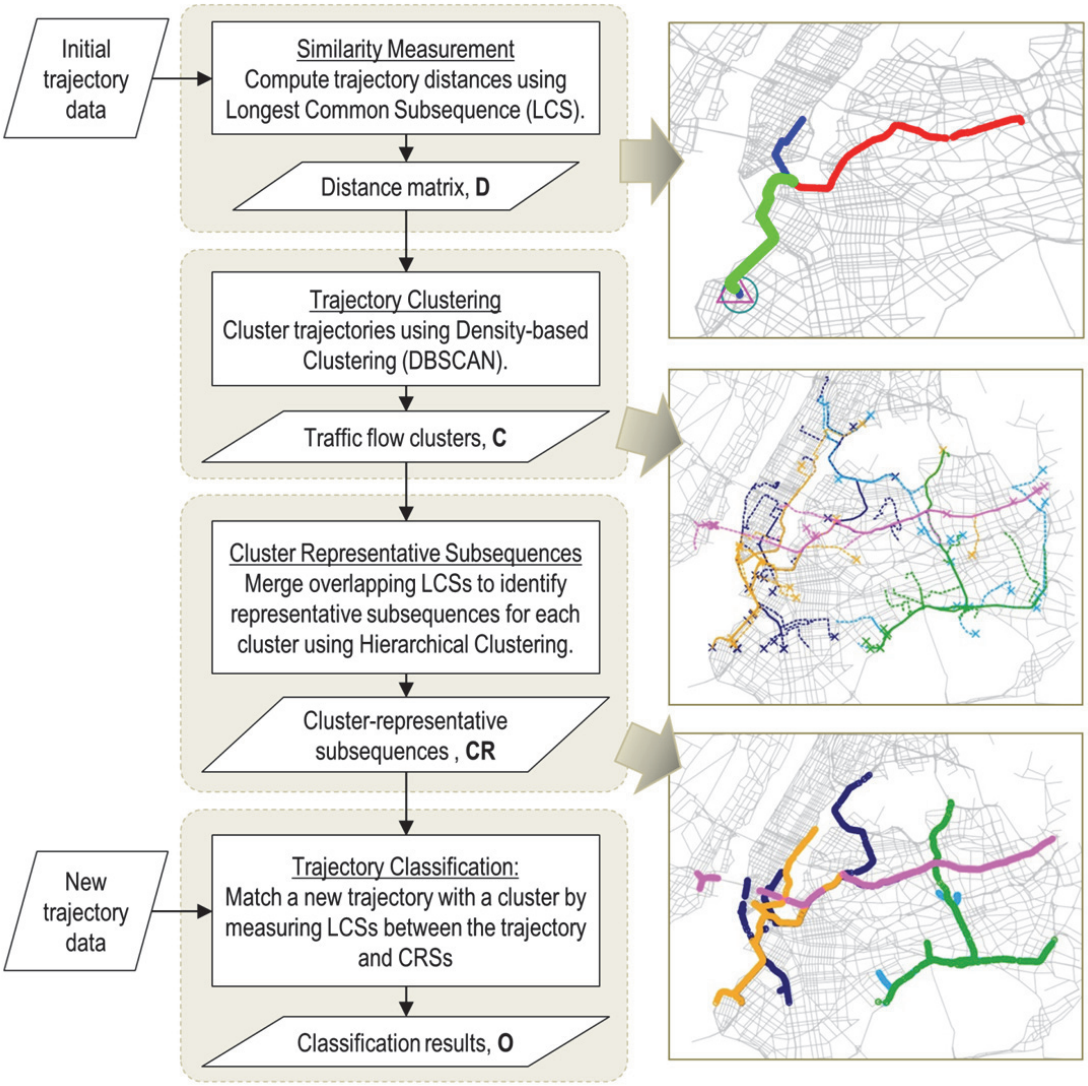

Fig. 1. Proposed framework for clustering and classifying vehicle trajectories.

The procedure follows four steps: Similarity Measurement, Trajectory Clustering, Cluster Representative Subsequence Generation, and Trajectory Classification. A brief description for each step is provided as follows:

- *Similarity Measurement:* Given the initial trajectory dataset, the first step is to measure similarities of trajectories. This paper proposes the use of the Longest Common Subsequence (LCS) as a similarity measure, assuming that the extent to which vehicles' routes overlap indicates the level of closeness and relatedness as well as potential interactions between these vehicles. Time is relaxed in this study. The output of this step of a distance matrix, denoted by **D**, containing pairwise trajectory distances.
- *Trajectory Clustering:* In this step, trajectories are grouped into similar traffic flow groups based on the LCS-based distance measure mentioned above, which captures (time-relaxed) spatial similarity. The goal of this

clustering is to identify time-invariant major traffic streams in the network, which will then be used for subsequent temporal analyses. For instance, instead of looking at temporal evolution of traffic states at thousands of detector points, we can focus on a small number of major traffic streams and their associated routes—it could be a few road segments or the entire stretch of a highway depending on the shape of the flow cluster. For the clustering method, we use DBSCAN, a density-based clustering algorithm. The output of this step is a set of traffic stream clusters, denoted by **C**.

- *Cluster Representative Subsequence Generation*: this step is responsible for defining cluster representatives that will be used for the trajectory classification step. In order to effectively represent a large number of trajectories in a given cluster, we define the notion of *cluster-representative subsequences* (CRS), which can be viewed as the union of all pairwise LCSs of the trajectories within the cluster. Given the LCS of each trajectory pair, which is available from the first step, we merge a large number of overlapping LCSs using a hierarchical (agglomerative) clustering method to identify a small number of distinct common subsequences that are shared by member trajectories of the given cluster. The output of this step is a set of cluster-specific CRS collections, denoted by **CR**.
- *Trajectory Classification*: Whenever a new trajectory is obtained, we can classify it to one of the existing clusters by evaluating the similarity between the trajectory and a cluster. Given the set of CRSs for each cluster, we check if the trajectory shares any of these CRSs and assign to the cluster if it has a significant portion that is overlapping those CRSs. The output of this step is the assignment results (cluster labels) for tested trajectories, denoted by **O**.

Next, we present detailed implementation methods for each step.

2.2. Similarity Measure for Vehicle Trajectories

In the context of traffic analysis, the similarity of vehicle trajectories can be defined in various ways (e.g., origin-based, destination-based, OD pair-based, route-based, etc. (Rinzivillo et al., 2008)). In this study, we focus on route similarity. We measure the similarity of two trajectories by investigating their spatial road use patterns such as whether these two vehicles have travelled similar routes and used similar road segments regardless of their departure times or travel speeds. One way to identify this is to check if there is an overlap between two trajectories. Since vehicle movements are constrained by the road network, portions of trajectories will be completely overlapping (at a macroscopic scale) if they travelled the same road section. To find overlapping portions of trajectories, we use the Longest Common Subsequence (LCS) algorithm. The LCS is an algorithm for finding the longest common subsequence of two sequences and originates in the field of string matching, where two strings with different lengths are given to find a set of characters that appear left-to-right, not necessarily consecutively, in both strings (Bergroth et al., 2000). The LCS has been widely used in time-series clustering as an alternative to traditional Euclidean distance to deal with time series with different lengths as well as to allow distortions in the time axis—the LCS model can "match two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing some elements to be unmatched" (Vlachos et al., 2002). Several studies have also adapted the LCS method to the context of trajectory clustering (Hermes et al., 2009; Vlachos et al., 2002). In these applications, the similarity between two time series (or trajectories) is expressed as the number of matched elements between two sequences. Both the size of the longest common subsequence and the subsequence itself can be obtained using a dynamic programming algorithm in $O(n^2)$ time (Bergroth et al., 2000).

The first step to define the LCS-based similarity measure for vehicle trajectories is to determine a rule for matching points in two sequences. Given two points $p_n^i = (x, y, t)_n^i$ and $p_m^j = (x, y, t)_m^j$, one could determine whether these two are matched or not in a number of different ways. One way is to consider spatial proximity only and another way is to consider both spatial and temporal proximity. In this paper, we will focus on the first case, where two points are considered to match (overlap) if their spatial similarity obtained by the following function *simPnt* is greater than zero:

$$simPnt(p_n^i, p_m^j) = \begin{cases} 0 & \text{if } dist(p_n^i, p_m^j) > \delta \\ 1 - \dfrac{dist(p_n^i, p_m^j)}{\delta} & \text{otherwise} \end{cases}, \tag{1}$$

In Eq. (1), $dist(p_n^i, p_m^j)$ represents the spatial distance between p_n^i and p_m^j, which can be defined as *Great-circle distance* between two geographic latitude and longitude coordinates, and δ is a parameter specifying the maximum allowable distance for two points to match. The function $simPnt(p_n^i, p_m^j)$ returns a normalized measure of similarity between two points p_n^i and p_m^j and takes a value between 0 and 1, where the value becomes zero when $dist(p_n^i, p_m^j)$ is greater than or equal to δ. Using *simPnt*, we identify matched elements in two sequences.

Next, we present a dynamic programing model to compute the LCS alignment between two trajectories $Tr_i = (p_1^i, p_2^i, \cdots, p_N^i)$ and $Tr_j = (p_1^j, p_2^j, \cdots, p_M^j)$. Let $Tr_{i(n)}$ denote the first *n* points of trajectory Tr_i such that $Tr_{i(n)} = (p_1^i, p_2^i, \cdots, p_n^i)$, where $n<N$. To obtain the LCS, we define a recursive function $scoreLCS(Tr_{i(n)}, Tr_{j(m)})$ as follows:

$$scoreLCS(Tr_{i(n)}, Tr_{j(m)}) = \begin{cases} 0 & \text{if } n = 0 \text{ or } m = 0 \\ \max \begin{pmatrix} scoreLCS(Tr_{i(n-1)}, Tr_{j(m-1)}) \\ \quad + simPnt(p_n^i, p_m^j), \\ scoreLCS(Tr_{i(n)}, Tr_{j(m-1)}), \\ scoreLCS(Tr_{i(n-1)}, Tr_{j(m)}) \end{pmatrix} & \text{otherwise} \end{cases}. \tag{2}$$

The function $scoreLCS(Tr_{i(n)}, Tr_{j(m)})$ returns the similarity score of the best alignment between Tr_i and Tr_j up to their n^{th} and m^{th} points, respectively, and represents the sum of all the *simPnt* values for the matched elements under the best alignment. After recursively computing $scoreLCS(Tr_{i(n)}, Tr_{j(m)})$ for all $n = 1, \cdots, N$ and $m = 1, \cdots, M$, we can find the LCS alignment between full trajectories Tr_i and Tr_j by identifying point matches (p_n^i, p_m^j) that have contributed to the final cumulative score $scoreLCS(Tr_{i(N)}, Tr_{j(M)})$ through $simPnt(p_n^i, p_m^j)$ in $scoreLCS(Tr_{i(n)}, Tr_{j(m)}) = scoreLCS(Tr_{i(n-1)}, Tr_{j(m-1)}) + simPnt(p_n^i, p_m^j)$ in Eq. (2).

The LCS model defined in Eq. (2) can be viewed as a modified version of the existing models (Hermes et al., 2009; Vlachos et al., 2002) in that it not only finds the longest common subsequence in terms of the number of matches but also maximizes the similarity score of the matched points. For a given point p_n^i in trajectory *i*, there

could be multiple matches in trajectory j that satisfy matching thresholds. For instance, we may have p_m^j that is a match with p_n^i producing $simPnt(p_n^i, p_m^j) > 0$, but we may have another point p_{m+1}^j that is also a match with p_n^i producing $simPnt(p_n^i, p_{m+1}^j) > 0$. In such cases, the existing models do not further search for a better match once a match is found. It is, however, necessary to find the closest pair among all valid matches to ensure that the final LCS can reflect the overlapping route between two trajectories as closely as possible. Fig. 2 illustrates this situation. The connection lines between points (both solid grey and dotted green) represent the valid matches, which produce *simPnt* > 0, and the numbers on the lines show the estimated *simPnt* values for the corresponding matches. In Fig.2 (a), P_2^i in Tr_i is matched with both P_2^j and P_3^j in Tr_j, but the most similar one (*simPnt*=0.8) is chosen as its final match, represented by thick dotted connection line. Similarly P_3^i is matched with P_4^j (*simPnt*=0.9) among three valid matches, P_3^j, P_4^j, and P_5^j. Let us consider the case where the similarity between P_3^i and P_4^j is 0.5 instead of 0.9 as shown in Fig.2 (b). Now, the second matching point changes from (P_3^i, P_4^j) to (P_3^i, P_5^j) as the largest similarity measure is found between P_3^i and P_5^j (*simPnt*=0.6). The proposed LCS model in Eq. (2) takes into account such a difference, illustrated in Fig.2 (a) and (b), in finding the final LCS.

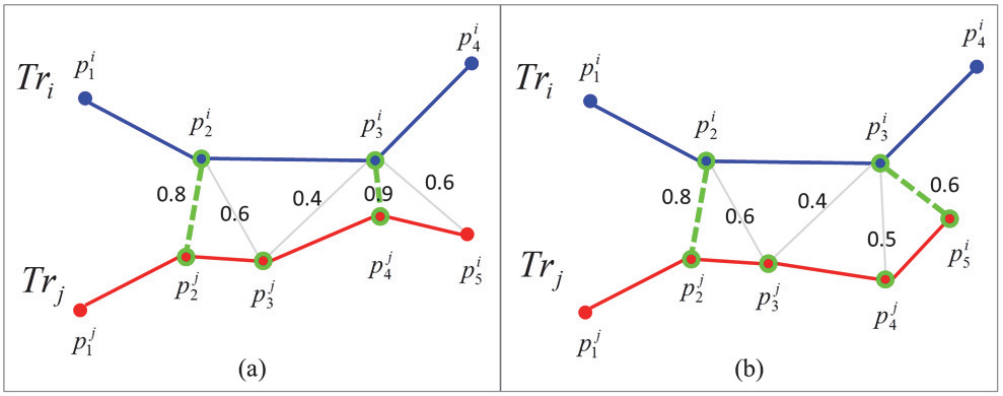

Fig. 2. Modified LCS model that consider the quality of the matches in the LCS; the clostest point match is selected as the element of LCS by measuring the similarity between each pair of valid point matches.

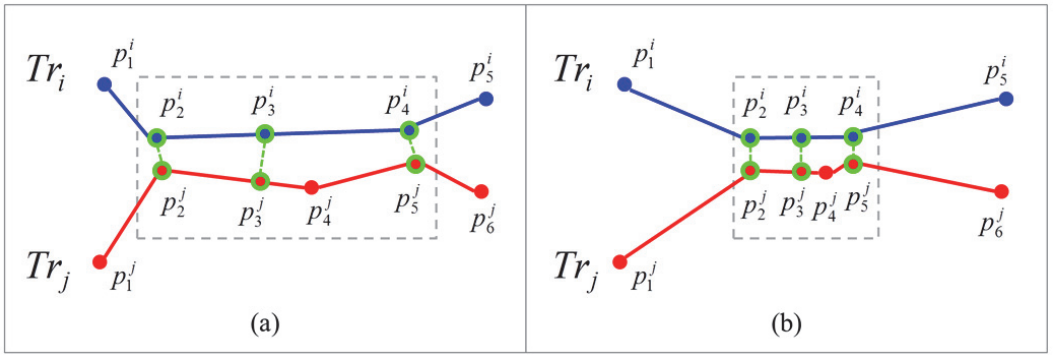

Fig. 3. Different similarity situations that produce the same LCS size: the size of the LCS between Tr_i and Tr_j is 3 for both (a) and (b), but the actual length covered by the LCS is different.

Once the LCS is identified, the next step is to define a LCS-based similarity measure suitable for vehicle trajectories. In the previous studies, LCS-based similarity measures are expressed in terms of the ratio of the size of a LCS (i.e., the number of matched elements) to the sizes of its original sequences (Hermes et al., 2009; Vlachos et al., 2002). The problem with determining the similarity measure based on the size of a LCS arises when trajectories have different and uneven sampling rates. In such cases, LCSs with the same size can have different similarity indications. Fig. 3 illustrates this situation using a simple example. Suppose that we wish to express the similarity between two trajectories $Tr_i = (p_1^i, p_2^i, p_3^i, p_4^i, p_5^i)$ and $Tr_j = (p_1^j, p_2^j, p_3^j, p_4^j, p_5^j, p_6^j)$ using their LCS. Given three matched point pairs (p_2^i, p_2^j), (p_3^i, p_3^j), and (p_4^i, p_5^j), the size of the LCS is 3 in both (a) and (b) in Fig.3 with the resulting LCSs of (p_2^i, p_3^i, p_4^i) for Tr_i and (p_2^j, p_3^j, p_5^j) for Tr_j, respectively. When it comes to the extent to which two trajectories overlap, however, the trajectories in Fig.3 (a) share their routes much more strongly than those in Fig.3 (b), suggesting that two trajectories in Fig.3 (a) should be considered to be more similar to each other than those in Fig.3 (b).

A reasonable approach to incorporating this aspect is to use the actual length (travel distance) of the LCS, instead of its size, in expressing the similarity. Let $L_{i,j} = (q_1, \cdots, q_H)$ be the identified LCS between Tr_i and Tr_j, where $q_1, \cdots, q_H$ represent the matched data points and H is the number of matches, $H \le \min(N, M)$. Since point matches are not exact but rather approximate (based on distance threshold δ), $L_{i,j} = (q_1, \cdots, q_H)$ are not uniquely determined; in fact, the points in $L_{i,j}$ can be from either Tr_i or Tr_j. To preserve consistency across $q_1, \cdots, q_H$ and ensure that $L_{i,j}$ is also a valid trajectory, we only consider two cases where $q_1, \cdots, q_H$ are taken either entirely from Tr_i or entirely from Tr_j. We use superscript notations $L_{i,j}^{(i)}$ and $L_{i,j}^{(j)}$ when it is necessary to indicate which trajectory the LCS components are taken from. For instance, the LCS shown in Fig. 3 can be represented as either $L_{i,j} = L_{i,j}^{(i)} = (p_2^i, p_3^i, p_4^i)$ or $L_{i,j} = L_{i,j}^{(j)} = (p_2^j, p_3^j, p_5^j)$.

Defining a function $lenSeq(X)$ that calculates the length (travel distance) of trajectory X, the actual length of the LCS is obtained by summing all the distance measures between consecutive data points in $L_{i,j}$ as follows:

$$lenSeq(L_{i,j}) = \sum_{h=1}^{H-1} dist(q_h, q_{h+1}) \tag{3}$$

Given the measured length of the LCS, we now define the similarity measure between two trajectories Tr_i and Tr_j using the following formula:

$$simSeq(Tr_i, Tr_j) = \begin{cases} 0 & \text{if } \min(lenSeq(L_{i,j}^{(i)}), lenSeq(L_{i,j}^{(j)})) < \gamma \\ \dfrac{lenSeq(L_{i,j}^{(i)})}{lenSeq(Tr_i)} & \text{if } lenSeq(Tr_i) < lenSeq(Tr_j) \\ \dfrac{lenSeq(L_{i,j}^{(j)})}{lenSeq(Tr_j)} & \text{otherwise} \end{cases}, \tag{4}$$

where γ is a parameter describing the minimum length for the LCS between two trajectories to be a valid LCS. The

similarity function *simSeq* takes a value between 0 and 1, where the similarity becomes zero if the length of the LCS —either $L_{i,j}^{(i)}$ or $L_{i,j}^{(j)}$ —is shorter than the minimum LCS length (γ). Eq.(4) indicates that the similarity between two trajectories is expressed as the portion of the shorter trajectory covered by the LCS. If one trajectory is completely contained in the other trajectory, the LCS length between two is equal to the length of the shorter trajectory and their similarity measure becomes one.

Given the similarity measure defined above, the distance between two trajectories Tr_i and Tr_j can be obtained by:

$$distSeq(Tr_i, Tr_j) = 1 - simSeq(Tr_i, Tr_j) \tag{5}$$

As an example, if we set γ=300m and obtain $distSeq(Tr_i, Tr_j) = 0.6$, it means that two trajectories overlap for a route longer than 300m and the overlapping portion covers 40% of the total travel distance of the shorter trajectory, i.e., $simSeq(Tr_i, Tr_j) = 0.4$.

2.3. Clustering Vehicle Trajectories

This section describes the second step of the framework in Fig. 1. Given a set of vehicle trajectory data **TR** and the distance matrix **D** containing the pairwise distance *distSeq*(Tr_i, Tr_j) for all pairs of trajectories in **TR**, this step groups similar trajectories into a cluster to form a few distinct trajectory clusters. We use DBSCAN (Density Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) as the main clustering algorithm. DBSCAN relies on a density-based notion of clusters—a cluster is a maximal set of density-connected points—and the number of clusters are determined based on the data provided depending on two input parameters: *Eps* and *MinPts*, where the former specifies the maximum radius of the neighborhood and the latter represents the minimum number of neighbors for a point to be a *core* point, distinguished from a *noise* (Ester et al., 1996). In order to apply this concept to clustering trajectories, we extend the notations and definitions for points to the case of vehicle trajectories as follows.

Definition 1: (*ε-neighbourhood* of a trajectory) The *ε-neighbourhood* of a trajectory Tr_i, denoted by $N_\varepsilon(Tr_i)$, is defined by $N_\varepsilon(Tr_i) = \{Tr_j \in \mathbf{TR} \mid distSeq(Tr_i, Tr_j) \le \varepsilon\}$, $0 \le \varepsilon \le 1$.

Definition 2 (*directly density-reachable*): A trajectory Tr_j is *directly density-reachable* from a trajectory Tr_i with respect to ε and *MinTrs* if

1) $Tr_j \in N_\varepsilon(Tr_i)$ and
2) $|N_\varepsilon(Tr_i)| \ge MinTrs$ (core trajectory condition)

Definition 3 (*density-reachable*): A trajectory Tr_j is *density-reachable* from a trajectory Tr_i with respect to ε and *MinTrs* if there is a chain of trajectories $Tr_i, \cdots, Tr_n, Tr_{n+1}, \cdots, Tr_j$ such that Tr_{n+1} is directly density-reachable from Tr_n (so all trajectories on the chain must be core trajectories, with the possible exception of Tr_j).

Definition 4 (*density-connected*): A trajectory Tr_j is *density-connected* to a trajectory Tr_k with respect to ε and *MinTrs* if there is a trajectory Tr_i such that both Tr_j and Tr_k are density-reachable from Tr_i with respect to ε and *MinTrs*.

Given these definitions, DBSCAN produces trajectory clusters, where a cluster is defined as a maximal set of density-connected trajectories with respect to density-reachability. There are three types of trajectories: *core*, *edge*, and *noise*. If a trajectory Tr_i has at least *MinTrs* trajectories in its *ε-neighborhood*, $N_\varepsilon(Tr_i)$, then it becomes a *core*

trajectory and forms a cluster together with all trajectories in $N_\varepsilon(Tr_i)$. If another member of $N_\varepsilon(Tr_i)$, $Tr_j \in N_\varepsilon(Tr_i)$, has a sufficient number of its *ε-neighborhood* trajectories such that $|N_\varepsilon(Tr_j)| \geq MinTrs$, then Tr_j itself becomes a *core* trajectory and expands the cluster further by adding its *ε-neighborhood* to the cluster. If Tr_j is not a core, then it is classified as an *edge* and does not add any other trajectories but itself. Thus, all trajectories in a cluster are either cores or edges. A trajectory that is neither a *core* nor an *edge* is classified as a *noise* and does not belong to any cluster.

With the proposed LCS-based distance measure, the following implications can be drawn from the clustering results.

- All trajectories in a cluster share a significant portion of their routes with at least one other trajectory in that cluster, where the length of any given LCS is longer than γ and its coverage is greater than (1- ε)*100% of the shorter of the associated two trajectories.
- A core trajectory in a cluster shares its route with at least *MinTrs* other trajectories in that cluster (either core or edge) with respect to the LCS-based distance $distSeq \leq \varepsilon$ (or $simSeq \geq 1\text{-}\varepsilon$). Each cluster contains at least one such core trajectory.
- An edge trajectory in a cluster shares its route with less than *MinTrs* other trajectories in that cluster with respect to the LCS-based distance $distSeq \leq \varepsilon$ (or $simSeq \geq 1\text{-}\varepsilon$), but is connected to at least one core trajectory.
- All trajectories within a cluster are connected through a chain of LCSs.
- Any two vehicles associated with the trajectories in a cluster could possibly affect each other—even if they do not directly share their routes—either through forward shockwave propagation or through backward shockwave propagation, assuming that a vehicle could travel its route again any time during the day.

Fig. 4 shows the algorithm used in performing trajectory clustering in this study. The algorithm is based on the version described in (Lee et al., 2007), but only adopting the basic DBSCAN part. The output of this procedure is a set of trajectory clusters $\mathbf{C} = \{C_1, \cdots, C_K\}$, where $C_k = \{Tr_1, \cdots, Tr_{|C_k|}\}$. It is noted that not all trajectories are assigned to a cluster and trajectories that do not belong to any cluster are classified as a noise group.

Trajectory Clustering using DBSCAN		
Input	A set of vehicle trajectories $\mathbf{TR} = \{Tr_1, \cdots, Tr_{\|\mathbf{TR}\|}\}$ Distance matrix **D** DBSCAN parameters ε and *MinTrs*	
Output	A set of trajectory clusters $\mathbf{C} = \{C_1, \cdots, C_K\}$, where $C_k = \{Tr_1, \cdots, Tr_{\|C_k\|}\}$	
Procedure		
1	Set *ClusterID* = 0;	
2	mark all trajectories in **TR** as unvisited;	
3	for each trajectory Tr_i in **TR**,	
4	if Tr_i is not visited,	
5	mark Tr_i as visited;	
6	get $N_\varepsilon(Tr_i)$ using **D**;	
7	if $\|N_\varepsilon(Tr_i)\| < MinTrs$,	
8	mark Tr_i as *noise*;	
9	else,	
10	set *ClusterID* = *ClusterID* + 1;	
11	assign *clusterID* to all trajectories in $N_\varepsilon(Tr_i)$;	
12	insert $N_\varepsilon(Tr_i) - \{Tr_i\}$ into queue *Q*;	
13	call *expandCluster*(*ClusterID*, *Q*, **D**, ε, *MinTrs*);	
14	return **C**.	

```
15      expandCluster(ClusterID, Q, D, ε, MinTrs):
16          while queue Q is not empty, do:
17              remove Tr_j from the front of queue Q;
18              get N_ε(Tr_j) using D;
19                  if | N_ε(Tr_j) | ≥ MinTrs,
20                      for each Tr_k in N_ε(Tr_j),
21                          if Tr_k is not visited or noise,
22                              assign ClusterID to Tr_k;
23                          if Tr_k is not visited,
24                              Insert Tr_k into queue Q;
```

Fig. 4. DBSCAN algorithm applied for clustering vehicle trajectories.

2.4. Generating Cluster Representative Subsequences

This section describes the third step of the framework in Fig. 1. Once trajectory clusters are obtained, the next step is to define certain representative features for each cluster so that a new trajectory can be evaluated and classified. For a given cluster, a certain representative trajectory could be constructed based on the trajectories in the given cluster (Lee et al., 2007) or the underlying road segments could be used if trajectories are mapped to the underlying road networks (Han et al., 2012; Kharrat et al., 2008; Roh and Hwang, 2010). In this paper, we propose another approach based on the LCS between trajectories that were obtained in the first step of the framework. Since we use the LCS as the distance measure for clustering trajectories, each of trajectories in a cluster is involved in at least one LCS with another trajectory in that cluster. We observe that a lot of these pairwise LCSs are actually heavily overlapping and their overall coverage can be represented by a few representative subsequences. As such, we introduce the notion of cluster-representative subsequence (CRS), which can be viewed as the union of all overlapping pairwise LCSs from a given cluster, to define a set of subsequences that effectively represent its member trajectories. Each cluster may have multiple CRSs, and these CRSs will be used for calculating the similarity between a new trajectory and the cluster in the trajectory classification step. In particular, we compute the LCS between a CRS and the new trajectory to measure its similarity using Eq. (4) and assign it to the cluster if the new trajectory shares its route with the given CRS (i.e., *simSeq* > 0, discussed in the next section). Since CRS is used in calculating the LCS, a CRS itself should be a valid trajectory, which neither diverges nor converges. Let us define the operator $\widetilde{\bigcap}$ such that $X \widetilde{\bigcap} Y$ returns the LCS between two sequences X and Y such that $L_{i,j} = Tr_i \widetilde{\bigcap} Tr_j$. Similar to $\widetilde{\bigcap}$, we define the operator $\widetilde{\bigcup}$ to express the procedure of merging two sequences. Given two sequences X and Y, $X \widetilde{\bigcup} Y$ returns a merged sequence that represents the union of X and Y in terms of their coverage. Fig. 5 illustrates the process of generating a CRS from a set of LCSs. Suppose that three trajectories Tr_i, Tr_j, and Tr_k are in the same cluster. We have three LCS from each of three trajectory pairs, $L_{i,j} = Tr_i \widetilde{\bigcap} Tr_j$, $L_{j,k} = Tr_j \widetilde{\bigcap} Tr_k$, and $L_{k,i} = Tr_k \widetilde{\bigcap} Tr_i$, respectively, as shown in Fig.5 (a). We observe that these LCSs are overlapping at segment BC as shown in Fig.5 (b) and can be represented as one long sequence covering segment AD as shown in Fig.5 (c).

Our goal in this step is therefore to convert a large number of redundant short LCSs to a small number of longer CRSs such that all the road segments covered by the LCSs can be fully covered by the set of CRSs. To generate CRSs, we use hierarchical agglomerative clustering, which merges two closest LCSs into one sequence one at a time and produces a set of merged CRSs. When merging two LCSs, the distance between these two is set to be very large if these are diverging or converging, in order to prevent them from merging and ensure the generated CRSs are valid

trajectories. Let CR_k denote a set of CRSs for the k^{th} cluster, C_k, and S_i denote the i^{th} CRS for cluster C_k. Fig. 6 presents the hierarchical clustering algorithm used in producing a set of CRSs, $\mathbf{CR}=\{CR_1,\cdots,CR_K\}$, where $CR_k=\{S_1,\cdots,S_{|CR_k|}\}$, for all clusters $\mathbf{C}=\{C_1,\cdots,C_K\}$.

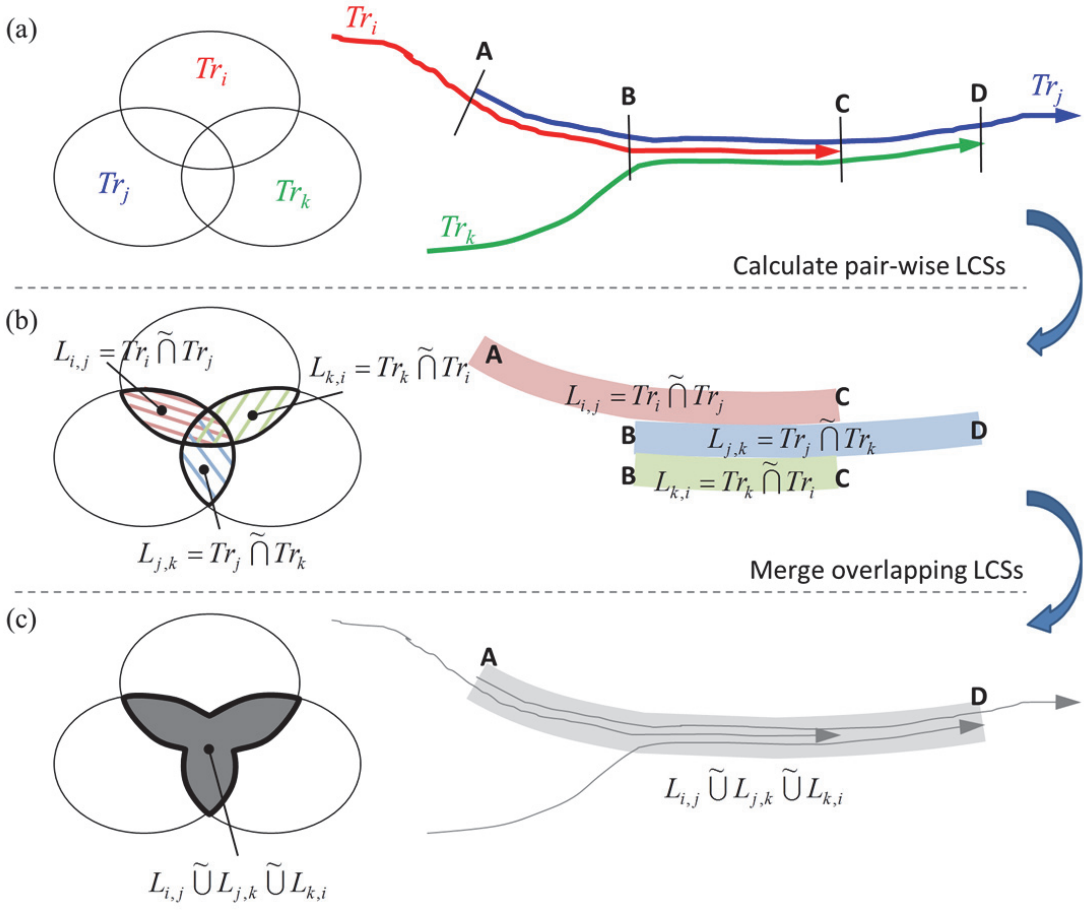

Fig. 5. Process of merging pairwise LCSs and creating cluster-represntative subsequence (CRS).

LCS Merging using Hierarchical Agglomerative Clustering

Input: Trajectory clusters $\mathbf{C} = \{C_1, \cdots, C_K\}$, where $C_k = \{Tr_1, \cdots, Tr_{|C_k|}\}$; LCS database **L**

Output: A set of cluster representative subsequences $\mathbf{CR} = \{CR_1, \cdots, CR_K\}$, where $CR_k = \{S_1, \cdots, S_{|CR_k|}\}$

Procedure

Set $\mathbf{CR} = \{\}$;
for each cluster $C_k \in \mathbf{C}$,
  set $CR_k = \{\}$;
  for each pair of trajectories (Tr_i, Tr_j) in C_k,
    find the associated LCS $L_{i,j} = Tr_i \widetilde{\bigcap} Tr_j$ from **L** and add to CR_k;
  call *mergeLCSs*(CR_k);
  add CR_k to $\mathbf{CR}$;
return $\mathbf{CR}$.

mergeLCSs(CR_k):
  set *MinDist* = 0;
  while *MinDist* < 1, do:
    for each pair of sequences (S_i, S_j) in CR_k, compute their distance:
      if S_i and S_j are on the same line (neither converging nor diverging),

$$distSeq(S_i, S_j) = 1 - \frac{lenSeq(S_i \widetilde{\bigcap} S_j)}{\min(lenSeq(S_i), lenSeq(S_j))};$$

      else,
        $distSeq(S_i, S_j) = 1$;
    find the closest pair of sequences, i.e., let (S_n, S_m) be the closest pair;
    set *MinDist* = $distSeq(S_n, S_m)$;
    merge S_n and S_m into a single long sequence and update S_n, i.e., $S_n = S_n \widetilde{\bigcup} S_m$;
    remove S_m from CR_k;

Fig. 6. Hierarchical agglomertiave clustering algorithm applied for generating cluster-representstive subsequences (CRSs).

2.5. Classifying Vehicle Trajectories

This section describes the last step of the framework in Fig. 1. Consider that we have a new trajectory, denoted by Tr_{new}, that was not in the initial trajectory dataset **TR**. Given the identified traffic stream clusters, $\mathbf{C} = \{C_1, \cdots, C_K\}$, and the associated cluster representatives, $\mathbf{CR} = \{CR_1, \cdots, CR_K\}$, we assign Tr_{new} to a cluster by applying the following rule.

Rule 1: (Trajectory classification) Trajectory Tr_{new} is assigned to cluster C_k if there exists $S_i \in CR_k$ such that $simSeq(Tr_{new}, S_i) > 0$.

This rule indicates that we assign a trajectory to a cluster if the trajectory overlaps with any of the CRSs. The CRSs for a given cluster represent dense road segments shared by the member trajectories—shared by at least two trajectories but in many cases more than two—and thus Rule 1 effectively captures spatial route similarity between the new trajectory and the existing cluster, which is the basis for forming the initial clusters, thereby ensuring within-cluster consistency and preserving cluster characteristics.

3. Experiment

3.1. Data

In this section, we test the proposed clustering and classification approaches using actual vehicle trajectory data. The data are from the Strategic Highway Research Program 2 (SHRP2) project L04 (Mahmassani et al., 2013) and were collected by vehicles equipped with TomTom GPS devices during a two-week period in May 2010 (May 2 – May 17) in New York City, USA. We sampled 1000 vehicle trajectories from May 4 between 6 AM and 8 PM to generate initial clusters.

3.2. Discovering Spatial Clustering Patterns

Similarity Measurement: For spatial clustering, we use *simPnt* in Eq. (1) to determine point matches and find the LCS. In Eq. (1), time information of trajectory data is ignored and only spatial proximity is considered. We set δ = 20m as a matching threshold, i.e., the maximum allowable distance for two points to match. The minimum required length for the LCS, γ, was set to γ=300m. Fig. 7 shows examples of the LCS-based similarity measures. Blue and red lines represent two trajectories, respectively, and the green portion represents their LCS. The start-points of two trajectories are marked as circle and triangle, respectively. Subfigures (a-e) show different situations that yield different similarity values, denoted by *s*: (a) diverging (*s*=0.63), (b) merging (*s*=0.30), (c) fully overlapping (*s*=0.99), (d) merging and diverging (*s*=0.29), and (e) branching (*s*=0.12). In (f), the similarity is zero because there is no LCS found due to their opposite directions.

Trajectory Clustering and CRS Generation: The DBSCAN clustering algorithm in Fig. 4 requires two parameters: ε (the maximum neighborhood distance) and *MinTrs* (the minimum number of required trajectories in a cluster). These parameters need to be specified by a user and there is no specific rule for finding optimal parameter values. Since the underlying clustering structure of data is inherently unknown and we use the cluster analysis to discover it, there is no formal procedure for calibrating the parameters and validating the clustering results, which is the case in many unsupervised learning problems. The choice of the DBSCAN parameters is not completely arbitrary, however, because each of the two parameters has a clear meaning and its role in the clustering process can be intuitively understood. As such, the user can choose the parameters based on domain knowledge and some experiment. Given the LCS-based distance measure *distSeq* (= 1-*simSeq*), the parameter ε controls the required length of the LCS that connects trajectories, specifying how loosely or tightly member trajectories are connected in a cluster. For instance, roughly speaking, ε=0.7 means that trajectories should be overlapped with others for at least 30% of their total travel distances, ε=0.2 means that the overlaps should be at least 80% of their travel distances, and so on. The parameter *MinTrs* controls the number of trajectories that a core trajectory is overlapped with during its journey.

The smaller ε and the larger *MinTrs*, the stricter the clustering criteria are and thus the more difficult it is to form a cluster. If a cluster is formed under the combination of a small ε and a large *MinTrs*, it is likely that core trajectories in that cluster are either (i) relatively long so that they can be overlapped with many other short trajectories along their journeys or (ii) relatively short but passing through busy road segments so that they can be easily overlapped by other trajectories passing through those segments. With an understanding of the roles of ε and *MinTrs*, one can indirectly control the number of final clusters and also their characteristics by finding a balance

between these two parameters. The numbers of clusters produced under the combination of a small ε and a small *MinTrs* and the combination of a large ε and a large *MinTrs* can be similar, but the characteristics of the resulting clusters would be very different. Intuitively, the combination of large ε and *MinTrs* values will produce loosely connected big traffic stream clusters. A cluster obtained under this setting tends to have a large spatial coverage and can provide insight into how far a chain of connected trajectories can reach. On the other hand, the combination of small ε and *MinTrs* values will produce tightly connected (i.e., heavily overlapped) small traffic stream clusters. The clusters produced under this setting tend to be more locally concentrated and can provide information about trajectories that are truly close in space in terms of their route similarity. As such, the choice of the clustering parameters might be based on intended use of the results.

Fig. 8 provides an example of the effects of ε and *MinTrs* on the clustering results. Three cases were tested: (a,b) ε=0.2 and *MinTrs*=10, (c,d) ε=0.4 and *MinTrs*=14, and (e,f) ε=0.9 and *MinTrs*=27. For each row, the left plot shows vehicle trajectories for different clusters (highlighted by different colors) and the right plot shows the associated CRSs. In all three cases, the number of generated clusters is five. In fact, we chose *MinTrs* in a way that the number of clusters remains the same as five for a given ε. We can see that each cluster expands significantly by relaxing (i.e., increasing the value of) ε (from top to bottom in Fig. 8) as discussed above. We use the clusters obtained under the combination of ε=0.4 and *MinTrs*=14 for the subsequent analyses.

Trajectory Classification: Given the five clusters obtained under ε=0.4 and *MinTrs*=14, we tested 1000 new trajectories for classification. Fig. 9 shows the clustering and classification results by cluster. The direction of each flow stream cluster is depicted by red arrows. The five traffic stream clusters in Fig. 9 (a)-(e) show distinct characteristics in terms of directions and geographical locations. Cluster 1 and Cluster 2 are formed on the roadways connecting Manhattan and Western Long Island, representing traffic streams from the southwest to the northeast and those from the northeast to the southwest, respectively. Cluster 3 and Cluster 4 are formed on the roadways adjacent to John F. Kennedy (JFK) International Airport, which is located at the intersection of Interstate Highway 678 and Belt Parkway in Queens, New York City, representing traffic streams to and from the airport, respectively. Cluster 5 represents traffic streams on the Long Island Expressway (LIE) in the westbound direction toward Manhattan. All the five clusters effectively capture major traffic streams in the area, revealing the most commonly used roads and routes. By plotting these five clusters together on the network as shown in Fig.9 (f), we observe that the clusters heavily overlap one another, which suggest that the clustering patterns in Fig.9 (a)-(e) could not have been easily identified using visual inspection or spatial clustering based geographical locations. The proposed flow-based clustering approach, thus, provides an effective way to decompose the overall network traffic flows into different traffic stream layers, which can be used to perform traffic analysis at a cluster-level, adding a new dimension to the traditional traffic analysis conducted at link, path, OD, or network-levels.

3.3. Comparing Temporal Characteristics of Spatial Flow Clusters

In this section, we demonstrate how we can use the identified spatial traffic stream clusters in discovering patterns in other dimensions such as daily traffic patterns in a more effective way. As mentioned earlier, the rationale behind the proposed spatial flow clustering is to reduce spatial dimensions in performing overall spatio-temporal pattern analysis in traffic flow and reliability analysis and discover temporal patterns more effectively by focusing on a particular spatial traffic stream group. The experiment demonstrated here is motivated by our previous work concerning identifying different daily scenarios in performing scenario-based travel time reliability analysis (Kim et al., 2013; Kim and Mahmassani, 2014; Mahmassani et al., 2013). We are interested in identifying different daily traffic patterns to construct a few representative daily scenarios to investigate possible variations in travel time and estimate the overall long-term travel time distribution. With this application in mind, we designed a simple experiment to investigate how the spatial clusters identified in this study can help in discovering temporal network traffic patterns.

First, we sampled five sets of 1000 new vehicle trajectories from each of 16 days (i.e., a total of 80 samples with size 1000). For each 1000 daily trajectory sample, we classified them into five clusters. For each cluster, we divided the sampling time window (6AM-8PM) into a series of 1-hr intervals and measured the number of vehicle trajectories for each time interval based on their departure times. As a result, for each cluster, we have 80 time series of cluster size, each of which represents the number of vehicles in the given traffic stream cluster over time on a

given day and thus each time series accounts for one daily traffic pattern of the given traffic cluster. Then, we performed another clustering analysis to cluster these 80 daily time series into two distinct day groups. This is to observe temporal (day-to-day) patterns of a given spatial traffic cluster. For each traffic stream cluster, we performed simple K-means clustering to group the associated 80 daily time series into two day groups. The results of this temporal K-means clustering are presented in Fig. 10—(a) Cluster 1 and (b) Cluster 2—, where each subfigure shows the entire 80 daily time series for the given traffic stream cluster, categorized by their day-of-week. The temporal clustering results are indicated by different line styles: the green solid line represents *Day group A* and the purple dotted line represents *Day group B*, based on K-means clustering. The result shows that, for Cluster 1, the time-dependent traffic stream size on Sunday is clearly different from that on the other weekdays (*Day group 2* in (a) is mainly composed of time series from Sundays), while a distinct daily pattern is observed on Monday for Cluster 2 (*Day group 2* in (b) is mainly composed of time series from Mondays). This clearly reveals that different spatial clusters can have different daily traffic patterns and identifying distinct spatial traffic stream clusters using the proposed clustering approach can help the discovery of the underlying temporal patterns for different parts of the network.

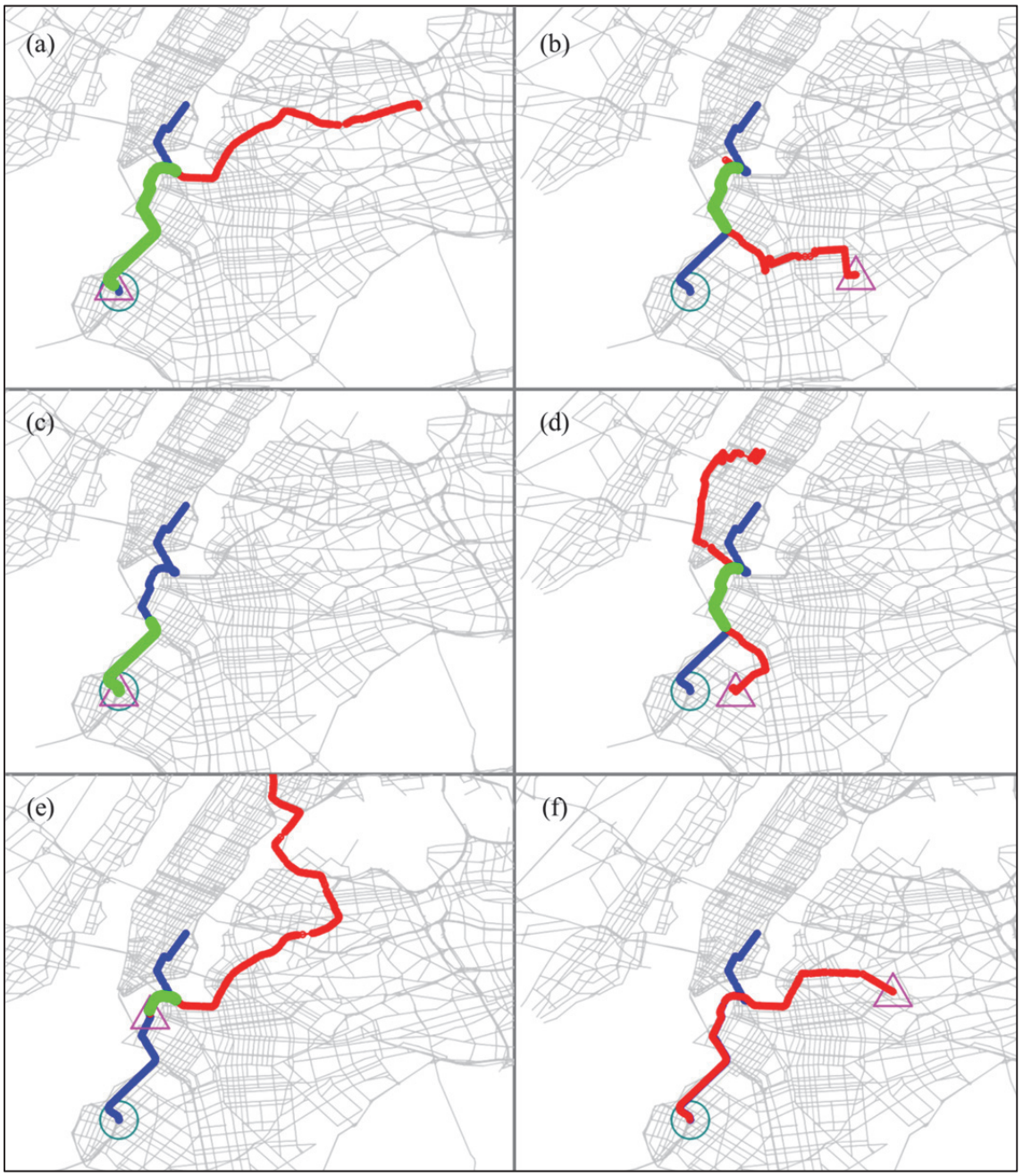

Fig. 7. Similarity measure s between two trajectories Tr_i and Tr_j, i.e., $s = simSeq(Tr_i, Tr_j)$: (a) s=0.63, (b) s=0.30, (c) s=0.99, (d) s=0.29, (e) s=0.12, and (f) s=0.00.

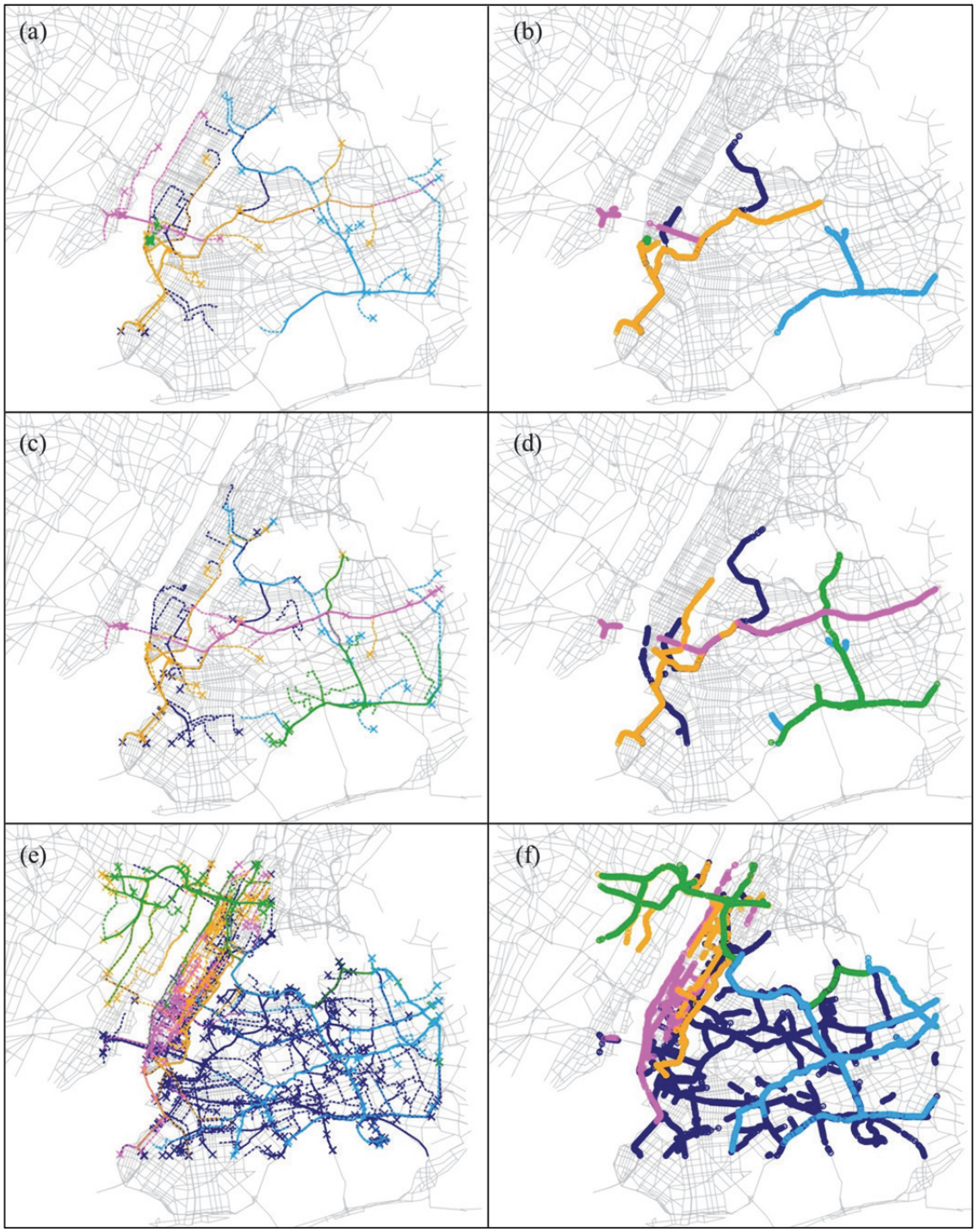

Fig. 8 The effect of DBSCAN parameters on clustering results, i.e., member trajectories (left) and the associated cluster-representative subsequences (right); three cases are tested and the number of final clusters is five in all cases: (a,b) ε=0.2 and *MinTrs*=10, (c,d) ε=0.4 and *MinTrs*=14, and (e,f) ε=0.9 and *MinTrs*=27.

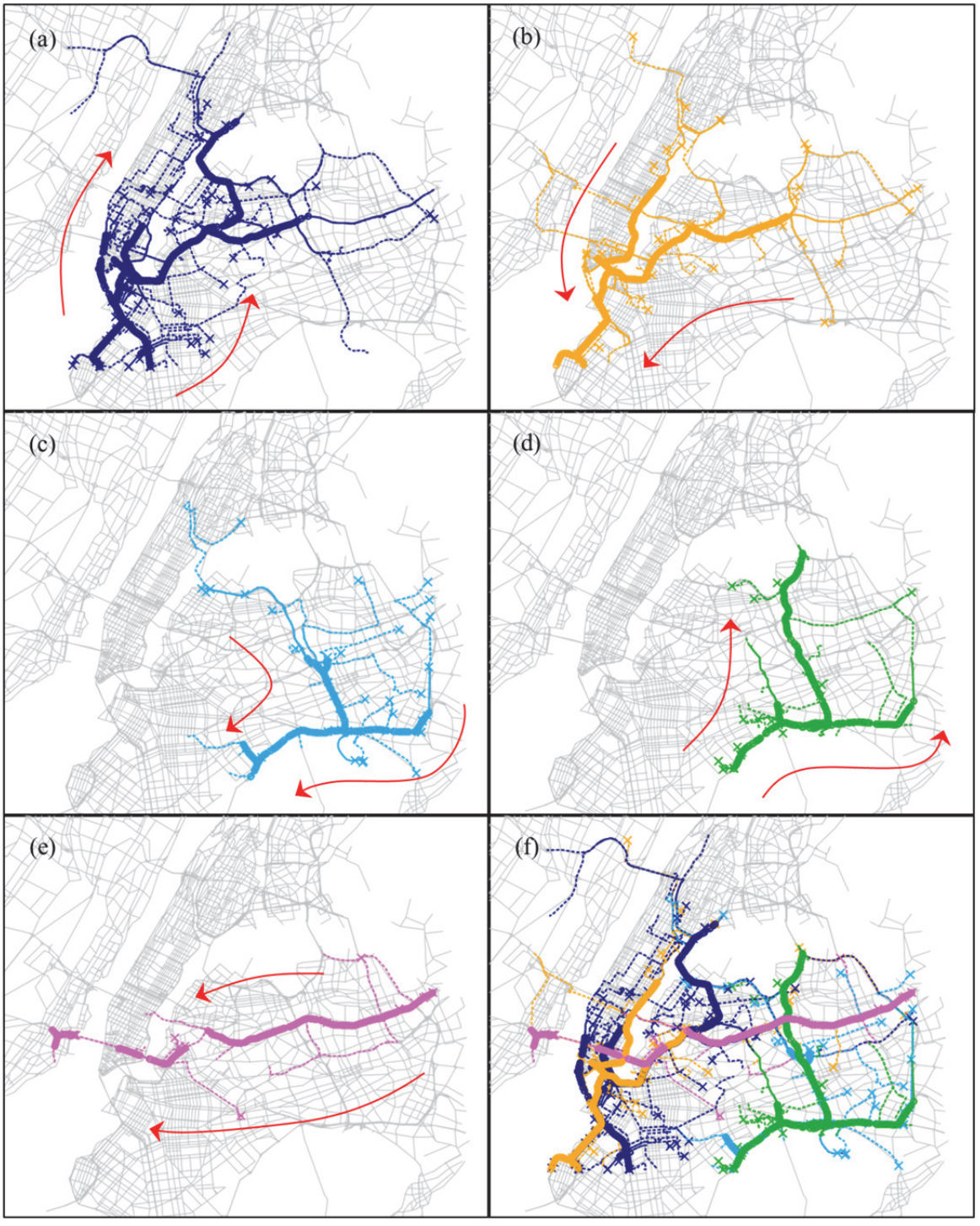

Fig. 9. Five traffic flow clusters obtained from DBSCAN clustering with ε=0.4 and *MinTrs*=14: (a) *Cluster 1*, (b) *Cluster 2*, (c) *Cluster 3*, (d) *Cluster 4*, (e) *Cluster 5*, and (f) all of 5 clusters.

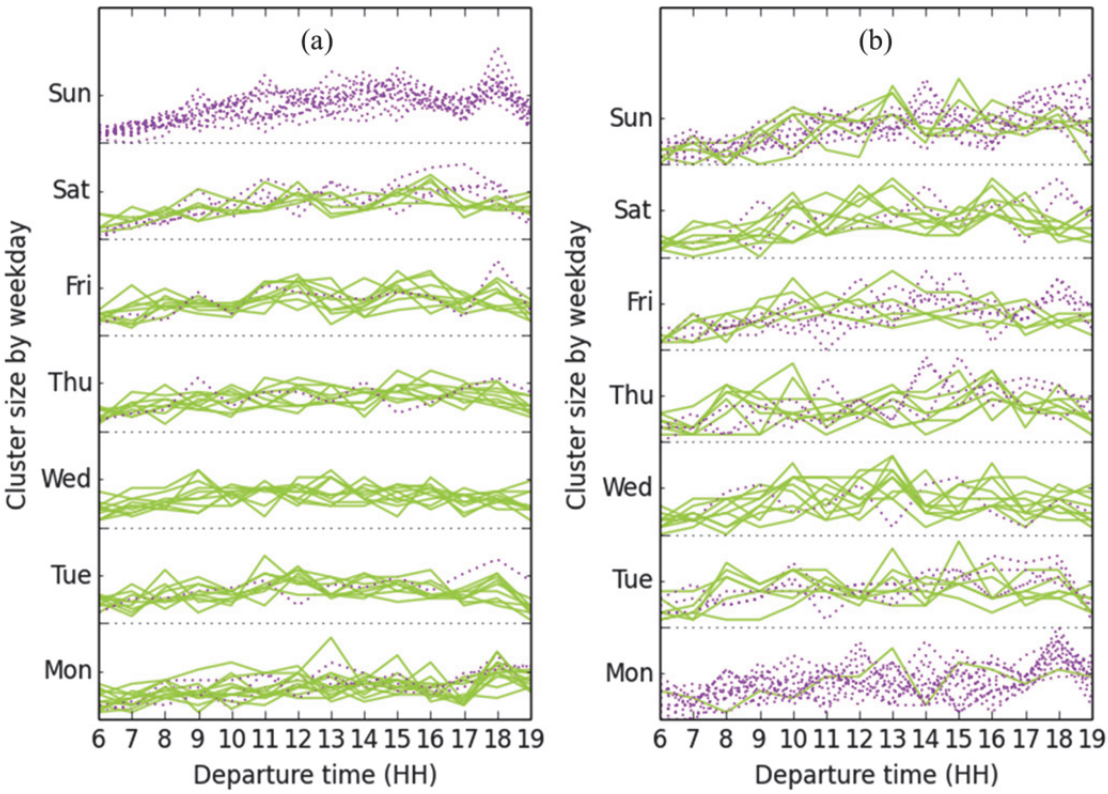

Fig. 10. Temporal clustering of daily traffic patterns of a selected spatial traffic stream cluster: (a) *Cluster 1* and (b) *Cluster 2*; 80 daily time series of cluster size evolution are grouped into two day groups, basedon time series clustering using a K-means method: *Day group A* (green solid line) and *Day group B* (purple dotted line).

4. Conclusion

The goal of a cluster analysis to find and visualize natural groupings and patterns in a data set in an unsupervised manner. Traditionally, traffic analysis is performed for a particular link, segment, OD-pair, and network, where the associated spatial boundaries are typically chosen by analysts based on geographical regions and/or pre-defined traffic analysis zones (TAZ). With the availability of vehicle trajectory data, combined with advanced data-mining techniques, however, it is now possible to detect traffic stream clusters in a more automatic and data-driven manner and perform traffic analysis within and across those clusters. This study proposes a trajectory clustering approach to revealing such patterns and providing the basis for performing cluster-level traffic analysis.

The paper focuses on identifying spatially distinct traffic flow groups using trajectory clustering and investigating temporal traffic patterns of each spatial group. The main contribution of this paper is the development of a systematic framework for clustering and classifying vehicle trajectory data, which does not require a pre-processing step known as *map-matching* and directly applies to trajectory data without requiring the information on the underlying road network. The framework consists of four steps: similarity measurement, trajectory clustering, generating cluster representative subsequences, and classifying trajectories. First, we propose the use of the Longest Common Subsequence (LCS) between two vehicle trajectories as their similarity measure, assuming that the extent

to which vehicles' routes overlap indicates the level of closeness and relatedness as well as potential interactions between these vehicles. We then extend a density-based clustering algorithm, DBSCAN, to incorporate the LCS-based distance in our trajectory clustering problem. The output of the proposed clustering approach is a few spatially distinct traffic flow clusters, which together provide an informative and succinct representation of major network traffic streams. Next, we introduce the notion of cluster representative subsequence (CRS), which reflects dense road segments shared by trajectories belonging to a given traffic stream cluster, and present the procedure of generating a set of CRSs by merging the pairwise LCSs via hierarchical agglomerative clustering. The CRSs are then used in the trajectory classification step to measure the similarity between a new trajectory and a cluster. The proposed framework is demonstrated using actual vehicle trajectory data collected from New York City, USA. A simple experiment was performed to illustrate the use of the proposed spatial traffic stream clustering in application areas such as network-level traffic flow pattern analysis and travel time reliability analysis.

Acknowledgements

This paper is based on research conducted under the Strategic Highway Research Program SHRP-2 project L04 "Incorporating Reliability Performance Measures in Operations and Planning Modeling Tools." The work presented here reflects valuable comments obtained from the SHRP-2 project Technical Expert Task Group. The authors are especially grateful to William (Bill) Hyman, SHRP-2 reliability program manager and Stephen Andrle, SHRP-2 capacity program manager, for their continued support and encouragement throughout this effort. The authors of course remain solely responsible for the content of this paper.

References

Ankerst, M., Breunig, M.M., Kriegel, H., Sander, J., 1999. OPTICS: Ordering Points To Identify the Clustering Structure. ACM Press, 49–60.

Bergroth, L., Hakonen, H., Raita, T., 2000. A survey of longest common subsequence algorithms, in: Seventh International Symposium on String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. 39–48. doi:10.1109/SPIRE.2000.878178

Ester, M., Kriegel, H., S, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, 226–231.

Han, B., Liu, L., Omiecinski, E., 2012. NEAT: Road Network Aware Trajectory Clustering, in: 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS). Presented at the 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS), 142–151. doi:10.1109/ICDCS.2012.31

Hermes, C., Wohler, C., Schenk, K., Kummert, F., 2009. Long-term vehicle motion prediction, in: 2009 IEEE Intelligent Vehicles Symposium. Presented at the 2009 IEEE Intelligent Vehicles Symposium, 652–657. doi:10.1109/IVS.2009.5164354

Kharrat, A., Popa, I.S., Zeitouni, K., Faiz, S., 2008. Clustering Algorithm for Network Constraint Trajectories, in: Ruas, A., Gold, C. (Eds.), Headway in Spatial Data Handling, Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg, 631–647.

Kim, J., Mahmassani, H., 2014. How Many Runs? Analytical Method for Optimal Scenario Sampling to Estimate Travel Time Variance in Traffic Networks. Transportation Research Record: Journal of the Transportation Research Board 2467, 49–61. doi:10.3141/2467-06

Kim, J., Mahmassani, H., Vovsha, P., Stogios, Y., Dong, J., 2013. Scenario-Based Approach to Analysis of Travel Time Reliability with Traffic Simulation Models. Transportation Research Record: Journal of the Transportation Research Board 2391, 56–68. doi:10.3141/2391-06

Lee, J.-G., Han, J., Whang, K.-Y., 2007. Trajectory Clustering: A Partition-and-group Framework, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07. ACM, New York, NY, USA, 593–604. doi:10.1145/1247480.1247546

Mahmassani, H.S., Kim, J., Stogios, Y., Currie, K., Vovsha, P., 2013. Incorporating Reliability Performance Measures in Operations and Planning Modeling Tools (SHRP2 Project L04 Final Report No. S2-L04-RR-1).

Mahrsi, M.K.E., Rossi, F., 2013. Graph-Based Approaches to Clustering Network-Constrained Trajectory Data. arXiv:1310.5249 [cs] 7765. doi:10.1007/978-3-642-37382-4_9

Nanni, M., Pedreschi, D., 2006. Time-focused clustering of trajectories of moving objects. J Intell Inf Syst 27, 267–289. doi:10.1007/s10844-006-9953-7

Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., Andrienko, G., 2008. Visually Driven Analysis of Movement Data by Progressive Clustering. Information Visualization 7, 225–239. doi:10.1057/palgrave.ivs.9500183

Roh, G.-P., Hwang, S., 2010. NNCluster: An Efficient Clustering Algorithm for Road Network Trajectories, in: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (Eds.), Database Systems for Advanced Applications, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 47–61.

Vlachos, M., Kollios, G., Gunopulos, D., 2002. Discovering similar multidimensional trajectories, in: 18th International Conference on Data Engineering, 2002. Proceedings. Presented at the 18th International Conference on Data Engineering, 2002. Proceedings, 673–684. doi:10.1109/ICDE.2002.994784