


Recognizing Network Trip Patterns Using a Spatio-Temporal Vehicle Trajectory Clustering Algorithm

Zihan Hong, Ying Chen, and Hani S. Mahmassani 

Abstract—This paper presents a **spatio-temporal trajectory clustering method for vehicle trajectories in transportation networks** to identify heterogeneous trip patterns and explore underlying network assignment mechanisms. The proposed algorithm **ST-TOPOSCAN** is designed to consider both temporal and spatial information in trajectories. We adopt the **time-dependent shortest-path distance measurement** and take advantage of topological relations of a predefined network to discover the shared sub-paths among trajectories and construct the clusters. The proposed algorithm is implemented with a trajectory dataset obtained in the Chicago area. The results confirm the method's ability to extract and **generate spatio-temporal (sub-)trajectory clusters and identify trip patterns**. Extensive numerical experiments verify the method's performance and computational efficiency. **Through spatio-temporal data mining**, this paper contributes to exploring traffic system dynamics and advancing state-of-the-art spatio-temporal clustering for vehicle trajectories.

Index Terms—Data mining, spatio-temporal data clustering, dynamic network trip patterns, vehicle trajectories, shortest path distance.

I. INTRODUCTION

IN RECENT decades, trajectory database management has emerged as an important problem due to rapid advances in tracking facilities, such as GPS/WIFI-enabled car navigation devices, smartphones, and high-sensitivity satellites [1], [2]. With the widespread use of location sensing technologies, spatial data is becoming readily accessible and widely available. Massive volumes of spatial datasets provide us with opportunities to work on spatial data mining and motivate the need for effective techniques for processing and mining trajectory data, fostering a broad range of applications [3].

Different types and large amounts of spatio-temporal data introduce challenges for data analysis and necessitate state-of-the-art approaches to uncover hidden knowledge in vehicle trajectory sets. Available databases cover data for both natural phenomena and human society, including animal movement data for path-level analysis [4], hurricane and storm tracking data for weather forecasting [5], trackers for buses,

cyclists, and runners [6], and activity locations uploaded to social networks [7]. Vehicle trajectory datasets provide great value in tracking and recording patterns of life. Our interest here is in vehicle trajectory clustering and trip pattern recognition in transportation networks to support a variety of intelligent transportation system applications.

A number of clustering algorithms for static data have been proposed; representative ones include K-Means [8], BIRCH [9], DBSCAN [10] and OPTICS [11]. By integrating the intrinsic semantics of moving behaviors, trajectory clustering algorithms have served to monitor and aggregate individual moving activities [12]–[17]. Spatio-temporal data mining for movement analysis requires further development because most existing studies do not consider information from both (temporal and spatial) dimensions simultaneously, but only focus on one dimension or separate them in analysis and applications [17]–[23]. The methods adopted in those studies can be categorized into four groups: (1) emphasize the temporal order of events at predefined locations of interest [18], [20]; (2) set a temporal domain (or time interval) and measure corresponding distances within the spatial dimension [17], [21], [23]; (3) use separate measurements for each dimension [19]; and (4) set a temporal domain and translate the time ratio into the spatial dimension [22].

Spatio-temporal vehicle trajectory clustering requires exploring both **spatial granularity levels** and **noteworthy temporal features**. It is not straightforward to identify the most efficient approach to this task amongst the many existing algorithms in the data mining and statistics literature. Instead of choosing among various methods to extract the representative vehicle trajectory patterns, **it is more appropriate to define and formalize the notion of similarity among trajectories or sub-trajectories**. The proposed trajectory clustering algorithm takes advantage of the **network topology and time-space distance measurements** within vehicle trajectory data. The time-space distance between two locations is dependent on the **time-dependent shortest-path (TDSP) distance** in the network, and the gap between the time stamps when the object is detected at each targeted location. Basing the clustering algorithm on this new space-time distance measurement differentiates this algorithm from prior studies.

The main contributions of this work lie in the **integration of traffic system dynamics into a spatio-temporal data-mining problem** and using the **hidden traffic assignment mechanism** to

Manuscript received February 12, 2017; revised July 6, 2017; accepted September 2, 2017. The Associate Editor for this paper was M. Zhou. (Corresponding author: Hani S. Mahmassani.)

The authors are with Northwestern University, Evanston, IL 60208 USA (e-mail: masmah@northwestern.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TITS.2017.2754401

recognize the driving behaviors and trip patterns in a systematic and continuous process. A case study with data obtained from the Chicago area is conducted to demonstrate the ability of the proposed framework to extract vehicle trajectory patterns and explore heterogeneous driving behaviors and trip patterns. The results are comparable with the spatial-only clustering results generated by TOPOSCAN [24], with the added ability to define temporally varying traffic system dynamics for traffic management and operation. This algorithm is also applicable with predicted trajectories in a simulation-based traffic estimation and prediction system or with suggested routes in a recommendation system. Within a connected environment, the real-time dataset of predicted routes can be linked with OD (Origin-Destination) pairs to identify promising (sub-)trajectory clusters as potential on-demand corridors or routes for car-sharing programs. Following a review of related literature, additional network management applications of spatial-network trajectory clustering are discussed in the next section. Section III presents the problem formulation and algorithmic procedures. Section IV describes an application to trajectories extracted from the Chicago network, including extensive numerical experiments to investigate the methodology's effectiveness and computational performance. Concluding comments, limitations, and future opportunities are discussed in Section V.

II. LITERATURE REVIEW

A number of clustering algorithms have been proposed and applied in a variety of disciplines. Typical models include:

- Connectivity-based models, also known as hierarchical models.
- Centroid-based models, where the centroid is regarded as the representative of a cluster. The well-known K-means [25] algorithm provides a formal approach to find K centroids for K clusters.
- Distribution-based models, which are closely related to statistics and used to filter the objects most likely belonging to the same distribution.

A. Clustering Algorithms for Trajectories

Most of classical clustering algorithms deal with discrete point data or fixed-dimensional vector data. Trajectory data are different in that a trajectory contains temporal sequential information and may not share the same number of data points as other trajectories in the dataset.

The first algorithm for trajectory clustering is a probabilistic regression model proposed by Gaffney *et al.* [26], which groups objects that are likely to be generated from a common core trajectory by adding Gaussian noise, and was applied to typhoon tracks [13], [14]. Nanni [27] adapted two classical distance-based clustering algorithms (the classical K-means algorithm and the hierarchical agglomerative method) to trajectories. Another type of approach focuses on discovering the longest common subsequence or the count of common subsequences of two series (e.g., see [28] and [29]). Lee and Whang [1] sought to identify sub-trajectory patterns

using a density-based algorithm, TraClus, within a partition-and-group framework and applied it to a hurricane tracker and animal movement.

All the algorithms mentioned are built on a common assumption that objects can move freely in a two-dimensional space. However, this is not true in physical transportation networks, where vehicles are subject to constructed infrastructure networks. A growing number of algorithms have been proposed to explore trajectory clustering on networks. Recent works by Won *et al.* [15], Kharrat *et al.* [2], and Mohamed *et al.* [30], [31] have advanced the field in that direction. NETSCAN [2] is the first algorithm that incorporates the continuity of movements in a constrained transport network. The continuity is represented by a $n \times n$ transition matrix, where n is the number of segments. This matrix could be regarded as the prototype of the network topology model. There are two drawbacks of such models. First, the algorithm could be costly with a sparse transition matrix; this can be solved by storing the matrix efficiently. Second, the trajectories are associated with dense paths, and thus the algorithm cannot be applied to pointed location trajectory data.

Besides, Han *et al.* [16] recognized that the DBSCAN-styled algorithms, including TraClus, are using the Euclidean distance, which is reasonable for freely moving objects but inappropriate for network-restricted trajectories. Han *et al.* [16] proposed an algorithm, NEAT, for a road network model with the shortest-path distance to define the neighborhood. NEAT is designed for undirected networks, which makes the bidirectional lanes share the same identifier and U-turn movements are not detected. The algorithm complexity also increases greatly by introducing flow and density factors. Another approach that does not consider the network constraint, was presented by Kim and Mahmassani [32].

B. Applications of Trajectory Clustering

Clustering vehicle trajectories and discovering similar individual trips lead to a range of location-based service applications [2], [16]. Applications of spatial clustering in transportation have received more attention in the last couple of years. Using individual trajectories, Palma *et al.* [33] proposed a speed-based spatio-temporal clustering method to uncover interesting places. Chen *et al.* [34] developed a coherence expanding algorithm and adopted the absorbing Markov chain model to investigate the problem of discovering the most popular route through GPS trajectories. Guo *et al.* [35] conducted a spatial clustering of massive GPS points to recognize potentially meaningful places and extract the flow measures of clusters to understand the spatial distribution and movements. Bahbouh *et al.* [36], [37] proposed a framework to identify on-demand corridors from origin-destination information. We propose three new promising application examples for traffic management and operations:

1) *Commuter Ride-Sharing*: Car-sharing, ride-sharing, and other forms of shared mobility have been advocated as alternative solutions [38], [39] to reduce congestion and improve air quality and other externalities imposed by the massive use of single-driver private cars. The clustering analysis of

drivers' daily trajectories (with locations published on social networks) or vehicle trajectories obtained through probe data enables tracking patterns of human travel behaviors and activities, thus capturing potential carpool solutions. The rationale is to group travelers who share similar trips patterns, behaviors, and preferences.

2) *Dynamic Path-Based Signal Coordination*: Synchronization of traffic signals along major arterials is commonly used to provide uninterrupted progression for traversing vehicles. Recognizing the time-varying nature of traffic patterns, Abdelghany *et al.* [40] proposed a path-based coordination scheme that provides progression along dominant paths, identified dynamically as traffic conditions unfold, especially in the aftermath of a major disruption. Recently, Bahboub *et al.* [36], [37] proposed a framework based on TraClus to identify on-demand corridors from origin-destination information. Since the on-demand corridors may vary under different operational conditions, spatio-temporal trajectory clustering helps to dynamically identify dominant paths along which to provide path-based signal coordination.

3) *Snow Maintenance Vehicle Route Design*: Snowplow vehicles play important roles in winter road maintenance. The snowplow vehicle route is usually predetermined by vehicle routing strategies, which is basically an optimization problem aimed at covering a set of routes and simultaneously satisfying a set of operational constraints [41]. Here we suggest a road hierarchy for snowplow routing that is generated from the demand side instead of the supply side. Frequently visited road segments are detected as sub-path patterns or on-demand corridors. Such road segments should receive more attention for snow maintenance vehicle routing.

III. METHODOLOGY

To identify heterogeneous trip patterns and explore network assignment mechanisms, the aim of this paper is to discover similarities among vehicle trajectories using a spatio-temporal clustering method. In this section, the problem statement is shown by mathematical notations and equations, where the clustering analysis among trajectories is defined as an optimization problem under thresholds of the distance and size of each cluster. Second, the distance measurement adopted in this method covers both spatial and temporal dimensions. Third, the logic design of a heuristic solution to this optimization problem is illustrated by an example of both 2-D and 3-D cases, where the 2-D case can be regarded as a special scenario with ultimate temporal constraint of the 3-D case. Finally, the procedures and algorithms for this heuristic solution are described with a flowchart and pseudo code.

A. Problem Statement

From a graph-theoretical point of view, a spatio-temporally moving object generates a trajectory TR , taken as a continuous function of time that returns the position of the object in a d -dimensional space at time t (typically, $d \in [2, 3]$). From a data-mining point of view, the continuous function is usually (piecewise) linearized and discretized into a trajectory vector

with a time-ordered sequence of location points. In this study, the trajectory refers to the trajectory vector, which is available from the spatial dataset.

Given a set of trajectories $T = \{TR_1, TR_2, \dots, TR_n\}$ obtained from a transport network, our goals are to involve the intrinsic semantics of the temporal dimension and generate a set of clusters $\Theta = \{C_1, C_2, \dots, C_k\}$ of trajectories or sub-trajectories, considering both temporal and spatial dimensions. The definitions of trajectory, network, link, path, and cluster are illustrated as follows.

Definition 1: A trajectory is a time-ordered sequence of location points. It is denoted as a vector (i.e., a trajectory vector) $TR = \{veh_{id}, p_1, p_2, \dots, p_n\}$, where veh_{id} is the vehicle identifier indicating which vehicle the trajectory belongs to. $p_k = \{x, y, t\}$ ($1 \leq k \leq n$) indicates the location and time when the record of p_k is reported. The location data are usually denoted as x and y , longitude and latitude coordinates (if a GPS system is used as the geographic coordination system). The number n for trajectories can vary. A subsequence of locations in a trajectory forms a sub-trajectory $STR = \{veh_{id}, p_i, p_{i+1}, \dots, p_j\}$, ($1 \leq i < j \leq n$).

Definition 2: A link is also known as a directed edge, denoted as $l = (l_{id}, n_i, n_j) \in E$. It represents a directed road segment from n_i to n_j with identifier l_{id} .

Definition 3: A network is represented by a single-directed graph $G = (V, E)$, consisting of node set $V = \{n_1, n_2, \dots, n_N\}$ and directed edges $E = \{(l_{id}, n_i, n_j) | n_i, n_j \in V\}$. The identifier l_{id} is the key to recognize links, and one link from n_i to n_j does not share an identifier with its opposite link from n_j to n_i . Nodes define junctions, and edges indicate street or road segments between two junctions in the network.

Definition 4: A path is a set of time-ordered, connected links traveled by a vehicle veh_{id} , denoted as $P = \{(veh_{id}, l_i, l_j, \dots, l_m) | l_i, l_j, \dots, l_m \in E\}$. There is a mapping between trajectories and paths, with shared veh_{id} . The set of trajectories $T = \{TR_1, TR_2, \dots, TR_n\}$ can be treated as the set of paths $TRP = \{P_1, P_2, \dots, P_n\}$. Similarly, a sub-trajectory can be assigned to its sub-path.

Definition 5: A cluster is a set of trajectories, **which share some portions of trajectories**. A trajectory can belong to multiple clusters, since it may have several shared portions, but the one with most portions is regarded as the main cluster for this trajectory.

Since a trajectory vector can be mapped and transferred into a path vector, a cluster of trajectories is equivalent to a cluster of mapping paths. To solve this problem and achieve our goals, this problem is defined mathematically as an optimization problem to group the trajectory (path) set into clusters, and in each cluster, the shared sub-path serves as a representative.

This study proposes a measure of within-cluster similarity which does not require one to calculate all the pairwise similarities within a cluster. Instead, the within-cluster similarity, γ_i , is computed as the ratio between the length of the common sub-path and the average length of the paths assigned to this cluster. The objective function is defined to maximize the within-cluster similarity under thresholds. In other words, the algorithm is to find the longest shared sub-path among vehicle

trajectories (paths) for each cluster under the constraints of cluster size and distance, given by:

$$\arg \max_{\Theta} \sum_i^k \gamma_i \quad (1)$$

$$\text{Such that } \gamma_i = \frac{||\cap P||}{\sum_{P \in C_i} ||P||/N_i} \quad (2)$$

$$N_i \geq \alpha \quad (3)$$

$$Dist(l_{k|P_m}, l_{k|P_n}) \leq \beta, \quad (4)$$

$$P_m, P_n \in C_i \text{ and } P_m \neq P_n$$

where, $\cap P$ refers to the shared common links or sub-paths of the paths within C_i . $||P||$ denotes the length of paths. k denotes the number of clusters. N_i is the number of paths in C_i , and it should be no smaller than the threshold α . γ_i denotes the representative ratio and within-cluster similarity of this cluster. It is calculated as the total length of the shared common links divided by the average length of paths in this cluster. γ_i is viewed as the similarity measurement in this study, as the longer the shared common links are within the cluster, the more similar paths are in that cluster. $l_{k|P_m}$ denotes link l_k within path m , and $l_{k|P_n}$ denotes link l_k within path n . $Dist(l_{k|P_m}, l_{k|P_n})$ is the time-space distance between the same link l_k traveled by different vehicles, i.e., path m and path n . The distance between the $l_{k|P_m}$ and $l_{k|P_n}$ should be close enough if path m and path n belong to the same cluster. The detailed distance measurement is described in the next section.

Note that the threshold α and β should be set case by case. It depends on the properties of the trajectory dataset as well as the practical application of the results. For example, if the results are to be applied for a commuter ride-sharing program, α may refer to the minimum number of vehicles in a ride-sharing group, and β means the time window of the pick-up and drop-off service of ride-sharing vehicles; if the results are to be used for path-based signal design, α is related to the volume of the vehicle to be served and β depends on the coordination among signals.

B. Distance Measurement

To define the distance measurement, we need to find how to explore the information of temporal dimension. Nanni and Pedreschi [17] showed that two trajectories may become very similar if restricted within a small temporal interval, but they vary a lot when the entire time horizon is considered. Accordingly, they proposed an algorithm with a focus on the temporal dimension aimed at searching for the most meaningful time intervals. This assumption works for the moving objects without any obvious temporal rules or patterns but not vehicles in a transport network as traffic conditions and assignment patterns are distinct for peak versus off-peak hours. Moreover, in a transport network, the temporal dimension is closely connected to the spatial dimension with the time-dependent vehicle moving speed, and thus it is more appropriate to focus on both dimensions simultaneously by using the TDSP distance measurement.

Take any trajectory point (x, y, t) in TR_m , the geographical information (x, y) could be first mapped to the link l_i with the

entry time $t_{i,m}$. This specific link l_i traveled in TR_m or P_m denoted as $l_{i|P_m}$. The trajectory dataset is thus transferred to the path dataset with sequential links and the time entering each link. The distance between two links l_i in path P_m and links l_j in path P_n is defined as the SP distance between two intersections where vehicles enter the two links. In other words, it is the SP distance between node n_p of l_i and node n_q of l_j . The total time-space distance is equal to the travel time on the shortest path plus the difference between the entry time, given by

$$Dist(l_{i|P_m}, l_{j|P_n}) = \min\left(\frac{SP_{P_m}(n_p, n_q)}{s_{P_m}}, \frac{SP_{P_n}(n_q, n_p)}{s_{P_n}}\right) + |t_{i,m} - t_{j,n}| \quad (5)$$

where, s_{P_m} represents the prevailing speed of vehicle along path P_m at $t_{i,m}$; similarly, s_{P_n} is the prevailing speed of a vehicle along path P_n at $t_{j,n}$. $SP_{P_m}(n_p, n_q)$ that denotes the shortest path from node n_p to node n_q for the vehicle in P_m , and it may differ from $SP_{P_n}(n_p, n_q)$ when generated by the TDSP algorithm.

Note that $Dist(l_{k|P_m}, l_{k|P_n})$ is a special case since the shortest path between $l_{k|P_m}$ and $l_{k|P_n}$ should be 0, and $Dist(l_{k|P_m}, l_{k|P_n})$ remains the difference between the entry time of each path.

C. Logic Design of Heuristic Solution

The intuitive notion of clusters in this study refers to the paths that share the most links within the constraint of spatio-temporal distance. The key idea is to find a start link and to extend this link to its consecutive links such that the extended links are shared by no less than a predefined number of paths in the space. Paths assigned to any cluster are regarded as useful data, whereas others are viewed as noise in this study. The presented algorithm should have the following capabilities:

- To differentiate between useful data clusters and noise in the original dataset [1].
- To extract an arbitrary number of clusters within an adjustable threshold to better fit the dataset, but with considerably lower complexity [10], [24].
- To generate non-spherical clusters with arbitrary shape and variable dimensions, unlike the connectivity-based models (hierarchical algorithms) and centroid-based models (e.g., K-means algorithm) [17].

Following the key idea, a heuristic solution is designed for this optimization problem. Fig.1 shows a detailed example of the generation procedure in both the 2-D and 3-D space. Several concepts adopted in the solution are defined as follows:

Definition 6: A target link l_{tar} is a link in the path set which has been shared by the most number of vehicles (paths) compared to any other link; if several links are shared by the same number of vehicles, the longest one is preferred. The target link is the first link assigned to the target segment.

Definition 7: A target segment TK , as the representative of a cluster, is a set of consecutive target links. It is initiated with the first target link in the cluster and extended to one of the incoming or outgoing links as the next target link. The extension procedure follows the restrictions of cluster size and distance.

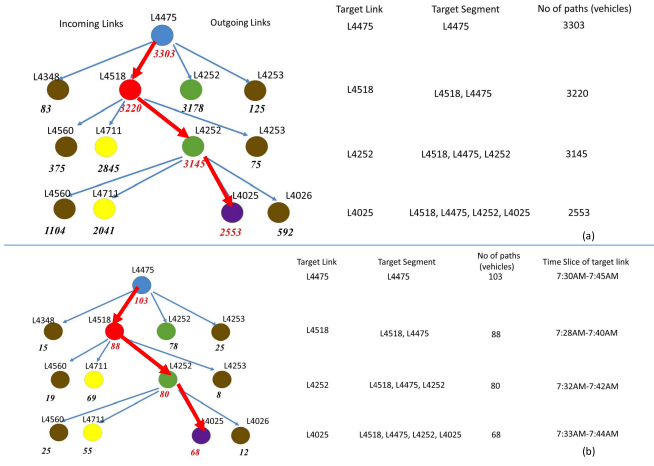


Fig. 1. A Generation Procedure Example of Shared Paths: (a) 2-D case; (b) 3-D case.

Definition 8: An incoming link is a link whose end node is the start node of the target link (or target segment). In other words, if $l = (l_{id}, n_i, n_j) \in E$ is the target link, the incoming links are $l_k = (l_{id_k}, n_{i_k}, n_{j_k} | n_{j_k} = n_i) \in E$. Similarly, an outgoing link is the link whose start node is the end node of the target link, i.e., $l_k = (l_{id_k}, n_{i_k}, n_{j_k} | n_{i_k} = n_j) \in E$. The link in the opposite direction, $l_o = (l_{id_o}, n_j, n_i) \in E$, belongs to both the incoming link set I_l and the outgoing link set O_l of the target link l in case a U-turn movement comes from l to l_o or from l_o to l .

In the 2-D space, the target links could only be extended to the connected links. The 2-D case can be regarded as a special scenario with ultimate temporal constraint such that the distance between any two paths sharing the same links is regarded as zero. The algorithm is initiated to find a link that is traveled by most vehicles.

For instance, the link L4475 is traveled by 3,303 vehicles in Fig. 1, which is also displayed in Fig. 1 (a) for the 2-D case and (b) for the 3-D case as the root node. The nodes on its left side are incoming links and the right ones are outgoing links. Each one is tagged with its link ID and the number of vehicles traveled through the target segment (shared links). The one with the most-traveled vehicles is selected. In this example, the algorithm needs to find which connected link is the next target link from four connected links (L4348, L4518, L4252, L4253) of link L4475. The four links are considered as incoming links (L4348, L4518) and outgoing links (L4252, L4253), respectively. The algorithm then confirms that L4518 is the most popular incoming link with 3,220 out of 3,303 vehicles from L4518 to L4475, and L4252 is the most popular outgoing link with 3,178 out of 3,303 vehicles from L4475 to L4252. Thus, L4518 is selected to join the target links. Meanwhile, the remaining 83 vehicles (paths) that did not travel both links are filtered as noise in this cluster. Next, the algorithm seeks the most popular incoming and outgoing links for the new target segment (L4518 to L4475) and determines that L4475 join the target links with 3,145 vehicles traveling through the entire target segment. The algorithm is repeated to search for the next link and add it into the target segment as the shared sub-path until the number of

vehicles traveling through is less than the predefined cluster size. The target segment with consecutive links forms the sub-path of the cluster.

In the 3-D space, the algorithm is also initiated to seek a target link. Different from the 2-D space search, where no temporal constraint exists, the algorithm is designed to find a link that is traveled by most vehicles within a time slice. This time slice is predefined as an extendable time domain of interest (e.g., morning peak from 6:30AM to 7:00AM) and set no more than the distance threshold, as in (4), such that the initial vehicles meet the threshold of distance measurement. Similar to the 2-D space search, the algorithm is repeated to find the next popular link to join the target segment and filter the vehicles (paths) that cannot meet the distance threshold or did not travel through the entire target segment. Note that the time slice (the earliest arrival time to the latest arrival time) of the target link is updated every iteration: if the target link is an incoming link, the time slice may move backward to an earlier period within the early arrival time of the incoming link; otherwise, the time slice may move forward for the outgoing link. If the time slice and spatio-temporal distance threshold is predefined as infinity, the 3-D clustering results are equivalent to the 2-D results and the algorithm searches for the clusters in the entire time horizon.

With the proposed logic design, the following improvements compared with other trajectory clustering algorithms can be achieved:

- The **bidirectional segment and U-turn movement** problems are addressed with the directed network definition in TOPOSCAN.
- The algorithm is more general for the network-constrained movements such that the network topology is adapted to show the connectivity of the links and possible movements. It also eliminates the redundancy of impossible movements. The network topology is also represented as a look-up table to scan the network.
- **The computation cost drops by the connectivity-based search of the next target link instead of a global search.** building an indexing structure. The time complexity of DBSCAN-styled models (including TraClus) is mostly influenced by the queue structure to expand clusters. Without the use of an indexing structure, the complexity remains $O(n^2)$. Once an accelerating index is used, the average complexity would drop to $O(n \log n)$.
- **This algorithm is compatible with multi-dimensional networks**, including social networks and traffic networks. For each traveler, the trip pattern is trackable through the vehicle trajectory data in the traffic network and the checking status from the social network.
- The algorithm is compatible with other traffic science-related factors, including traffic flow, density, or travel-time reliability to form a comprehensive driver pattern and traffic assignment analysis incorporated with traffic condition diagnoses.

D. Procedures

The procedures of spatio-temporal trajectory clustering are displayed in Fig. 2. The input data include the trajectory

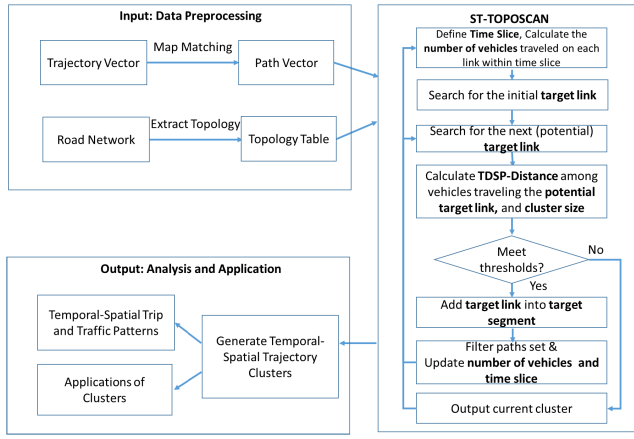


Fig. 2. Procedures of Spatio-Temporal Trajectory Clustering.

vectors and the road network. Before evolving into the algorithm, the input data need preprocessing to map the trajectories onto the network as paths and build the topology table by extracting topology information from the road network. The detailed data preprocessing steps are similar to what is needed for TOPOSCAN as a 2-D case (illustrated in [24]).

This algorithm, named ST-TOPOSCAN, is designed in a recursive framework for path clustering. First, a time slice is defined and the number of vehicles traveling through each link within the time slice are counted. Second, the start link l_{tar} is selected from the path set into the target segment. Third, the target segment is extended by iteratively selecting the next start link according to the network topology attribute table. If the potentially extended target segment satisfies the predefined thresholds, this start link joins the target segment and builds the shared sub-path of this cluster. Then, the algorithm needs to filter the path set and exclude the one that does not contain the entire target segment and update the number of traveling vehicles and time slice according to the new path set and target link. The search for a start link and an update of path sets are conducted iteratively until no more links can be found to satisfy the thresholds. When $l(l \in E)$ is selected in the target segment, it is marked to avoid repeated selection in this cluster.

E. Algorithm

1) *TDSP Algorithm*: III-E shows a label-setting algorithm based on a backward search procedure to generate the TDSP and to calculate the shortest-path travel time [42]. The input data for this algorithm include the origin intersection node, destination intersection node, departure time, and the estimated time-dependent link traversal time.

The time-dependent link traversal time from node n_i to node n_j for an entry time h is defined as t_{ij}^h , which is extractable from the trajectory dataset. Then the time when the vehicle leaves this link is defined as $h + t_{ij}^h$. The TDSP algorithm helps to calculate the shortest travel time $u_o^{h_0}$ and generate the shortest path $p_o^{h_0}$ from origin o to destination s departing at time h_0 .

The computational complexity of the algorithm mainly depends on the number of links (m) and the number

Algorithm TDSP

Input: (1) Time-dependent link traversal time matrix with entry time h, t_{ij}^h
 (2) The origin node o , and destination node s
 (3) Departure time h_0

Output: (1) Shortest path travel time u_o^h
 (2) Shortest Path p_o^h

Algorithm:
 /*Step 0: Initialization*/
 01: $u_i^h \leftarrow \infty, \forall h, i \neq s$
 02: $u_s^h \leftarrow 0, \forall h$
 03: $p_s^h = 0$
 /*Step 1: Update*/
 04: Initialize $q = \{s\}$
 05: **While** $q \neq \emptyset$, **Select** a node j from q , and remove it from q
 06: **For all** ij
 07: **For every** $h, t_e = \min\{T, h + t_{ij}^h\}$
 08: **If** $u_j^{t_e} + t_{ij}^h < u_i^h$
 09: $u_i^h = u_j^{t_e} + t_{ij}^h, p_i^h = ij$
 10: **Insert** i into q
 11: **End If**
 12: **End For**
 13: **End For**

Fig. 3. TDSP Algorithm.

of nodes (n). Update of link traversal time (line 4 to line 13) takes no more than $O(mn)$ times. Thus, the complexity of this algorithm is $O(mn)$, which is practically acceptable. T in line 7 represents the upper bound for the temporal dimension (horizon length). If the entry time of node n_j from node n_i is beyond the upper bound, i.e., $h + t_{ij}^h > T$, no record of $t_{jk}^{h+t_{ij}^h}$ can be retrieved from the dataset as the link traversal time from node j to any other node k at the entry time $h + t_{ij}^h$; the link traversal time at any entry time beyond the upper bound T is regarded at the entry time of T .

Note that the historical trajectory dataset provides most information for the shortest paths taken by vehicles; the TDSP algorithm is not adopted for those cases. The TDSP algorithm is important when dealing with any virtual path or potential path for trajectory clustering.

2) *ST-TOPOSCAN Algorithm*: Fig. 4 shows the detailed algorithm with comments. It consists of 4 steps. Aside from Step 0 initialization, Step 1 is to find the link with most number of vehicles traveled during the initial time slice. Step 2 is to extend the target segment from both incoming and outgoing links. Step 3 is to update the path set and exclude those without the extended target segment from Step 2. Note that line 15 and 27, \bar{P} means a path traveling l_k at the central timeline of T_{SL} for l_k ; it is a virtual path generated through TDSP algorithm. It means that any path to be added in the cluster requires to meet the distance threshold from this virtual path. Step 2 and 3 are conducted repeatedly until the size of path cluster is smaller than the size threshold.

As illustrated in Fig. 4, the parameter α suggests the size of the sub-path cluster, which is the number of vehicles that traverse the target segment. A larger α imposes a higher requirement to select the target link and leads to fewer loops to extend the target segment. Therefore, a long consecutive target segment cannot be detected if α is set large enough. The parameter β refers to the maximal time-space distance between two links.

Algorithm ST-TOPOSCAN

Input: (1) A set of Path $TRP = \{P_1, P_2, \dots, P_n\}$
(2) Network Topology Attribute Table
(3) The threshold parameter α , minimal size of a path cluster
(4) The threshold parameter β , maximal distance between the core link and the connected link
(5) Initial Time Slice T_{SL}

Output: (1) A set of target segments, $\Psi = \{TK_1, TK_2, \dots, TK_m\}$, representative of clusters
(2) A set of Path Clusters $\Theta = \{C_1, C_2, \dots, C_m\}$

Algorithm:
/*Step 0: Initialization*/
01: $\Psi, \Theta \leftarrow \emptyset$
02: $i \leftarrow 1$ ---iteration, the number of clusters to be generated
/*Step 1: Find Target Link*/
03: $TK_i, C_i \leftarrow \emptyset$
04: Calculate the number of vehicles k_{l_k} traveling $l_k \in TRP$ within T_{SL}
05: Select l_{tar} from l_k such that $k_{l_{tar}} \geq k_{l_k}$, for $l_k \in TRP$ & $l_k \neq l_{tar}$
06: Mark $k_{l_{tar}} \leftarrow -1$
07: InsertFirst(l_{tar}, TK_i) --- put l_{tar} as the first element in TK_i
08: $\forall P \in TRP$, Insert P into C_i if $l_{tar} \in P$ --- select potential paths for this cluster
09: Go to Step 2, if $|C_i| \geq \alpha$
10: Return Ψ, Θ , otherwise
/*Step 2: Extend Target Segment */
11: $TK_i \leftarrow TK_i, C_i \leftarrow C_i$
12: If $|TK_i| = 1$
13: $l_{start} = l_{end} \leftarrow TK_i[0]$ ---get the initial link of the target segment
14: Calculate Dist($l_{ks}|P_m, l_{ks}|P_n$), for $P_m, P_n \in C_i$ where $l_{ks} \in l_{start} \cup O_{l_{end}}$
15: Remove P_m from C_i Dist($l_{ks}|P_m, l_{ks}|P_n$) $> \frac{1}{2}\beta$,
16: Calculate the number of vehicles in C_i traveling l_k where $l_k \in l_{start} \cup O_{l_{end}}$ and denote as k_{l_k}
17: Select l_{ks} from l_k such that $k_{l_{ks}} \geq \max(k_{l_k}, \alpha)$ where $l_k \in l_{start} \cup O_{l_{end}}$
18: Mark l_{ks} as the new target link l_{tar} and $k_{l_{tar}} \leftarrow -1$
19: If $l_{tar} \in l_{start}$
20: InsertFirst(l_{tar}, TK_i) --- put l_{tar} as the start link of TK_i
21: Else
22: InsertLast(l_{tar}, TK_i) --- put l_{tar} as the end link of in TK_i
23: Else
24: $l_{start} \leftarrow TK_i[0], l_{end} \leftarrow TK_i[|TK_i| - 1]$
25: Calculate Dist($l_{ks}|P_m, l_{ks}|P_n$), for $P_m, P_n \in C_i$ where $l_{ks} \in l_{start}$
26: Calculate Dist($l_{ke}|P_m, l_{ke}|P_n$), for $P_m, P_n \in C_i$ where $l_{ke} \in O_{l_{end}}$
27: Remove P_m from C_i Dist($l_{ks}|P_m, l_{ks}|P_n$) $> \frac{1}{2}\beta$, where $l_k \in l_{start} \cup O_{l_{end}}$
28: Calculate the number of vehicles in C_i traveling l_k where $l_k \in l_{start}$ and $l_k \in O_{l_{end}}$ respectively, denote as $k_{l_{ks}}$ and $k_{l_{ke}}$.
29: Select l_{ks} from l_k such that $k_{l_{ks}} \geq \max(k_{l_k}, \alpha)$ where $l_k \in l_{start}$
30: Select l_{ke} from l_k such that $k_{l_{ke}} \geq \max(k_{l_k}, \alpha)$ where $l_k \in O_{l_{end}}$
31: If $k_{l_{ks}} \geq k_{l_{ke}}$
32: InsertFirst(l_{ks}, TK_i)
33: Mark $k_{l_{ks}} \leftarrow -1$
34: Else
35: InsertLast(l_{ke}, TK_i)
36: Mark $k_{l_{ke}} \leftarrow -1$
37: End If
38: End If
/*Step 3: Filter Path Set*/
39: $\forall P \in C_i$, Remove P from C_i if $TK_i \not\subseteq P$ --- update the cluster set
40: If $|C_i| \geq \alpha$
41: Go back to Step 2
42: Else
43: Insert TK_i into Ψ , C_i into Θ --- update results
44: $\forall P \in C_i$ Update TRP such that $P \leftarrow P \setminus \{TK_i\}$ --- update the path set
45: $i \leftarrow i + 1$
46: Go to Step 1
47: End if

Fig. 4. ST-TOPOSCAN Algorithm.

The computational complexity of the algorithm mainly depends on the number of paths (n) and the number of links (m). Calculating the number of vehicles takes no more than $O(n \log m)$ times for a predefined time slice Δt . Selection of a target link l_{tar} and connected link executes no more than $O(pq)$ times, where p and q , respectively, refer to the number of nodes and links within and close to the neighborhood of the

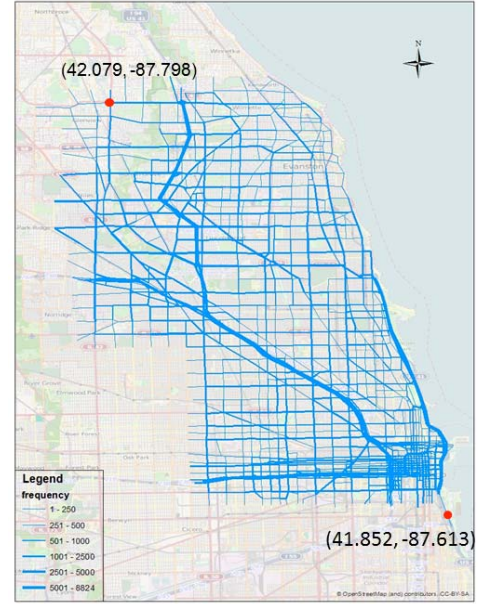


Fig. 5. Frequency of vehicle trajectory paths.

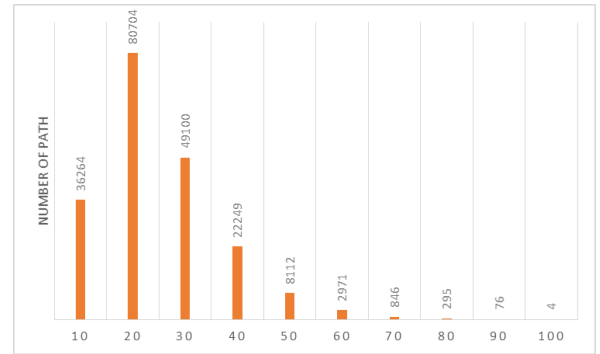


Fig. 6. Histogram of number of links for the path vector.

core links; the complexity of updating the path cluster is $O(n)$. In a small network, it is possible that q is equal to m . Given α and β , the entire algorithm executes $O(n \log m) \times \frac{T}{\Delta t} + O(pq) + i * (O(n \log m)) + O(pq) + O(n)$ times, where i is a reasonable positive value that depends on thresholds, and $\frac{T}{\Delta t}$ is related to the predefined temporal range for 3-D shape. As we are aiming at huge trajectory dataset within a real-world network, where $n \gg p$, and thus the complexity of this algorithm is $O(n \log m)$, which is practically acceptable.

IV. EXPERIMENT AND EVALUATION

A. Simulation Tests and Data Description

The proposed method is applied to the vehicle trajectory data obtained for the Chicago network. The network has 1,578 nodes (545 of which are signalized), 4,805 links (131 of which have road detectors), and 218 traffic analysis zones. It includes the Chicago downtown area located in the central part of the network and four important freeway segments (I-90, I-94, I-290, and Lake Shore Drive). The origin and destination (OD) matrix was generated based on

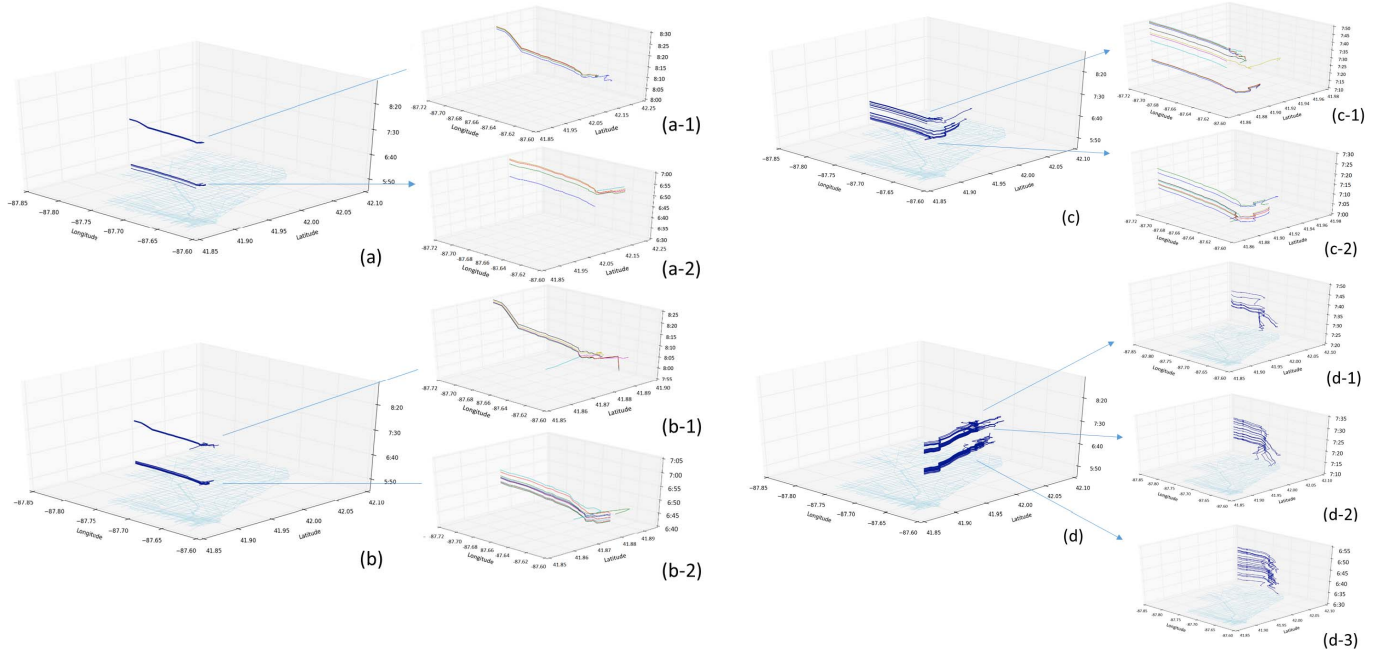


Fig. 7. Examples of clustering results.

the origin-destination data obtained from the CMAP (Chicago Metropolitan Agency for Planning) survey and calibrated using real-world, multiple-day observations obtained from freeway loop detectors of the Chicago area. The simulation tool, DYNASMART [43], is capable of assigning each vehicle from its origin to destination through the dynamically updated shortest path. The vehicle trajectory data is generated from the simulated dynamic traffic assignment through DYNASMART. In the simulation, the location of each generated vehicle is updated every 0.1 minute, and the trajectory of each vehicle is recorded along its routing path from the origin to the destination.

A total of 788,584 vehicles are generated and simulated from 5:00AM to 10:00AM, of which 200,621 vehicles are extracted during the morning peak hours (6:30AM to 8:30AM) for the trajectory clustering approach. Fig. 5 shows the frequency of the extracted vehicle trajectory paths, where the two red dots indicate the geographic range of the network.

B. Clustering Results

More than 200,000 vehicle trajectories are extracted and map-matched to vehicle paths. Each path is comprised of a variable number of links, ranging from 7 to 98, with an average number of 20 and a median number of 18. Fig. 6 shows that the size of most path vectors is ranging from 10 to 30.

Fig. 7 shows four sets of cluster examples that can be categorized into several groups. The left illustration of each set shows the clusters that can be generated when we set the time slice as the simulation horizon. The illustrations on the right are the subsets of the clusters shown in the left illustration, and each of them is the detailed spatio-temporal cluster when the time slice is “zoomed” into the trajectory sets. Both clusters in Fig. 7 (a) show similar trip patterns

that share comparable spatial distributions on an east–west corridor. Fig. 7 (a-2) displays a faster travel speed as the range of the temporal dimension is small. This situation is reasonable as vehicles displayed in Fig. 7 (a-2) depart around 7:00AM when traffic is less congested whereas in Fig. 7 (a-1) vehicles travel around 8:15AM. Similar patterns can be observed for Fig. 7 (b).

Fig. 7 (c) and its sub-figures show the trip patterns that traverse to a main corridor from some portions of the highway. The difference between Fig. 7 (c-1) and 7 (c-2) is due to taking a larger part of the highway and the distance between trajectories is much less (see Fig. 7 (c-2)). Moreover, the travel speed of Fig. 7 (c-2) is faster as the departure time is earlier in the morning peak.

Fig. 7 (d) and its sub-figures show a group of trip patterns that represent the vehicles on the highway to downtown Chicago from the western (and northwestern) suburban areas. The spatial distributions of trajectories are similar to each other, and they are most likely to be put into the same cluster if temporal information was not provided and adopted. The spatio-temporal clusters in these figures show that there are some departure time peaks, i.e., around 6:30AM, 7:30AM, and 8:00AM, which contribute a lot to the morning peak congestion.

C. Performance Measurement

Several runs of the algorithm over an input dataset, which grew from 1,000 to 10,000, are conducted to verify the performance of the proposed algorithm. To cover the entire temporal range and handle variable traffic conditions occurring at different times, the path samples for tests are randomly selected around different departure timestamps. All the experiments are conducted on the server with Intel(R) Xeon(R)

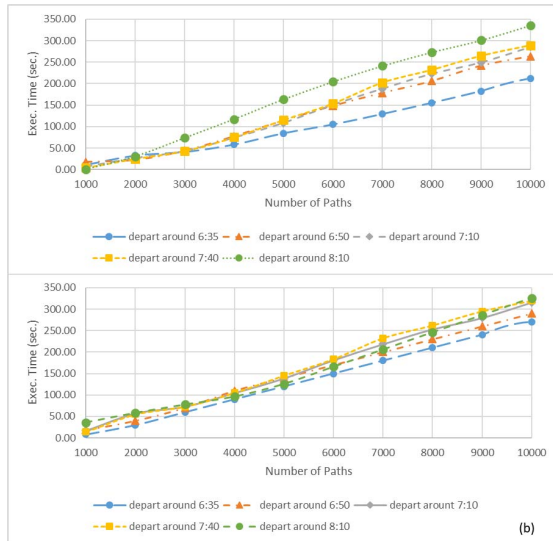


Fig. 8. Performance measurement comparison (a) ST-TOPOSCAN algorithm; (b) ST-DBSCAN algorithm.

CPU X5482 of 3.2GHZ and 3.19GHZ as well as installed memory (RAM) of 32GB. The program to implement the algorithm is written from scratch in Python 2.7.

A summary of execution times is displayed in Fig. 8. The later the vehicle departs within morning peak hours, the more execution time takes. This confirms the increasing complexity of the traffic conditions reflected from the vehicle trajectories when traffic starts getting congested in the morning peak. The curves of ST-TOPOSCAN in Fig. 8. (a) grow less than ST-DBSCAN in Fig. 8 (b) for a less congested network in early morning, and it grows slightly more than ST-DBSCAN when traffic conditions are getting worse. As with ST-DBSCAN, the curves grow almost linearly, which has been explored in the literature as the general output behavior of the density-based clustering algorithm over any type of data that use an accelerating indexing structure [17]. The greatest part of the computational cost of the algorithm is due to the computation of the number of traversing vehicles and distance as well as updating the path sets, which keeps consistent with the statement in Section III-E. The increasing execution time is positively correlated with the increasing size of the path set. According to the complexity as $O(n \log m)$, where n is the number of paths varying in the test, and m is the number of links fixed for the Chicago network, it shows a linear relationship and confirms the effectiveness and efficiency of the proposed algorithm.

V. CONCLUSION AND REMARKS

This paper illustrates a **spatio-temporal trajectory clustering method for vehicle trajectory databases** to discover trip patterns, including departure time, route choice, and travel speed. As an extension of a 2-D clustering algorithm, the proposed algorithm is designed to involve the temporal information of trajectories. The distance measurement used in the algorithm is also extended to the 3-D space, and it is related to the travel time on **the time-dependent shortest path** (TDSP).

The algorithm also takes advantage of topological relations to construct the accelerating indexing structure and find the core links, which help to save computational costs.

One of the main contributions of this study is **exploring traffic system dynamics in the formulation of a spatio-temporal data-mining problem** accounting for heterogeneous driving behaviors and trip purposes. The second contribution pertains to logic and framework design to extend the 2-D method into a 3-D application. Third, the paper contributes to addressing a gap in the literature by **developing a method for spatio-temporal clustering for vehicle trajectories**. Last, but not least, the inclusion of TDSP distance (and/or time) into the clustering procedures contributes to new and promising avenues to exploit the information contained in vehicle trajectories.

The proposed algorithm is implemented with experiments using a dataset obtained in the Chicago area. The results confirm the method's ability to extract and generate spatio-temporal (sub-) trajectory clusters, help identify heterogeneous trip patterns, and explore traffic assignment mechanisms. The numerous experiments verify the performance and establish the computational efficiency of the algorithm.

With future development, this study can be deployed towards **real-time ride-sharing by incorporating trajectory based clustering in the ride-matching logic**. Furthermore, the potential applications of this algorithm listed in Section II-B remain to be fully explored. With the growing accessibility to real-time vehicle trajectory data, the clustering process can be applied and tested under more realistic situations.

REFERENCES

- [1] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2007, pp. 593–604.
- [2] A. Kharrat, I. S. Popa, K. Zeitouni, and S. Faiz, "Clustering algorithm for network constraint trajectories," in *Headway in Spatial Data Handling*. Berlin, Germany: Springer, 2008, pp. 631–647.
- [3] Y. Zheng, "Trajectory data mining: An overview," in *Proc. ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 6, 2015, p. 29.
- [4] S. A. Cushman and J. S. Lewis, "Movement behavior explains genetic differentiation in American black bears," *Landscape Ecol.*, vol. 25, no. 10, pp. 1613–1625, 2010.
- [5] S. D. Aberson and J. L. Franklin, "Impact on hurricane track and intensity forecasts of GPS dropwindsonde observations from the first-season flights of the NOAA Gulfstream-IV jet aircraft," *Bull. Amer. Meteorol. Soc.*, vol. 80, no. 3, pp. 421–427, 1999.
- [6] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "BikeNet: A mobile sensing system for cyclist experience mapping," *ACM Trans. Sensor Netw.*, vol. 6, no. 1, p. 6, Dec. 2009.
- [7] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: A content-based approach to geo-locating twitter users," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 759–768.
- [8] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.
- [9] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Rec.*, 1996, pp. 103–114.
- [10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226–231.
- [11] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Rec.*, 1999, pp. 49–60.
- [12] I. V. Cadez, S. Gaffney, and P. Smyth, "A general probabilistic framework for clustering individuals and objects," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2000, pp. 140–149.

- [13] S. J. Camargo, A. W. Robertson, S. J. Gaffney, P. Smyth, and M. Ghil, "Cluster analysis of typhoon tracks. Part I: General properties," *J. Climate*, vol. 20, no. 14, pp. 3635–3653, 2007.
- [14] S. J. Camargo, A. W. Robertson, S. J. Gaffney, P. Smyth, and M. Ghil, "Cluster analysis of typhoon tracks. Part II: Large-scale circulation and ENSO," *J. Climate*, vol. 20, no. 14, pp. 3654–3676, 2007.
- [15] J.-I. Won, S.-W. Kim, J.-H. Baek, and J. Lee, "Trajectory clustering in road network environment," in *Proc. IEEE Symp. Comput. Intell. Data Mining (CIDM)*, Mar. 2009, pp. 299–305.
- [16] B. Han, L. Liu, and E. Omiecinski, "NEAT: Road network aware trajectory clustering," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2012, pp. 142–151.
- [17] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, 2006.
- [18] Y. Sadahiro and T. Kobayashi, "Exploratory analysis of time series data: Detection of partial similarities, clustering, and visualization," *Comput., Environ. Urban Syst.*, vol. 45, pp. 24–33, 2014.
- [19] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.
- [20] Z. Fang, S.-L. Shaw, W. Tu, Q. Li, and Y. Li, "Spatiotemporal analysis of critical transportation links based on time geographic concepts: A case study of critical bridges in Wuhan, China," *J. Transp. Geogr.*, vol. 23, pp. 44–59, Jul. 2012.
- [21] M. Wang, A. Wang, and A. Li, "Mining spatial-temporal clusters from geo-databases," in *Advanced Data Mining and Applications*. Berlin, Germany: Springer, 2006, pp. 263–270.
- [22] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," in *Proc. VLDB Endowment*, vol. 1, 2008, pp. 1068–1080.
- [23] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wollé, "Reporting flock patterns," *Comput. Geometry*, vol. 41, no. 3, pp. 111–125, 2008.
- [24] Z. Hong, S. Xu, and H. S. Mahmassani, "Network topology aware moving object trajectory clustering," presented at the Transp. Res. Board Annu. Meet., Washington, DC, USA, 2016.
- [25] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k -means clustering algorithm," *Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.
- [26] S. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," presented at the 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Diego, CA, USA, 1999.
- [27] M. Nanni, "Clustering methods for spatio-temporal data," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pisa, Pisa, Italy, Tech. Rep., 2002.
- [28] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 673–684.
- [29] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim, "Fast similarity search in the presence of noise, scaling, and translation in time-series databases," in *Proc. 21th Int. Conf. Very Large Data Bases*, 1995, pp. 490–501.
- [30] M. K. El Mahrssi and F. Rossi, "Graph-based approaches to clustering network-constrained trajectory data," in *Proc. Int. Workshop New Frontiers Mining Complex Patterns*, 2012, pp. 124–137.
- [31] K. Mohamed, R. Guigourès, F. Rossi, and M. Boullé, "Co-clustering network-constrained trajectory data," in *Advances in Knowledge Discovery and Management*. Cham, Switzerland: Springer, 2016, pp. 19–32.
- [32] J. Kim and H. S. Mahmassani, "Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories," *Transp. Res. C. Emerg. Technol.*, vol. 59, pp. 375–390, Oct. 2015.
- [33] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, "A clustering-based approach for discovering interesting places in trajectories," in *Proc. ACM Symp. Appl. Comput.*, 2008, pp. 863–868.
- [34] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Proc. IEEE 27th Int. Conf. Data Eng. (ICDE)*, Apr. 2011, pp. 900–911.
- [35] D. Guo, X. Zhu, H. Jin, P. Gao, and C. Andris, "Discovering spatial patterns in origin-destination mobility data," *Trans. GIS*, vol. 16, no. 3, pp. 411–429, 2012.
- [36] K. Bahbouh and C. Morency, "Encapsulating and visualizing disaggregated origin-destination desire lines to identify demand corridors," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2430, pp. 162–169, Nov. 2014.
- [37] K. Bahbouh, J. R. Wagner, C. Morency, and C. Berdier, "TraClus-DL: A desire line clustering framework to identify demand corridors," in *Proc. 94th Annu. Meet. Transp. Res. Board*, 2015, paper 15-3508.
- [38] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *Eur. J. Oper. Res.*, vol. 223, no. 2, pp. 295–303, 2012.
- [39] E. Kamar and E. Horvitz, "Collaboration and shared plans in the open world: Studies of ridesharing," in *Proc. IJCAI*, 2009, pp. 187–194.
- [40] K. Abdelghany, D. Valdes, A. Abdelfatah, and H. S. Mahmassani, "Real-time dynamic traffic assignment and path-based signal coordination; application to network traffic management," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1667, pp. 67–76, Jan. 1999.
- [41] N. Perrier, A. Langevin, and C.-A. Amaya, "Vehicle routing for urban snow plowing operations," *Transp. Sci.*, vol. 42, no. 1, pp. 44–56, 2008.
- [42] A. K. Ziliaskopoulos and H. S. Mahmassani, "Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications," *Transp. Res. Record*, no. 1408, pp. 94–100, 1993.
- [43] H. S. Mahmassani and H. Shabtyi, "DYNASMART-P version 1.6 user's guide," Northwestern Univ. Transp. Center, Evanston, IL, USA, Tech. Rep., 2009.



Zihan Hong received the B.S. degree in geographic information science from Nanjing University, China, in 2010, and the M.Eng. degree in civil engineering from the Tokyo Institute of Technology, Japan, in 2012. She is currently pursuing the Ph.D. degree in transportation systems analysis and planning with Northwestern University. Her main research focus is on active traffic management strategy design and evaluation, specifically data-driven or performance-driven operational practices. Her other research interests include **urban shared mobility, traffic system characterization, and traffic assignment pattern recognition.**



Ying Chen received the B.S. degree in computer science from Jilin University, China, the M.S. degree in computer science in 2011, and the Ph.D. in civil and environmental engineering from Northwestern University in 2015. She is currently a Research Assistant Professor with the Transportation Center, Civil and Environmental Engineering Department, Northwestern University. Her research focuses on data analytics in transportation applications, specifically big social media data mining for the analysis of **travelers' behavior and large-scale vehicle trajectory data analysis, data visualization, connected vehicles, and smart cities.**



Hani S. Mahmassani holds the William A. Patterson Distinguished Chair in transportation with Northwestern University. He serves as the Director of the Transportation Center. He specializes in multimodal transportation systems analysis, planning and operations, dynamic network modeling and optimization, dynamics of user behavior, and real-time operation of logistics and distribution systems. He received the 2010 IEEE Outstanding Intelligent Transportation Systems Application Award, recognizing pioneering contributions to development of dynamic network modeling software.