Transportation Research Part C 85 (2017) 573–590

Contents lists available at ScienceDirect

Transportation Research Part C

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

Commuter ride-sharing using topology-based vehicle trajectory clustering: Methodology, application and impact evaluation

Zihan Hong[a], Ying Chen[b], Hani S. Mahmassani[a,*], Shuang Xu[c]

[a] Transportation Center, Northwestern University, 600 Foster Street, Evanston, IL 60208, United States
[b] Department of Civil and Environmental Engineering, Northwestern University, 600 Foster Street, Evanston, IL 60208, United States
[c] Department of Geographical and Sustainability Sciences, The University of Iowa, 223 Jessup Hall, Iowa City, IA 52242, United States

ARTICLE INFO

Keywords:
Ride-sharing
Data mining
Travel demand management
Trajectory clustering

ABSTRACT

This paper illustrates a ride matching method for commuting trips based on clustering trajectories, and a modeling and simulation framework with ride-sharing behaviors to illustrate its potential impact. It proposes data mining solutions to reduce traffic demand and encourage more environment-friendly behaviors. The main contribution is a new data-driven ride-matching method, which tracks personal preferences of road choices and travel patterns to identify potential ride-sharing routes for carpool commuters. Compared with prevalent carpooling algorithms, which allow users to enter departure and destination information for on-demand trips, the proposed method focuses more on regular commuting trips. The potential effectiveness of the approach is evaluated using a traffic simulation-assignment framework with ride-sharing participation using the routes suggested by our algorithm. Two types of ride-sharing participation scenarios, with and without carpooling information, are considered. A case study with the Chicago tested is conducted to demonstrate the proposed framework's ability to support better decision-making for carpool commuters. The results indicate that with ride-matching recommendations using shared vehicle trajectory data, carpool programs for commuters contribute to a less congested traffic state and environment-friendly travel patterns.

1. Introduction

The massive use of private cars is associated with systemic congestion and negative impacts on the environment. To mitigate traffic congestion, the Federal Highway Administration (FHWA) has invoked the Dynamic Mobility Applications (DMA) and the Active Transportation and Demand Management (ATDM) programs for the development and deployment of dynamic traffic management strategies and tools (Yelchuru et al., 2013; Vasudevan and Wunderlich, 2013). An underlying premise of these strategies is to influence travelers' behavior towards sustainable travel choices in order to manage congestion, reduce externalities and maximize overall system efficiency.

Carpooling and other forms of ride-sharing emerged in the 1970s as an environment-friendly and sustainable travel alternative. The rationale is that effective usage of empty car seats by ride-sharing may lead to an important opportunity to increase occupancy rates and substantially increase the efficiency of urban transportation systems. Various benefits of carpooling have been shown in the literature, including congestion relief, emission reduction, and individual monetary savings (Katzev, 2003; Fellows and Pitfield, 2000; Chan and Shaheen, 2012; Shaheen et al., 2006, 2012), where the incentives for ride-sharing behaviors and the probability of

* Corresponding author.
E-mail addresses: zihanhong2012@u.northwestern.edu (Z. Hong), y-chen@northwestern.edu (Y. Chen), masmah@northwestern.edu (H.S. Mahmassani), shuang-xu@uiowa.edu (S. Xu).

http://dx.doi.org/10.1016/j.trc.2017.10.020
Received 14 July 2016; Received in revised form 11 October 2017; Accepted 16 October 2017
Available online 24 October 2017

mode and shift towards carpooling have been explored and estimated, primarily for commuting trips (Eash et al., 1974; Ben-Akiva and Atherton, 1977; Train and McFadden, 1978; Dumas and Dobson, 1979; Daniels, 1981; Levin, 1982; Huang et al., 2000; Li et al., 2007; Habib et al., 2011).

From the suppliers' perspective, carpooling serves as a form of public transit for a group of travelers, affording greater spatial coverage but requiring less significant public investment (Buliung et al., 2009). Therefore, planning authorities, especially in congested metropolitan areas, have generally encouraged carpool programs, regarded as an environment-friendly and cost-efficient element of traffic demand management in urban areas. Similarly, employers have facilitated carpooling by employees as a means of reducing on-site parking requirements.

From the users' perspective, the impedance towards greater adoption of carpool commuting is rooted in concerns regarding its reliability, flexibility, efficiency, and security (Li et al., 2007; Chan and Shaheen, 2012). The social, environmental, transportation, and land management benefits from carpooling (Chan and Shaheen, 2012; Shaheen et al., 2006, 2012) encourage the incentives for ride-sharing behaviors but do not provide solutions to its implementation. A practical solution would be to provide the answers to "when and where to carpool" (concerning flexibility and efficiency) and "with whom" (related to security and reliability).

Some on-demand personal mobility providers, such as Uber or Lyft, have emerged and are of growing significance in the urban transportation industry, redefining this market and addressing some of the above concerns. However, these providers primarily aim to provide services for occasional trips, which could occur during any time of the day without predictable nor recognizable patterns (Agatz et al., 2012). Commuting trips, viewed as recurring demand, are beyond their service scope. Unlike occasional trips, commuting trips usually require concentrated supply across spatial and temporal dimensions. The hidden patterns of shared origins, destinations, and compatible arrival times provide clues to a solution for carpooling commuters. The prevalent carpooling algorithms adopted by most providers allow users to enter departure time, origins and destinations for a specific trip; on the other hand, the proposed algorithm focuses more on commuting trips and provides a solution using the hidden patterns of the trips.

This paper aims to explore this solution from a technical point of view, rooted in the analysis of spatio-temporal patterns manifested through individual trajectories. With the widespread use of location sensing technology, spatial data is widely available and easily accessible through advanced information technology. Richer data availability and associated information processing technologies offer promising avenues to promote carpool programs. Particularly, the clustering analysis of drivers' daily trajectories (with locations published on social networks (Cheng et al., 2010) or vehicle trajectories (obtained through probe data) enables tracking patterns of human travel behaviors and activities, thus capturing potential carpool solutions (Han et al., 2012; Kharrat et al., 2008). The rationale behind the proposed solution is that travelers who share similar patterns of trips, including travel behaviors, route choices, and more personalized preferences (i.e. departure time, walking distance from the carpool location, and size of carpool group), could be grouped together. Following this rationale, this paper aims to solve three research questions: (1) what is the hidden pattern of commuter trips, and how to extract those; (2) how to integrate the trip patterns into ride-sharing route design, and (3) how to evaluate the effectiveness of proposed approach in a virtual simulation environment.

The paper first presents a new data-driven, trajectory-based ride-matching method, which tracks personal preferences and aggregated travel patterns, to identify potential ride-sharing routes for carpool commuters. The core part is a trajectory clustering algorithm, which takes advantage of the network topology to detect common sub-paths in a road network with archived vehicle trajectory data and provides potential routes for carpool commuters. To integrate the extracted patterns into ride-sharing route design, we assume that travelers sharing common sub-paths are likely to be matched into a carpooling group. If carpooling information is available in the back-end system (where the ride-matching algorithm is functioning) to extract additional information (e.g., departure time, origins, and destinations), more detailed solutions are achievable to answer implementation questions, such as "When and where do we start and end, and with whom do we carpool?" Finally, the potential effectiveness of the approach is evaluated using a traffic simulation-assignment framework with ride-sharing participation taking the routes suggested by our algorithm. The modeling and simulation framework thus "implements" the carpooling programs, and evaluates the resulting traffic performance with ride-sharing behaviors. Two types of ride-sharing participation tests, with or without carpooling information, are considered. A case study with the Chicago testbed is conducted to demonstrate the proposed framework's ability to support better decision-making for carpool commuters.

The next section presents the framework for the study, followed by a description of the specific algorithms that define the ride-matching and ride-sharing application. Section 4 focuses on the key trajectory matching algorithm, called TOPOSCAN. Section 5 introduces the Chicago testbed, describes the simulation modeling framework used to evaluate the effectiveness of the ride-matching process for commuter carpools, and discusses the results of the case study. Concluding remarks end the paper.

2. Framework

The process of this study is illustrated in Fig. 1 with five separate parts: data, method, results, application and evaluation. The input data include the predefined road network data and vehicle trajectory data. The road network consists of the geographic information of intersections and streets. The vehicle trajectory is a series of sequential locations and timestamps for each vehicle. It provides foundational information for tracking travel behaviors. The trajectory data could be obtained from probe dataset.

The TOPOSCAN algorithm is a network topology-based algorithm, which generates the candidates of shared trips and routes. It consists of four parts: network topology modeling, trajectory mapping, path clustering, and similarity evaluation. The first two parts belong to data preprocessing for path clustering. The path clustering is developed according to the density-based clustering (DBSCAN) principle. The similarity evaluation is a post-processing procedure that measures the accuracy of the clustering results. The results would reveal potential carpool commuting routes.

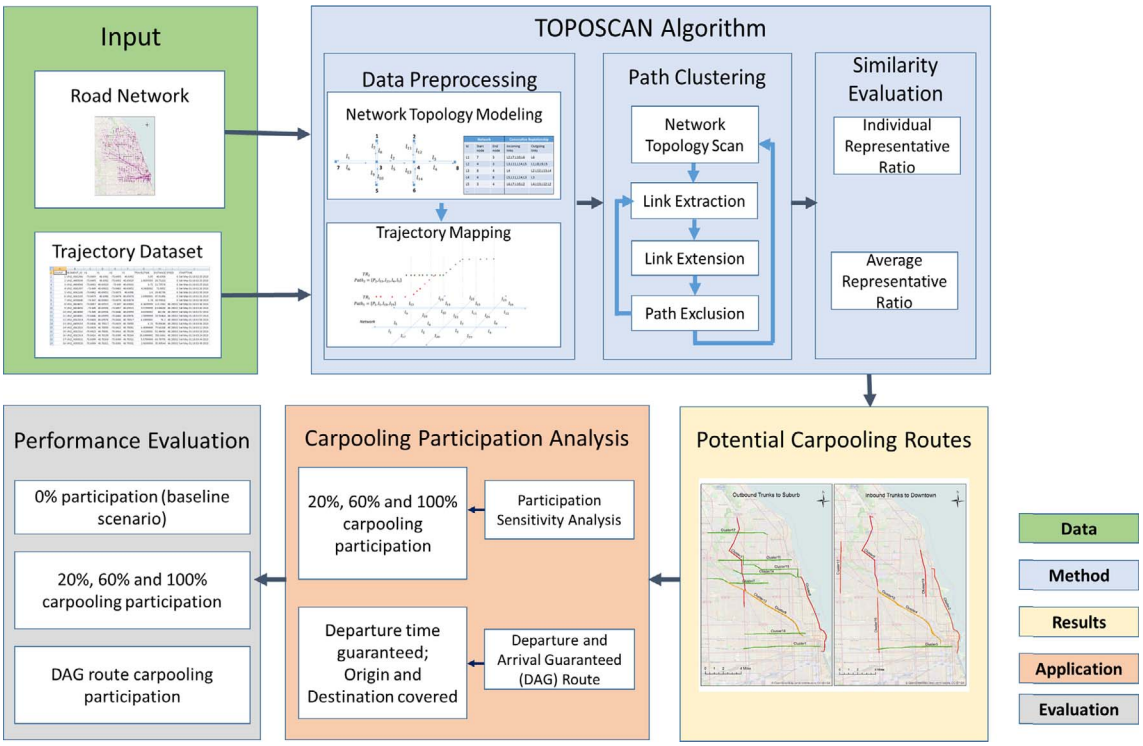


Fig. 1. Process and algorithm for potential carpooling route recognition and participation.

The carpooling participation analysis is conducted with the recognized routes to predict traffic performance. An experimental carpooling case with departure and arrival guaranteed (DAG) routes is simulated from the user perspective to increase the applicability of carpool programs in the real world. Specifically, the simulations are categorized into two groups: with or without carpooling information. The tests without carpooling information help to examine the sensitivity of traffic performance with respect to the percentage of shared trips. The tests with carpooling information are designed for experimental carpooling cases. Given preferred or prevailing departure or arrival information provided by the carpool participants, the DAG routes are designed and selected for ride-sharing drivers. The concept of DAG routes refers to both temporal and spatial dimensions: each proposed carpooling group is suggested to take a DAG route, which shares neighboring origins and destinations among the commuters in that group, with their departure/arrival time within a limited variance. The impact of commuter ride-sharing is evaluated by comparing traffic performance of the baseline scenario with the simulated carpooling scenarios.

3. Related algorithms

This section delineates the technical aspects of the ride-sharing methods and algorithms. One of the first definitions of ride sharing was proposed in an operational test in Sacramento, CA in 1994. It defines dynamic ride-sharing as “a one-time rideshare match obtained for a one-way trip either the same day or the evening before” (Kowshik et al., 1995). Researchers have focused on whether it is possible to find a one-time rideshare trip. Hall and Qureshi (1997) analyzed the likelihood that a person would be successful in finding a ride match, given a pool size of potential ride matches. Using probabilistic analysis, they observed that there were many obstacles, primarily in terms of communication. Therefore, the likelihood of finding a ride match in practice might be smaller.

Most of the dynamic ride-matching applications and pilot tests in the 1980s and 90s failed to provide enough users with consistently successful ride-sharing matches. Subsequent efforts switched to more reliable strategies to encourage ride-sharing, including online ride matching and traveler information services (Ghoseiri, 2013). The new definition emphasized that dynamic ride-sharing is occasional by nature and does not require advanced notice to establish the shared trip. Agatz et al. (2012) summarized the objectives of ride-sharing systems: minimizing system-wide vehicle miles and system-wide travel time, and maximizing the number of participants. Instead of stochastic estimation and operational tests, the majority of works in the recent literature are based on mathematical optimization models and heuristic approaches (Ghoseiri, 2013; Herbawi and Weber, 2012).

However, it is the availability of massive volumes of spatial datasets that provides interesting opportunities for spatial pattern recognition to support ride matching. Data-mining techniques also offer promising avenues to promote carpool programs.

3.1. Clustering algorithms

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 (Hartigan and Wong, 1979) 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.
- Density-based models, which define core points and neighborhoods (within a certain distance from the core point) to generate the density for clustering. The areas with higher density are regarded as a cluster. For example, DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999) (a variant of DBSCAN) are widely used and efficient even for large spatial databases.

3.2. Clustering algorithms for trajectories

Most of the 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 dimension as other trajectories in the dataset (Kim and Mahmassani, 2015).

The first algorithm for trajectory clustering is a probabilistic regression model proposed by Gaffney and Smyth (1999), and applied to typhoon tracks (Camargo et al., 2007a, 2007b). Lee et al. (2007) sought to identify sub-trajectory patterns. They developed a density-based algorithm, TRACLUS, within a partition-and-group framework and applied it to a hurricane tracker and animal movement. TRACLUS shares many characteristics with DBSCAN.

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 transportation, where vehicles are subject to constructed transport networks. There is a growing literature exploring trajectory clustering on networks. Recent works by Won et al. (2009) and Kharrat et al. (2008) have advanced the field in that direction; NETSCAN by Kharrat et al. (2008) is the first algorithm that incorporates the continuity of movements in a constrained transport network. The continuity is represented by an $n \times n$ transition matrix, where n is the number of segments. The matrix gives the number of moving objects traveling from one segment to another a positive value for connected pairs. This matrix model could be regarded as the prototype of the network topology model. However, there are two drawbacks of such models. First, the algorithm is computationally costly, since it uses a lot of zeros for any unconnected pairs. Second, the transition matrix is constructed for linked segments, and thus the algorithm cannot be applied to pointed location trajectory data.

Besides, Han et al. (2012) recognized that the DBSCAN-styled algorithms, including TRACLUS and NETSCAN, are using the Euclidean distance, which is reasonable for freely moving objects but inappropriate for network-restricted trajectories. To solve this problem, Han et al. (2012) proposed an algorithm, NEAT, for a road network model. Instead of using Euclidean distance, they used the shortest path (SP) distance to define the neighborhood and also took into account flow, density, and speed limit. However, 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, which are highly correlated to vehicle speed and the number of trajectories.

Applications of spatial clustering in transportation have received more attention in the last couple of years. Using individual trajectories, Palma et al. (2008) proposed a speed-based spatio-temporal clustering method to uncover interesting places that may not have been expected by the users. Chen et al. (2011) 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. (2012) 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 and Morency (2014) and Bahbouh et al. (2015) proposed a framework based on TraClus (Lee et al., 2007) to identify demand corridors from origin-destination information.

3.3. Improvement of TOPOSCAN

This paper is motivated to develop a more general method for trajectory clustering. The proposed algorithm (TOPOSCAN) offers the following improvements:

- 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 adopted 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 exclusion of correlated factors (flow and density) and dimensionality reduction of trajectory data. 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 (such as a tree data structures) is used, the average complexity would drop to $O(n\log n)$.
- This algorithm can be applied to multi-dimensional networks, including social networks and traffic networks. For each traveler, the route choice pattern is trackable through the traffic network and vehicle trajectory data; the daily travel (trip chain) pattern,

including where and when to go and whom to meet, is also trackable through the social network. With parallel matching on travel behaviors and daily travel patterns, the likelihood that a person successfully finds a ride match is greatly increased. Note that any data from either traffic network or social network is only used as the input to the ride-matching algorithm. The vehicle trajectory data will not be mixed into the public dataset on social networks, to circumvent privacy issues.

Details of the TOPOSCAN algorithm, and its application to the carpool ride-matching problem of interest, are described in the next section.

4. TOPOSCAN algorithm

4.1. Problem statement

Given a set of trajectories $\mathbf{T} = \{\mathbf{TR}_1,\mathbf{TR}_2,...,\mathbf{TR}_n\}$, our goals are to assign them to paths (consecutive network links) to generate a set of clusters $\mathbf{\Theta} = \{\mathbf{C}_1,\mathbf{C}_2,...,\mathbf{C}_m\}$ and obtain common sequential links (common sub-paths or trunks (Sadahiro et al., 2013)) for each cluster $\mathbf{C}_i$, where trajectory, network, link, path, cluster, and trunk are defined as follows.

Definition 1. A trajectory is a time-ordered sequence of location points. It is denoted as $\mathbf{TR} = \{veh_{id},\mathbf{p}_1,\mathbf{p}_2...\mathbf{p}_n\}$, where veh_{id} is the vehicle identifier indicating which vehicle the trajectory belongs to. $\mathbf{p}_k = \{x,y,t\}(1 \leqslant k \leqslant n)$, or more accurately $\mathbf{p}_k = \{x,y,z,t\}(1 \leqslant k \leqslant n)$, indicates the location and time when the record of $\mathbf{p}_k$ is reported. The location data are usually denoted as x and y for longitude and latitude coordinates (if the GPS system is used as the geographic coordination system); z is optional, representing elevation of the location. The number n for any trajectories can vary. A subsequence of locations in a trajectory forms a sub-trajectory $\mathbf{STR} = \{veh_{id},\mathbf{p}_i,\mathbf{p}_{i+1}...\mathbf{p}_j\}$, $(1 \leqslant i < j \leqslant n)$.

Definition 2. A network is represented by a single directed graph $\mathbf{G} = (\mathbf{V},\mathbf{E})$, consisting of nodes $\mathbf{V} = \{n_1,n_2...n_N\}$ and directed edges $\mathbf{E} = \{(l_{id},n_i,n_j)|n_i,n_j \in \mathbf{V}\}$. Nodes define junctions, and edges indicate street or road segments between two junctions in the network.

Definition 3. A link is also known as a directed edge, denoted as $\mathbf{l} = (l_{id},n_i,n_j) \in \mathbf{E}$. It represents a directed road segment from n_i to n_j with the identifier l_{id}. 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.

Definition 4. A path is a set of time-ordered, connected links traveled by a vehicle veh_{id}, denoted as $\mathbf{P} = \{(veh_{id},\mathbf{l}_i,\mathbf{l}_j...\mathbf{l}_m)|\mathbf{l}_i,\mathbf{l}_j...\mathbf{l}_m \in \mathbf{E}\}$. There is a mapping between trajectories and paths, with shared veh_{id}. The set of trajectories $\mathbf{T} = \{\mathbf{TR}_1,\mathbf{TR}_2...\mathbf{TR}_n\}$ can be treated as the set of paths $\mathbf{TRP} = \{\mathbf{P}_1,\mathbf{P}_2...\mathbf{P}_n\}$. Similarly, a sub-trajectory can be assigned to its sub-path.

Definition 5. A cluster is a set of paths, which share the same sub-path (trunk). A path can belong to multiple clusters since it may have several shared sub-paths.

Definition 6. A trunk is a set of time-ordered, connected links, which indicates the shared sub-path.

4.2. Overview

When referring to the density-based clustering (DBSCAN) algorithm, which is applied to the trajectories of freely movable objects, such as animal trajectories discussed in Section 3.2, three concepts need to be clarified as follows:

- **Eps-neighborhood of a point:** the Eps-neighborhood of a point p is defined as a set of points where the distance between the point p to any point in the set is less than the threshold Eps. In a 2D space, the Eps-neighborhood of a point p is regarded as all points in a circle, which has p as the center and Eps as the radius.
- **Minpts:** Minpts is a minimum number of points in an Eps-neighborhood of the target point. It is also known as the minimum size of a cluster.
- **Core Point:** If density of the Eps-neighborhood of a point p is larger than Minpts, the point p is regarded as a core point. The density refers to the number of points in that area.

To apply these concepts to path clustering (as described in Fig. 1) designed for the vehicle trajectories (i.e. network constraint movable objects), several modifications are shown in Fig. 2 and Table 1. In a freely movable area (Fig. 2(a)), the trajectory points can be freely distributed in the two-dimensional space. The key algorithm of DBSCAN is to calculate density of the Eps-neighborhood of each core point to form a cluster using a queue-based search structure. The queue-based search is a first-in-first-out structure with time complexity of $O(n^2)$ for a database of n points, and it can be reduced to $O(n*\log n)$ using a well-designed tree data structure. Meanwhile, NEAT (especially opt-NEAT) is also included in this comparison in Table 1 as it is closest to the proposed algorithm. NEAT algorithm has three phases: base-NEAT, flow-NEAT and opt-NEAT. Base-NEAT (to map points to road segment) is slower than the data preprocessing of the proposed algorithm since it increases the redundancy by introducing the road junctions into the trajectory dataset, instead of reducing number of points. Flow-NEAT is to transform a list of trajectory points into a list of road segments and form the route. Opt-NEAT is to form the cluster using the DBSCAN principle. However, the data structure and

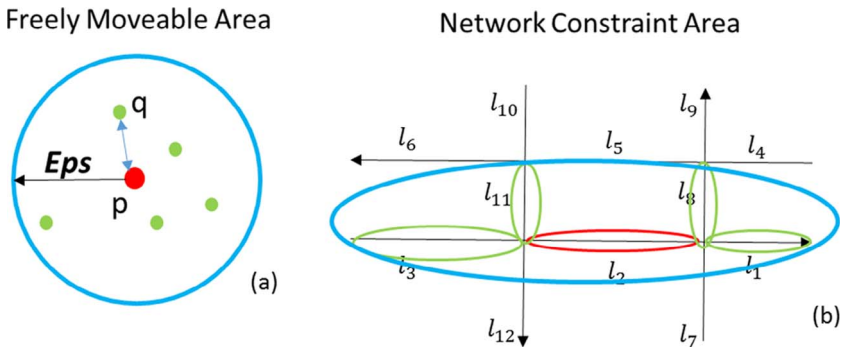


Fig. 2. Density-based clustering algorithm: (a) freely moveable area for DBSCAN; (b) network constraint area for TOPOSCAN.

Table 1
Comparison between DBSCAN, TOPOSCAN and NEAT.

	DBSCAN	NEAT (Opt-NEAT)	TOPOSCAN- Path Clustering
Object	Point	Route (flow cluster)	Link
Neighborhood	A circle with radius of Eps in a 2D space	A circle with radius of Eps, center of ending points of the unit in a 2D space	Connected links. Eps could be regarded as the link length, but Eps is not adopted.
Density	Number of points in the Eps-neighborhood	Number of routes in the Eps-neighborhood of the ending points	Number of vehicles passing the link
Core	Core point	Core Route (No minimum density/cardinality required)	Core link, where its density is larger than threshold
Start	Arbitrary unit	Longest Unit	Most dense unit
Key algorithm	Queue-based search	Unknown	Topology indexing -based search (Tree structure)
Complexity	$O(n^2)$, can be reduced to O (n*logn)	Unknown	O(n*logn)

complexity is not discussed for NEAT algorithm.

In Fig. 2(b), the concepts are applied for vehicle trajectories. The moving direction is constrained by the connected network links. The neighborhood no longer forms a circle with an Eps as the radius but is rather shown as connected links to the core link and can take on any shape. Due to the connectivity between the core and its neighborhood, the distance from connected links to the core could be regarded as zero. The key algorithm is transferred into a linked-list-based structure (or an index-based search structure) to calculate the density of the connected link. An index-based structure adopts a topology indexing of the core link to reduce its complexity to O(log n).

4.3. Data Pre-processing

4.3.1. Network topology model

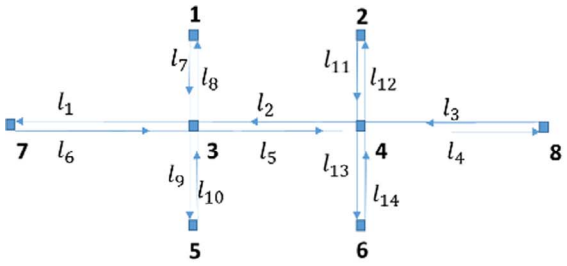
A network is defined as a single directed graph $\mathbf{G} = (\mathbf{V},\mathbf{E})$. In order to detect U-turn movement and distinguish bidirectional road segments, we adopt distinctive identifiers to define network links. In other words, the original bidirectional road segment is separated into two links, with each representing one direction by its identifier l_{id}, start node n_i, and end node n_j. For a better and easier representation, the connection between links is extracted and stored in a topology attribute table. The connected links are categorized into incoming links and outgoing links defined as follows:

Definition 7. An incoming link is known as the link whose end node is the start node of the object link. In other words, if $\mathbf{l} = (l_{id},n_i,n_j) \in \mathbf{E}$ is the object link, the incoming links are$\mathbf{l}_k = (l_{id_k},n_{i_k},n_{j_k} | n_{j_k} = n_i) \in \mathbf{E}$. Similarly, an outgoing link is the link whose start node is the end node of the object link, i.e. $\mathbf{l}_k = (l_{id_k},n_{i_k},n_{j_k} | n_{i_k} = n_j) \in \mathbf{E}$. The link in the opposite direction, $\mathbf{l}_o = (l_{id_o},n_j,n_i)$, belongs to both incoming link set $\mathbf{I_l}$ and outgoing link set$\mathbf{O_l}$ of the object link $\mathbf{l}$ in case that the U-turn movement may come from $\mathbf{l}$ to $\mathbf{l}_o$ or from $\mathbf{l}_o$ to $\mathbf{l}$.

Example 1. Consider the grid network in Fig. 3 We have 8 nodes and 14 single-directed links denoted by their IDs. The table shows its network definition with link ID, start node, and end node, as well as the consecutive relationships with other links in the network.

4.3.2. Trajectory mapping

Trajectory mapping procedures consist of two phases: trajectory location map-matching and path redundancy elimination. Map-matching algorithms have been fully developed in the past decades (Quddus et al., 2007). The objective of map-matching is to replace point sequence trajectory data with link sequence path data. It helps eliminate location errors from the original trajectory dataset and denote the exact location of moving objects. The errors could be caused by either GPS sensors or multilevel road segments.



Network			Consecutive Replationship	
Id	Start node	End node	Incoming links	Outgoing links
L1	7	3	L2,L7,L10,L6	L6
L2	4	3	L3,L11,L14,L5	L1,L8,L9,L5
L3	8	4	L4	L2.L12,L13,L4
L4	4	8	L5,L11,L14,L3	L3
L5	3	4	L6,L7,L10,L2	L4,L13,L12,L2
...				

Fig. 3. An example of a grid network and its topology attribute table.

The map-matching process is initiated with modal matching to identify the correct link among all the links connected with the closest node to the trajectory point and analyze whether the next trajectory point can be matched to the selected link. It is vital to select the correct link carefully and reliably among the candidate links (Definition 8). The weighting system proposed by Quddus et al. (2003) has been used to evaluate candidate links for a correct match. The criteria are the similarity in orientation (the degree of parallelism between the street network link and the arc formed by two consecutive trajectory points) and the proximity of a trajectory point to the link (the perpendicular distance to the link).

Definition 8. The candidate link set **L** for a point $\mathbf{p}_k$ consists of any link with a non-empty intersection of a predefined buffer zone of the point as $\mathbf{L} = \{\mathbf{l}_k | \mathbf{l}_k \in \mathbf{E} \text{and} \mathbf{l}_k \cap^N (p_k) \neq \varnothing\}$, where $N(\mathbf{p}_k)$ represents the neighborhood or buffer of $\mathbf{p}_k$. In this study, the buffer zone is defined as a zone around $\mathbf{p}_k$ within a block size. Similarly, N(**TR**) represents the neighborhood of a trajectory **TR**, where $N(\mathbf{TR}) = \cup^{(} N(\mathbf{p}_k) | \mathbf{p}_k \in \mathbf{TR}$.

Suppose that four points P1 to P4 of a trajectory in an eight-link network (Fig. 4(a)), the link direction for each link is defined relative to a northerly direction. In this case, 0° is for the north bound links (l_5 and l_8), 90° for the eastbound links (l_2 and l_4), and so forth. Given that P1 to P3 are identified on l_2, now the problem is to determine the correct link for P4 from the candidate links l_7, l_4, l_5 and l_1 (if the U-turn behavior is allowed). The selection of candidate links are based on the angle and distance features in Definitions 9 and 10 (see Fig. 5).

Definition 9. The degree of parallelism $\Delta\beta$ is defined as the difference between the driving direction (β) and the link direction (β'). The driving direction β is given by the direction of the current point P_4 and the previous one P_3. The similarity in orientation is formulated by the cosine function of $\Delta\beta$ in Eq. (1).

$$\cos(\Delta\beta) = \cos(|\beta - \beta'|) = \cos(\beta - \beta') \tag{1}$$

Definition 10. The perpendicular distance D from a trajectory point P(x, y) to a link l connecting n1 (x1, y1) and n2 (x2, y2) in Fig. 4 (b) is defined in Eq. (2).

$$D = \frac{x(y_1 - y_2) - y(x_1 - x_2) + (x_1 y_2 - x_2 y_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} \tag{2}$$

The path redundancy elimination aims at reducing dimension and eliminating redundancy for clustering. According to the definition, every trajectory point data has at least three dimensions (x, y, t). When the trajectory data with multi-dimensional points are used for pattern recognition and clustering, we may experience low computational efficiency with huge memory consumption.

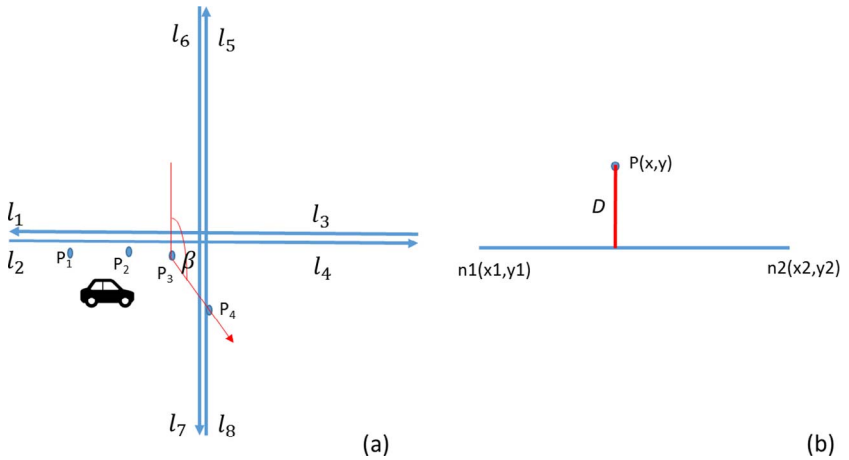


Fig. 4. (a) The degree of parallelism and (b) the perpendicular distance.

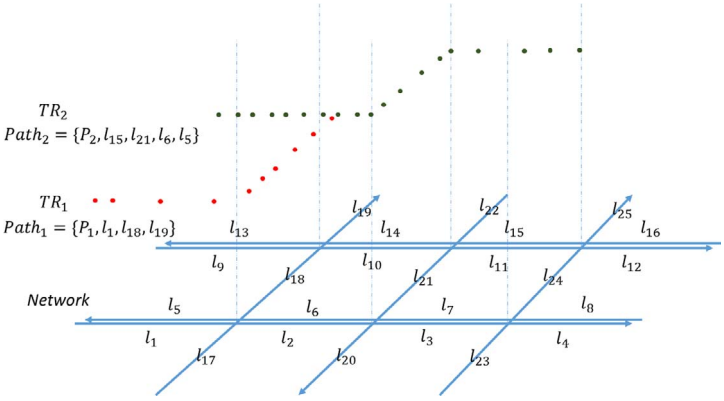


Fig. 5. An example of trajectory mapping on a grid network.

With map matching, points are assigned to a one-dimensional link. However, redundancy may still exist, since it is possible to assign successive points to the same link and repeated links may appear in the path. The proposed algorithm records the link ID and entry time when it first occurs in the successive link assignment and skips the repeated records until a different one occurs.

Example 3. Consider the grid network with bidirectional east-west road segments and one-way north-south road segments in Figure 5.2. Trajectories are shown respectively in red and green dots. Since the example network is defined with one-way segments, the directions of movements are easily identified when it is map-matched to the one-way road. However, in a more general case, it is necessary to distinguish the sequence of points when map matching is conducted to generate path data.

In the trajectory mapping procedures, the second phase could be combined with the first phase by focusing on the trajectory points around the intersections on the trajectories, (i.e., the points that probably indicate the change of driving direction) and skipping the consecutive trajectory points that might be assigned to the same link.

Fig. 6 illustrates the detailed algorithm of Trajectory Mapping, which adopts a traffic network $\boldsymbol{G}$ to transfer a vehicle trajectory $\boldsymbol{TR}$ into a path $\boldsymbol{P}$. It consists of three main steps: initialization, link matching (finding correct link for point assignment) and path assignment. In the initialization step, we first define a vector A with the same length of $\boldsymbol{TR}$ (excluding the veh_{id}), and each member a_i in A means the correct link id that trajectory point p_i is assigned to. The link id in a_i is defaulted as 0 (Line 01). The intersections along the trajectories are retrieved as E (Line 02) and the neighboring trajectory points are denoted as N (Line 03).

In the link matching step, we are trying to find the correct link for points N in each trajectory using a for-loop (Line 04). To be specific, the candidate link set $\boldsymbol{L}_k$ (in Definition 8) for the point is firstly generated (Line 05). Within the candidate link set, the optimal assignment $\boldsymbol{l}_k$ (i.e. correct link assignment) is computed by comparing the similarity in the orientation and the perpendicular

Algorithm **Trajectory Mapping**

Input: (1) A trajectory $\boldsymbol{TR} = \{veh_{id}, \boldsymbol{p}_1, \boldsymbol{p}_2 \ldots \boldsymbol{p}_n\}$,
(2) The network $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = \{n_1, n_2 \ldots n_N\}$ and $\boldsymbol{E} = \{(l_{id}, n_i, n_j) | n_i, n_j \in \boldsymbol{V}\}$
Output: A path $\boldsymbol{P} = \{(veh_{id}, \boldsymbol{l}_i, \boldsymbol{l}_j \ldots \boldsymbol{l}_m)\}$
Algorithm:
/*Step 0: Initialization*/
01: $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, where $a_i = 0, i = 1 \ldots n$ --- *initialize assignment set*
02: $\mathcal{E} = \boldsymbol{E} \cap N(\boldsymbol{TR})$ --- *find all intersections along* $\boldsymbol{TR}$
03: $\mathcal{N} = N(\mathcal{E}) \cap \boldsymbol{TR}$ --- *find the trajectory points around intersections along* $\boldsymbol{TR}$
/*Step 1: Find correct link*/
04: **For** each $\boldsymbol{p}_k \in \mathcal{N}$
05: **Find** $\boldsymbol{L}_k$ for $\boldsymbol{p}_k$ --- *find the candidate link set*
06: **Find** $\boldsymbol{l}_k \in \boldsymbol{L}_k$ such that $arg \min_{\boldsymbol{l}_k}(w_1 \cos(\Delta\beta) + w_2 D|\boldsymbol{l}_k)$, where w_1 and w_2 are predetermined weights with default 1. --- *find the correct link*
07: **Update** $\mathcal{A}$ by $a_k = \boldsymbol{l}_k$ --- *update the assignment set*
08: **End For**
/*Step 2: Path Assignment*/
09: **For** each $a_i \in \mathcal{A}$
10: **Insert** a_i into $\boldsymbol{P}$ ***if*** $a_i \neq 0$ and $a_{i-1} \notin \boldsymbol{P}$ --- *form* $\boldsymbol{P}$ *using non-zero link assignment*
11: **End For**

Fig. 6. A map matching based trajectory mapping algorithm.

distance (Line 06). Then the correct link id is used to update the assignment value a_k in the assignment set A (Line 07).

In the path assignment step, the path ***P*** for the trajectory ***TR*** is generated from the assignment set A using a for-loop (Line 09). For each non-zero link id in A, insert it into the path ***P*** and skip the duplicated ones (Line 10).

The time required to conduct the trajectory mapping for one trajectory depends on N, where N⊑ **V**. The worst case for N occurs when a vehicle goes through the overall network and visits every intersection, and in such a caseN= **V** and the number of $\mathbf{p}_k$ (intersections) in Step 1 is assumed to be |**V**| (the cardinality of **V**). If there are n trajectories to be mapped, the computational complexity is n∗O(|**V**|), which indicates the mapping procedure is a linear time algorithm.

4.4. Path clustering

With data preprocessing procedures, traditional clustering of trajectory point data is simplified and transformed into clustering of one-dimensional path data. Following the density-based principle, a framework with recursive steps of extraction, extension, and exclusion is developed as explicitly as follows:

- Step 1 link extraction: the target link $\mathbf{l}_{tar}$is extracted from the path set;
- Step 2 link extension: $\mathbf{l}_{tar}$is extended by choosing a consecutive link $\mathbf{l}_{con}$ with the highest density from the incoming links and outgoing links (Definition 7). $\mathbf{l}_{tar}$ and the selected $\mathbf{l}_{con}$ construct the trunk;
- Step 3 path exclusion: the path set is filtered and updated to exclude the one that does not contain the trunk.

The second and third steps will be conducted repeatedly until the size of path cluster is smaller than the threshold parameterα. When $\mathbf{l}_k$ is set as $\mathbf{l}_{tar}$ or$\mathbf{l}_{con}$, its density k_{l_k} is marked as -1 to avoid repeated selection. The detailed steps of the algorithm are shown in Fig. 7.

The Path Clustering algorithm adopts a path set ***TRP***, a network topology attribute table as the indexing look-up table, and a predetermined threshold parameter, to generate the path clusters **Θ** and the representative sub-path (trunk) of each cluster. It consists of four main steps: initialization, link extraction, link extension, and path exclusion. In the initialization step, we initialize an empty set for the path clusters and trunks respectively, and define the iteration count (indicating the number of clusters to be generated).

In the link extraction step, for the i^{th} iteration, we initialize the i^{th} cluster and trunk to be an empty set (Line 03). The link density is computed for each link in the path set and the link with maximal link density is selected as the target link (Line 04–06). The target link is inserted in the trunk set (Line 07), and any path containing the target link is inserted into the cluster set (Line 08). If the size of the cluster is larger than the threshold, we continue to step 2; otherwise, the iterations end (Line 09–10), indicating that no more cluster could be found given that threshold.

In the link extension step, we need to find the most densed link (or direction) to extend the trunk. For the other links that belong to the current cluster but not in the trunk, the link density is computed (Line 11). The moving direction determines the location of the consecutive link in the trunk: the second link is the outgoing link of the first link along its direction and so force. For each trunk, if there is only one target link, we select the link with the highest density from its incoming and outgoing links as the consecutive link and insert it into the trunk set (Line 13 to Line 22); otherwise, the consecutive link should be found from the incoming links of the first link or the outgoing links of the last link in the trunk (Line 23 to Line 34).

In the path exclusion step, the threshold parameter is checked again to tell whether the generated trunk and cluster from step 2 can be further extended. If so, the i^{th} path cluster is updated and the link extension step is revisited to find the next link for the trunk (Line 35–37); otherwise, the trunk and path cluster for the i^{th} iteration (i.e. the i^{th} trunk and path cluster) are determined, and step 1 is revisited for the next iteration (Line 38–43).

As illustrated in Fig. 7, a larger α imposes a higher requirement on trunks (meaning that a trunk should be shared with more vehicles) and leads to fewer loops to extend the trunk. Therefore, long trunks cannot be detected if α is set large enough. Generally, within a length range, the number of trunks decreases when α is increasing; for a fixed α, the number of trunks decreases with an increase in trunk length. Similar to the **Minpts** (minimum number of points or minimum density in the cluster) defined in DBSCAN, there is no specific rule for generating threshold α (minimum number of path in the cluster). Instead, it is treated as a predetermined constraint in this algorithm. In the practical implementation, it is usually selected according to the post evaluation, sensitivity analysis, or application constraint.

The computational complexity of the algorithm mainly depends on the number of paths (n) and the number of links (m). In the i^{th} iteration, there will be no more than n paths and m links in the i^{th} path set to be searched. For one path with no more than m links, the complexity to search all the links in this path using a linked-list based search algorithm is O(logm). Therefore, for no more than n paths in the i^{th} path set, the complexity of calculation of density (line 4 and line 11) takes no more than O(nlogm) times. Selection of a target link and consecutive link (line 5, 16, 25 and 26) executes no more than O(m) times, and the complexity of updating the path cluster (line 8 and 36) isO(n). Given α, the entire algorithm executes O(nlogm) + O(m) + k(O(nlogm) + O(m) + O(n)) times, where k is a reasonable positive value and its value depends on α. Thus, the complexity of this algorithm is O(nlogm), which is practically acceptable.

4.5. Similarity evaluation

A path may belong to multiple clusters and has different representative trunks in each cluster, since it could have several shared

Algorithm **Path Clustering**

Input: (1) A set of Path $\boldsymbol{TRP} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots \boldsymbol{P}_n\}$
(2) Network Topology Attribute Table
(3) The threshold parameter α, minimum size of a path cluster (minimum number of vehicles in a path cluster)
Output: (1) A set of trunks, $\boldsymbol{\Psi} = \{\boldsymbol{TK}_1, \boldsymbol{TK}_2, \ldots \boldsymbol{TK}_m\}$
(2) A set of Path Clusters $\boldsymbol{\Theta} = \{\boldsymbol{C}_1, \boldsymbol{C}_2 \ldots \boldsymbol{C}_m\}$
Algorithm:
/*Step 0: Initialization*/
01: $\boldsymbol{\Psi}, \boldsymbol{\Theta} \leftarrow \emptyset$
02: i ←1 *---iteration, the number of clusters to be generated*
/*Step 1: Extraction*/
03: $\boldsymbol{TK}_i, \boldsymbol{C}_i \leftarrow \emptyset$ *--- initialize the result sets as blank*
04: **Calculate** k_{l_k} for $l_k \in \boldsymbol{TRP}$ *--- calculate the density for every link, where the link is in the set of path*
05: **Select** l_{tar} such that $k_{l_{tar}} \geq k_{l_k}$, for $l_k \in \boldsymbol{TRP}$ & $\boldsymbol{l}_k \neq \boldsymbol{l}_{tar}$ *--- select the link with highest density as the target link*
06: **Mark** $k_{l_{tar}} \leftarrow$ -1 *--- make a tag for the target link to avoid repeated selection*
07: **InsertFirst** $(\boldsymbol{l}_{tar}, \boldsymbol{TK}_i)$ *--- insert the target link into the trunk set*
08: $\forall \boldsymbol{P} \in \boldsymbol{TRP}$, **Insert** $\boldsymbol{P}$ into $\boldsymbol{C}_i$ if $\boldsymbol{l}_{tar} \in \boldsymbol{P}$ *--- insert the path into the path cluster set*
09: **Go to Step 2,** if $|\boldsymbol{C}_i| \geq \boldsymbol{\alpha}$ *--- check the threshold*
10: **Return** $\boldsymbol{\Psi}, \boldsymbol{\Theta}$ *--- if it did not go to Step 2 at Line 09, it means the results cannot satisfy the threshold, and thus Return and End.*
/*Step 2: Extension*/
11: **Calculate** k_{l_k} for $l_k \in \boldsymbol{C}_i \backslash \{\boldsymbol{TK}_i\}$ *--- calculate the density for every link, where the link is in the path set but not trunk*
12: $\boldsymbol{TK}_i^{'} \leftarrow \boldsymbol{TK}_i$, $\boldsymbol{C}_i^{'} \leftarrow \boldsymbol{C}_i$*--- set a backup for the results*
13: **If** $|\boldsymbol{TK}_i| = 1$*--- Case 1: there is only one link in the trunk*
15: $l_{tar} \leftarrow \boldsymbol{TK}_i[0]$ *--- this link must be the target link*
16: **Select** $\boldsymbol{l}_{con}$ such that $k_{l_{con}} \geq max(k_{l_k}, 1)$ where $\boldsymbol{l}_k \in \boldsymbol{I}_{l_{tar}} \cup \boldsymbol{O}_{l_{tar}}$ *--- find the consecutive link from incoming and outgoing links of the target link*
17: **Mark** $k_{l_{con}} \leftarrow$ -1*--- make a tag for the consecutive link to avoid repeated selection*
18: **If** $l_{con} \in \boldsymbol{I}_{\boldsymbol{l_{tar}}}$*--- check the location to insert the consecutive link into the trunk*
19: **InsertFirst** $(\boldsymbol{l}_{con}, \boldsymbol{TK}_i)$
20: **Else**
21: **InsertLast** $(\boldsymbol{l}_{con}, \boldsymbol{TK}_i)$
22: **End If**
23: **Else** *--- Case 2: there are more than one link in the trunk*
24: $\boldsymbol{l}_{tar_0} \leftarrow \boldsymbol{TK}_i[0]$, $\boldsymbol{l}_{\boldsymbol{tar_1}} \leftarrow \boldsymbol{TK}_i[|\boldsymbol{TK}_i| - \boldsymbol{1}]$*--- find the consecutive links respectively for the first and the last link*
25: **Select** $\boldsymbol{l}_{con_0}$ such that $k_{l_{con_0}} \geq max(k_{l_{k_0}}, 1)$ where $k_{l_{k_0}} \in \boldsymbol{I}_{l_{con_0}}$
26: **Select** $\boldsymbol{l}_{con_1}$ such that $k_{l_{con_1}} \geq max(k_{l_{k_1}}, 1)$ where $k_{l_{k_1}} \in \boldsymbol{O}_{l_{con_1}}$
27: **If** $\boldsymbol{l_{con_0}} \geq \boldsymbol{l_{con_1}}$*--- check the location to insert the consecutive link into the trunk*
28: **InsertFirst** $(\boldsymbol{l}_{con_0}, \boldsymbol{TK}_i)$
29: **Mark** $k_{l_{con_0}} \leftarrow$ -1
30: **Else**
31: **InsertLast** $(\boldsymbol{l}_{con_1}, \boldsymbol{TK}_i)$
32: **Mark** $k_{l_{con_1}} \leftarrow$ -1
33: **End If**
34: **End If**
/*Step 3: Exclusion*/
35: **If** $|C_i| \geq \alpha$ *--- check the threshold*
36: $\forall \boldsymbol{P} \in \boldsymbol{C}_i$, **Remove** $\boldsymbol{P}$ from $\boldsymbol{C}_i$ if $\boldsymbol{TK}_i \nsubseteq \boldsymbol{P}$*--- update the cluster set*
37: **Go back to Step 2**
38: **Else**
39: **Insert** $\boldsymbol{TK}_i^{'}$ into $\boldsymbol{\Psi}$, $\boldsymbol{C}_i^{'}$ into $\boldsymbol{\Theta}$ *--- if it did not satisfy the threshold, insert the backup results into the output set*
40: $\forall \boldsymbol{P} \in \boldsymbol{C}_i^{'}$ **Update** $\boldsymbol{TRP}$ such that $\boldsymbol{P} \leftarrow \boldsymbol{P} \backslash \{\boldsymbol{TK}_i^{'}\}$ *--- update the path set*
41: i ← i+1
42: **Go to Step 1**
43: **End if**

Fig. 7. A density-based path clustering algorithm.

parts with others. Mathematically and logically speaking, such multi-assignment is reasonable. However, if the identified path patterns are to be applied in real-world applications, such as carpool commuting in this study, it is necessary to evaluate the similarity or coverage between the path set and its trunks to find the optimal representative trunk for the target paths. The optimal representative trunk could be either the longest shared part from its trunks or the one maximizing the average coverage ratio or the

total length of shared paths for the entire path sets. The former maximizes the shared number of vehicles for this path set only, and the latter maximizes the entire utilization of shared paths in a carpooling program.

Two methods are proposed to measure the representative ratio. One is to calculate the average coverage ratio of each trunk with the basic statistic method by:

$$r_i = \frac{||TK_i|| * \sum_{j=1}^{n_i} \frac{1}{||P_j||} n_i}{,} P_j \in C_i \tag{3}$$

where n_i is the size of cluster C_i, $||TK_i||$ is the length of the trunk TK_i for C_i, and $||P_j||$ is the length of the path P_j. The value $r_i$indicates to what extent TK_i could effectively describe the paths inC_i. This value is positively related to the similarity of the trunk to be measured, ranging from 0 to 1.

The other calculates the average coverage ratio of all the trunks by:

$$R = \frac{\sum_{i=1}^{k} n_i ||TK_i||}{\sum_{j=1}^{n_i} ||P_j||}, P_j \in C_i \tag{4}$$

where k is the number of clusters and n_i is the size of cluster C_i. The value R reflects the average similarity of all clusters and trunks. Given the path sets, the longer the detected trunks, the larger the similarity. The value of R also ranges from 0 to 1.

5. Chicago case study

5.1. Data

The proposed method is applied to the vehicle trajectory data obtained for the Chicago network. The network has 1578 nodes (545 of which are signalized), 4805 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 congestion observed along I-90 and I-94 in the morning peak has an average speed of about 25 miles per hour, even lower than the afternoon peak.

The origin and destination (OD) matrix was generated based on the origin-destination data obtained from 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 (Jayakrishnan et al., 1994; Mahmassani, 2001; Mahmassani and Sbayti, 2009) is capable of assigning each vehicle from its origin to the destination through the dynamically updated shortest path. The vehicle trajectory data is generated from the simulated dynamic traffic assignment through DYNASMART.

The baseline scenario is designed to load vehicles from the OD matrix such that 20% vehicles had access to the en-route information and 80% vehicles followed the user equilibrium (UE) assignment rule in which travel time is minimized from the user (traveler) perspective. A total of 788,584 vehicles are generated and simulated from 5:00AM to 10:00AM, of which 204,660 vehicles are extracted during the morning peak hours (6:30AM to 8:30AM) for the trajectory clustering approach. The simulation results produce vehicle trajectory as well as detailed vehicles attributes (ID, origin, destination, departure time, etc.), which are used for sensitivity analysis in the following section. Fig. 8 shows the frequency of the extracted vehicle trajectory paths.

5.2. Route recognition

There is no specific rule for generating threshold values. They are usually selected according to the post evaluation and the sensitivity analysis. Fig. 9 shows the distribution and statistics of clustering results given different threshold values. It illustrates how the value of α affects the number, length, and size of trunks. Within the sameα, the number of trunks decreases with an increase in trunk length. Long trunks cannot be detected if α is set large enough. Within the same length range, the number of trunks decreases when α is increasing. However, there is an exception when the trunks are shorter than 2 miles.

In this study, the threshold α is set to 100 such that the average number of vehicles could be around 4 vehicles every 5 min for the clustered sub-paths, and the minimum length of the trunk is predetermined to be 5 miles to avoid short sub-paths. Fig. 10 illustrates the clustering results for the identification of potential carpool routes for both inbound and outbound vehicles. Cluster 1 and 5 belong to I-290; Cluster 4, 8, 10, and 13 represent I-90; Cluster 3 and 9 are part of I-94; Cluster 2 and 6 show Lake Shore Drive; the rest indicate important arterial roads.

The features of the recognized 19 clusters are shown in Fig. 11, including the length of the trunk, number of vehicles, vehicle miles traveled (VMT), and representative ratio. The representative ratio of single trunk in Fig. 11(d) is denoted by yellow bars, while the average ratio of all trunks are represented by a grey dash line. The large ratio implies that the paths are effectively represented by the trunk of their assigned clusters. Six clusters with a ratio larger than 0.7, i.e., clusters 1, 2, 3, 5, 12, and 18, were chosen to be examined for carpooling participation sensitivity analysis and carpooling scenarios with DAG routes. A significant reduction in VMT and energy consumption was expected from carpooling simulations in the following sections.

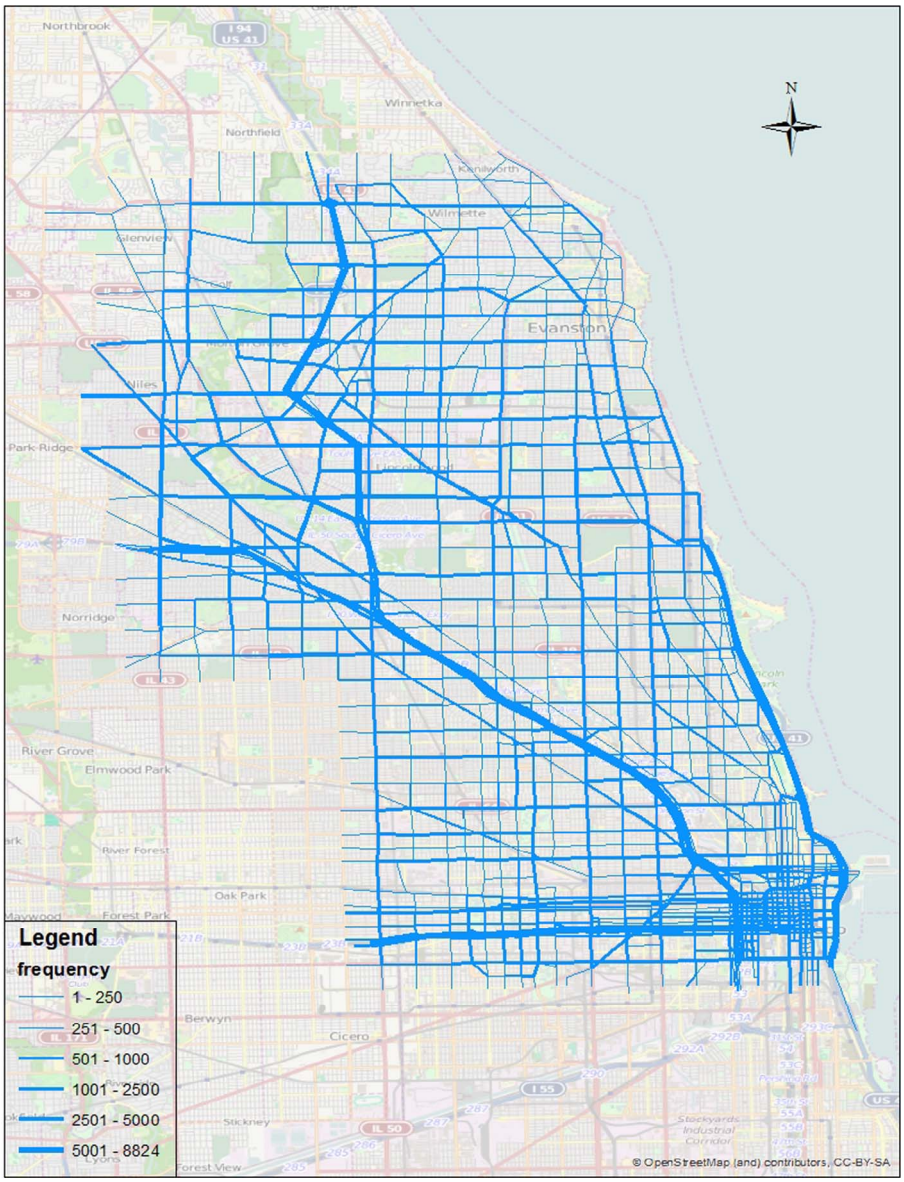


Fig. 8. Frequency of vehicle trajectory paths.

5.3. Application and evaluation

Two groups of simulation tests are conducted and presented in this section. First, the detailed carpooling information (including departure time, origin and destination) is not available in the back-end system for ride-matching algorithm; the system can only track the vehicles' locations. From the perspective of network management, these tests help examine the sensitivity of traffic performance with respect to the percentage of shared trips. In another scenario, where carpooling information is available for the system, the tests are designed for experimental carpooling cases to increase the applicability of carpool programs in the real world from the user perspective.

5.3.1. Tests without carpooling info

In this scenario, the departure and arrival information is not available for ride matching. Instead, the vehicles are matched according to the detected routes and clustering results. The effect on traffic performance is analyzed under different levels of carpooling participation (20%, 60%, and 100%). Note that the carpooling vehicles are only selected from the six clusters mentioned above where they are sharing over 70% of their routes (i.e., clusters 1, 2, 3, 5, 12, and 18). The total number of vehicles to be considered is 1472 vehicles during the morning peak hours 6:30 to 8:30.

Theoretically, it is assumed that the averaged carpool size is 2, and thus the number of vehicles for each potential route will

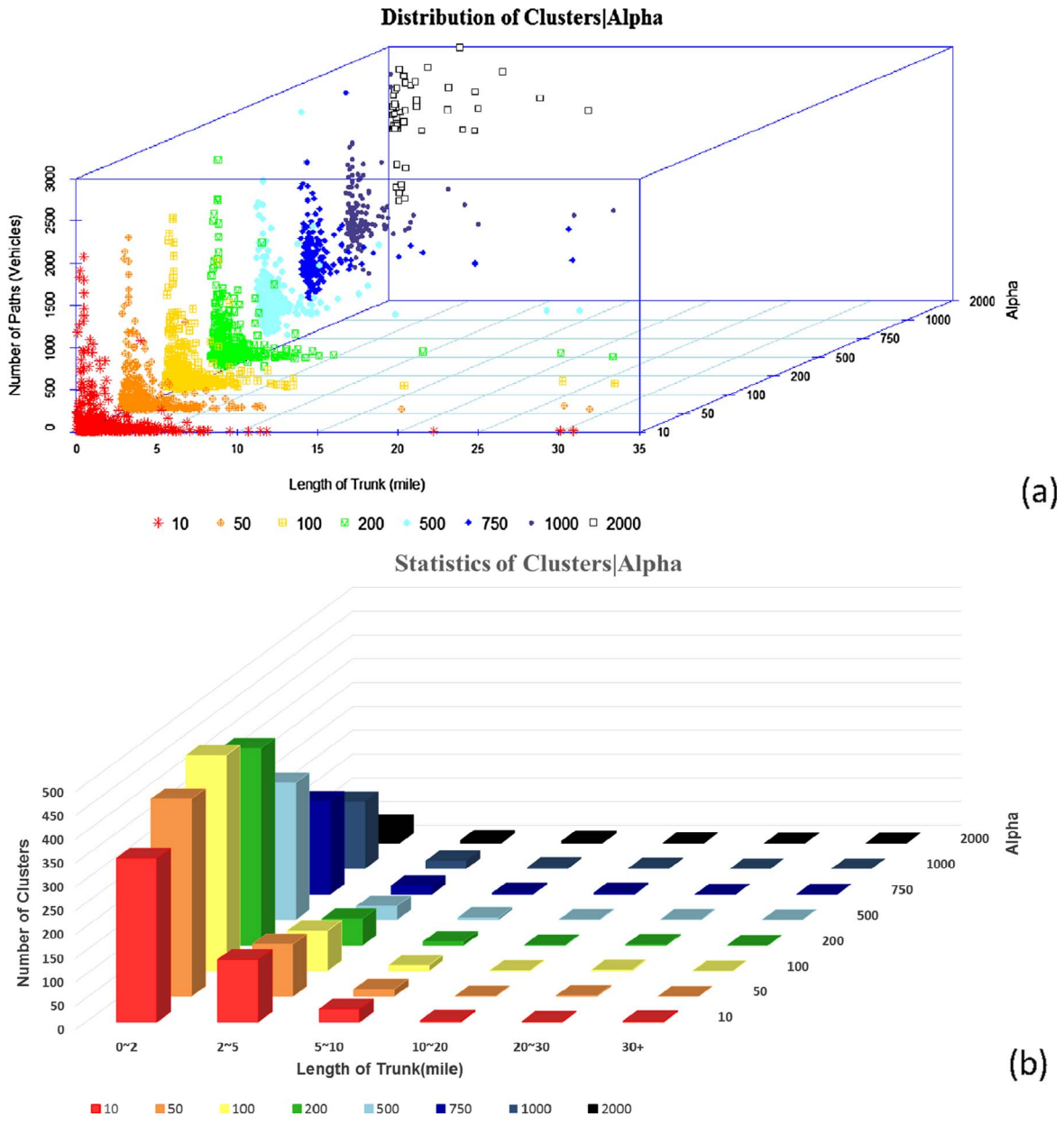


Fig. 9. The relationship between α and clustering results.

decrease by 10%, 30%, and 50% due to the carpooling behavior. Experimentally, the corresponding vehicles are chosen from the selected 6 routes in the clustering results.

According to the summary in Table 2, the number of vehicles is reduced by less than 1%, even though 100% of clustered vehicles are assumed to take carpools from the selected 6 routes. However, in spite of this small reduction, the total travel time decreases more than 10,000 h, which covers 2.35% of the baseline scenario. Considering the high occupation of carpool vehicles and the monetary value of saved time for each individual, the significance of carpool programs is apparent.

5.3.2. Tests with carpooling info

From a users' perspective, the carpool vehicles that can cover origins and destinations with preferred departure times are the key to promote carpool commuting programs. Rather than using randomly selected vehicles, the following three rules are designed to select and combine vehicles into carpool groups. The rules can be updated when participants have different preferences of departure time, walking distance, car type and size of carpooling group.

- Rule 1: Within the same cluster, the vehicles/routes that share similar OD pairs (origins/ destinations vary within 0.15 mile or 3-

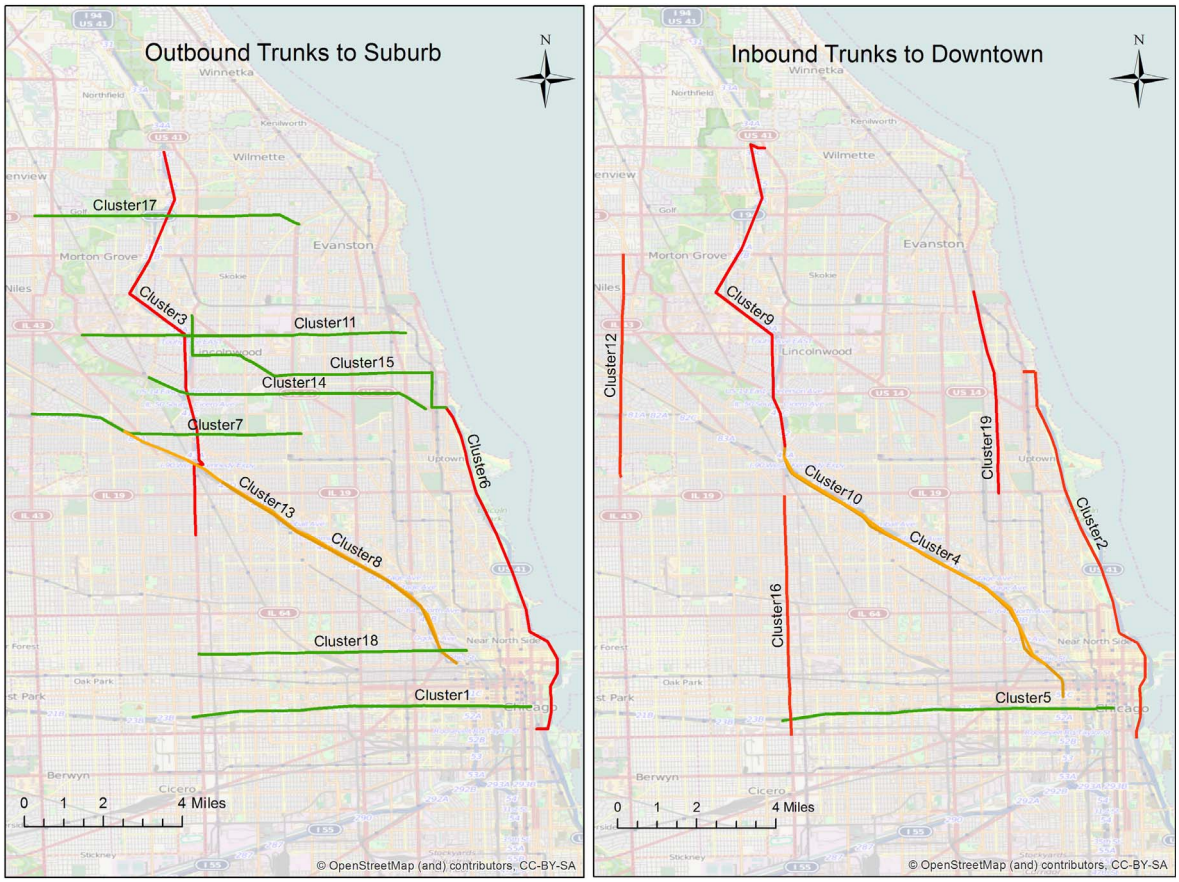


Fig. 10. Trunks longer than 5 miles with at least 100 vehicles in each trunk.

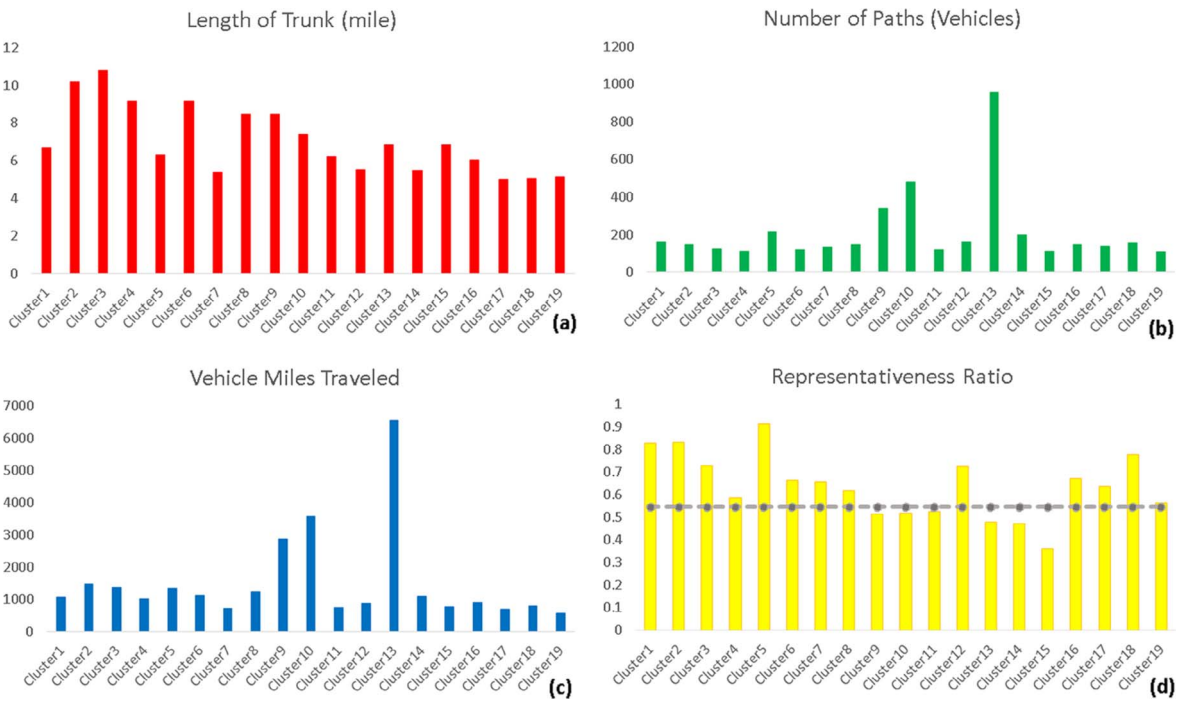


Fig. 11. Features of clusters.

Table 2
Summary of Simulation Results.

Carpooling Participation	Overall Vehicles	Percent Decrease	Average Travel Time (min)	Percent Decrease	Total Travel Time (h)	Percent Decrease
0 (baseline scenario)	788,584	0	34.60	0	454,812.66	0
20%	788,438	0.02%	34.47	0.40%	452,896.78	0.42%
60%	788,142	0.06%	33.88	2.10%	445,009.69	2.16%
100%	787,848	0.09%	33.82	2.26%	444,136.09	2.35%

min walking distance) and schedules (departure time differs within 5 min) are the candidates to be combined into one group.

- Rule 2: Given that the commuting vehicles are usually four-cylinder cars with 5 available seats (including the driver), the number of vehicles to be combined (the group size) for each route should be fewer than 5. This rule is based on the assumption that, except the driver, there are no extra passengers in the vehicle to be grouped.
- Rule 3: Following Rule 2, if more than 5 vehicles become candidates from Rule 1, the group is divided into sub-groups according to a more accurate departure time. If any sub-group still has more than 5 vehicles, then the vehicle candidates are to be equally divided into more sub-groups.

The selected 6 clusters (with representative ratio larger than 0.7 in Fig. 11) are filtered under these rules. In Fig. 12, the horizontal axis represents the selected 6 clusters, and the vertical axis denotes the number of groups under Rule 1. Each column has several parts marked by different colors, representing the number of vehicles within each group (i.e., group size). For example, cluster 1 has 10 groups (in orange), each of which has 2 vehicles sharing similar origin(s) and destination(s), and 6 groups (in grey) with 3 vehicles sharing the same thing, and so forth.

One may wonder whether it is consistent and reasonable to have rule 2 and rule 3 that up to 5 persons can carpool in one vehicle, as the assumption for tests without carpool info above is that the carpool size is 2. In this scenario, the size of group is also determined by the origin, destination and departure time. The average group size is 1.988; for each selected cluster, we have the size of 2.09, 1.90, 1.41, 2.76, 2.38 and 1.55. Therefore, the assumption of group size as 2 above is not contrast with the rules here.

With the filtered groups shown in Fig.12, Rules 2 and 3 are applied to selected vehicles to be combined. Fig.13 gives detailed information regarding the number of vehicles in every potential ride-sharing route before (orange bars) and after (blue bars) vehicle combination and assignment is conducted. It is worth mentioning that potential routes may vary from the number of groups in the given cluster before and after vehicle combination. Due to the rules for departure time, the number of potential carpool routes would not exceed the number of groups in each cluster. For example, cluster 1 has 31 groups in Fig. 12 and 30 potential routes in Fig. 13, since there are vehicles that cannot be merged into a carpool vehicle due to the departure time. To be specific, in cluster 1, Vehicle ID 316421 and ID 438517 share the same origin and destination, but their departure time varies as long as 40 min. Similarly, it is not guaranteed that the vehicles that have been categorized into groups (in Fig. 12) will be definitely combined with other vehicles, since some vehicles must be kept alone due to special departure times. For example, in cluster 12, there are 25 vehicles in potential route ID 2 before vehicle combination. If the combination of vehicles were optimized, the number after combined vehicles could drop to 5, fewer than the current 9 vehicles. There are 4 vehicles with distinct departure times, and they were not possible to categorize into any sub-groups; the remaining 21 vehicles were divided into 5 sub-groups subject to the constraint of departure time and group size.

The traffic assignment with DAG routes for carpooling is simulated with the input file, where the combined vehicles have been updated. Table 3 summarizes the results from DAG routes simulation and compares with the baseline scenario and the best carpooling participation scenario (100% participation). The overall vehicle number of a DAG scenario is not the minimum of the three, however, the improvement of travel time is much more than that of the 100% participation scenario. The huge improvement comes from the maximal combination of vehicles and thus decreases the probability of vehicles gathering on the same road simultaneously.

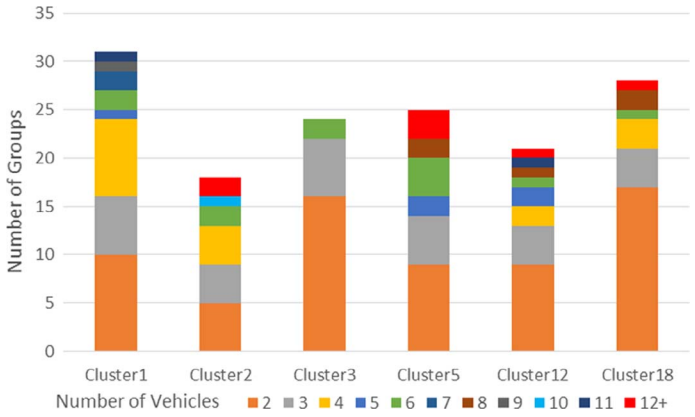


Fig. 12. Groups of vehicles under rules for origin and destination.

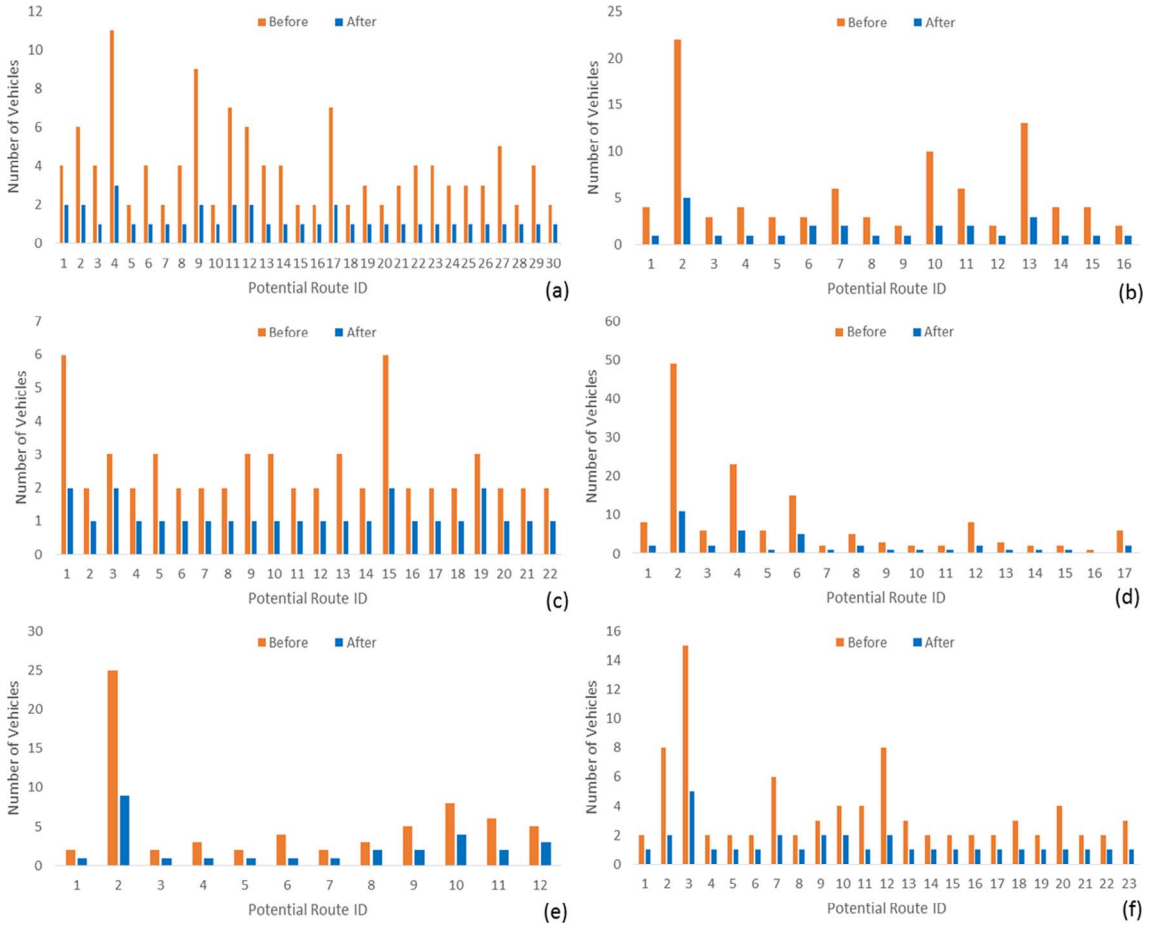


Fig. 13. Number of vehicles in potential routes for each cluster: (a) cluster 1; (b) cluster 2; (c) cluster 3; (d) cluster 5; (e) cluster 12; (f) cluster 18.

Table 3
Comparison of DAG Routes and Other Cases.

Scenario	Overall Vehicles	Percent Decrease	Average Travel Time (Min)	Percent Decrease	Total Travel Time (Hr)	Percent Decrease
0 (baseline scenario)	788584	0	34.60	0	454812.66	0
100% (best scenario in Table 2)	787848	0.09%	33.82	2.26%	444136.09	2.35%
DAG routes	788185	0.05%	31.89	7.83%	418887.85	7.90%

5.4. Performance measurement

One may wonder what if TOPOSCAN is compared with NEAT on the same input. To address this concern, several runs of the algorithm over input dataset of growing size from 5000 to 50000 are performed to verify the performance of the proposed algorithm. To cover different traffic conditions, including traffic flow, density and speed, varying with time, the samples for tests are randomly selected around different departure timestamps during the peak hours. All the experiments are conducted on the server machine with Intel(R) Xeon(R) CPU X5482 of 3.2GHZ and 3.19GHZ and installed memory (RAM) of 32 GB. The program to implement the algorithm is written from scratch in Python 2.7.

The data structure for opt-NEAT is not informed in the reference. It is not easy to have a specific tree structure for the path set with varied starting and ending points adopted for NEAT. The topology-based tree structure (using for TOPOSCAN) is also applicable in the experimental comparison as NEAT is trying to search the neighboring junctions. However, the units for clustering of the two algorithms are different. TOPOSCAN is to find the shared portion of trajectories (i.e. the clusters of sub-trajectories), and NEAT is to cluster the trajectory vector as a whole. Therefore, the clustering results are not completely comparable from the practical view. We only have tests to show the efficiency comparison between the algorithms.

A summary of execution times is displayed in Fig. 14. The later vehicle departs within morning peak hours, the more execution

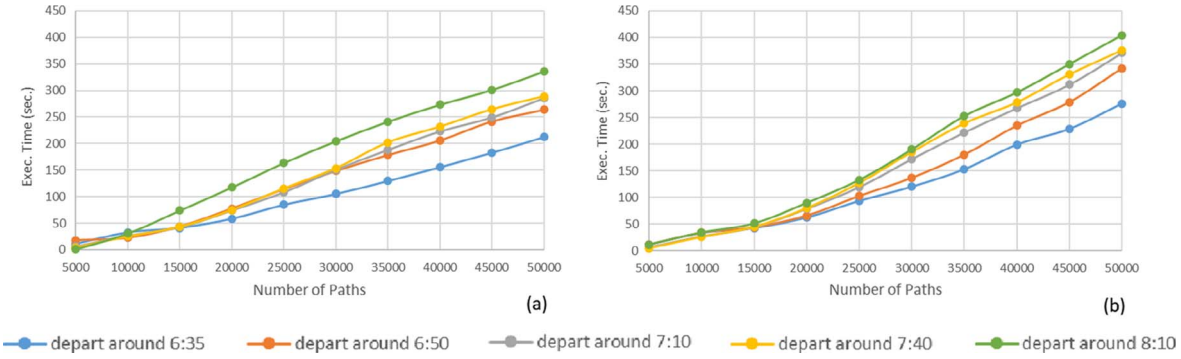


Fig. 14. Performance Measurement (a) TOPOSCAN; (b) NEAT.

time takes. This confirms the increasing complexity of the traffic conditions reflected from the vehicle trajectories when traffic start getting congested in the morning peak. The increasing execution time is positively correlated with the increasing size of path set. The curves of NEAT grow more than TOPOSCAN with larger dataset and more complex traffic conditions, which is due to the increasing computation for distance measurement in NEAT.

6. Conclusion and future work

With the goal of building an environmentally sustainable urban area, clustering vehicle trajectories provides an effective method to identify potential routes for carpooling participants. The modeling and simulation framework proposed in this paper is capable of helping achieve this goal and evaluating the traffic state with ride-sharing behaviors. The new ride-matching method proposed in this paper, taking advantage of various data sources and information processing technologies, opens up new and promising avenues to promote carpool programs.

One of the main contributions of this paper is the new data-driven based ride-matching method, which tracks personal preferences of route choices and travel patterns of commuting behaviors (recurring demand) instead of occasional trips (non-recurring demand) with a ride-sharing mode. The extracted trajectory patterns provide valuable information for building a library of route choice patterns for travel behavior researchers in the Chicago area. Inclusion of ride-sharing programs into traffic demand management strategies would offer promising opportunities for congestion relief and travel time reduction.

One may have concerns about the incomplete and delayed information and biased samples of the overall travel pattern of the commuter population obtained from social networks. To address this issue, the well-calibrated travel demand is preferred to make an estimation of the distribution of the commuter population. Besides, the proposed method is to provide a potential solution for carpool programs given any obtainable and reliable trajectory dataset. We are trying to make a good match for the potential participants who may be willing to share the information.

With future development, this study can be extended to the real-time ride-sharing system by incorporating the trajectory-based ride-matching method with dynamic routing and up-to-date customized preference settings. The extension with dynamic routing is necessary, since travelers in the real world may come from different origins and travel to different destinations rather than what is examined in the DAG experiment. Some commuters may be willing to share common parts of their routes and merge at some locations. Therefore, it is desirable to conduct experiments with combined ODs and take into account the rerouting algorithms for picking up and dropping off passengers. Furthermore, with the growing accessibility to real-time vehicle trajectory data, the clustering process can be applied and tested in a more realistic situation.

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. 223, 295–303.
Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J., 1999. OPTICS: ordering points to identify the clustering structure. ACM Sigmod Record, ACM, pp. 49–60.
Bahbouh, K., Morency, C., 2014. Encapsulating and visualizing disaggregated origin-destination desire lines to identify demand corridors. Transport. Res. Rec. J. Transport. Res. Board 162–169.
Bahbouh, K., Wagner, J.R., Morency, C., Berdier, C., 2015. TraClus-DL: A Desire Line Clustering Framework to Identify Demand Corridors. In: Transportation Research Board 94th Annual Meeting.
Ben-Akiva, M., Atherton, T.J., 1977. Methodology for short-range travel demand predictions: analysis of carpooling incentives. J. Transp. Econ. Policy 224–261.
Buliung, R., Soltys, K., Habel, C., Lanyon, R., 2009. Driving factors behind successful carpool formation and use. Transport. Res. Rec. J. Transport. Res. Board 31–38.
Camargo, S.J., Robertson, A.W., Gaffney, S.J., Smyth, P., Ghil, M., 2007a. Cluster analysis of typhoon tracks. Part I: general properties. J. Clim. 20, 3635–3653.
Camargo, S.J., Robertson, A.W., Gaffney, S.J., Smyth, P., Ghil, M., 2007b. Cluster analysis of typhoon tracks. Part II: large-scale circulation and ENSO. J. Clim. 20, 3654–3676.
Chan, N.D., Shaheen, S.A., 2012. Ridesharing in North America: past, present, and future. Transp. Rev. 32, 93–112.
Chen, Z., Shen, H.T., Zhou, X., 2011. Discovering popular routes from trajectories. In: 2011 IEEE 27th International Conference on Data Engineering (ICDE), IEEE, pp. 900–911.
Cheng, Z., Caverlee, J., Lee, K., 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ACM, pp. 759–768.
Daniels, P., 1981. Vehicle sharing for the journey to work by office employees. Transport. Res. Part A: Gener. 15, 391–398.

Dumas, J.S., Dobson, R., 1979. Linking consumer attitudes to bus and carpool usage. Transport. Res. Part A: Gener. 13, 417–423.
Eash, R.W., Swanson, A., Kaplan, M., 1974. An analysis and evaluation of alternate schemes to increase auto occupancy. Transp. Res. 8, 335–341.
Ester, M., Kriegel, H.-P., Sander, J., Xu, X, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd, pp. 226–231.
Fellows, N., Pitfield, D., 2000. An economic and operational evaluation of urban car-sharing. Transport. Res. Part D: Transp. Environ. 5, 1–10.
Gaffney, S., Smyth, P., 1999. Trajectory clustering with mixtures of regression models. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego, California, USA: ACM.
Ghoseiri, K., 2013. Dynamic Rideshare Optimized Matching problem. PhD Dissertation at University of Maryland.
Guo, D., Zhu, X., Jin, H., Gao, P., Andris, C., 2012. Discovering spatial patterns in origin-destination mobility data. Trans. GIS 16, 411–429.
Habib, K.M.N., Tian, Y., Zaman, H., 2011. Modelling commuting mode choice with explicit consideration of carpool in the choice set formation. Transportation 38, 587–604.
Hall, R.W., Qureshi, A., 1997. Dynamic ride-sharing: theory and practice. J. Transport. Eng. 123, 308–315.
Han, B., Liu, L., Omiecinski, E., 2012. NEAT: road network aware trajectory clustering. Distributed Computing Systems (ICDCS). In: IEEE 32nd International Conference on, 2012. IEEE, pp. 142–151.
Hartigan, J.A., Wong, M.A., 1979. Algorithm AS 136: a k-means clustering algorithm. Appl. Stat. 100–108.
Herbawi, W.M., Weber, M., 2012. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In: Proceedings of the 14th Annual Conference On Genetic And Evolutionary Computation, ACM, pp. 385–392.
Huang, H.-J., Yang, H., Bell, M.G., 2000. The models and economics of carpools. Ann. Reg. Sci. 34, 55–68.
Jayakrishnan, R., Mahmassani, H.S., Hu, T.Y., 1994. An evaluation tool for advanced traffic information and management systems in urban networks. Transport. Res. Part C: Emerg. Technol. 2 (3), 129–147.
Katzev, R., 2003. Car sharing: a new approach to urban transportation problems. Analyses Soc. Issues Public Policy 3, 65–86.
Kharrat, A., Popa, I.S., Zeitouni, K., Faiz, S., 2008. Clustering Algorithm for Network Constraint Trajectories. Headway in Spatial Data Handling. Springer, Heidelberg.
Kim, J., Mahmassani, H.S., 2015. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. Transport. Res. Part C: Emerg. Technol. 59, 375–390.
Kowshik, R.R., Jovanis, P.P., Kitamura, R., 1995. Evaluation of the Sacramento-area Real-time Rideshare Matching Field Operational Test: Final Report, California PATH Program, Institute of Transportation Studies, University of California at Berkeley.
Lee, J.-G., Han, J., Whang, K.-Y., 2007. Trajectory clustering: a partition-and-group framework. Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ACM, pp. 593–604.
Levin, I.P., 1982. Measuring tradeoffs in carpool driving arrangement preferences. Transportation 11, 71–85.
Li, J., Embry, P., Mattingly, S.P., Sadabadi, K.F., Rasmidatta, I., Burris, M.W., 2007. Who chooses to carpool and why?: examination of Texas carpoolers. Transport. Res. Rec.: J. Transportat. Res. Board 2021, 110–117.
Mahmassani, H.S., 2001. Dynamic network traffic assignment and simulation methodology for advanced system management applications. Networks Spatial Econ. 1 (3), 267–292.
Mahmassani, H.S., Sbayti, H., 2009. DYNASMART-P Version 1.6 User's Guide. Transportation Center, Northwestern University.
Palma, A.T., Bogorny, V., Kuijpers, B., Alvares, L.O., 2008. A clustering-based approach for discovering interesting places in trajectories. In: Proceedings of the 2008 ACM Symposium on Applied Computing. ACM, pp. 863–868.
Quddus, M.A., Ochieng, W.Y., Noland, R.B., 2007. Current map-matching algorithms for transport applications: state-of-the art and future research directions. Transport. Res. Part C: Emerg. Technol. 15, 312–328.
Quddus, M.A., Ochieng, W.Y., Zhao, L., Noland, R.B., 2003. A general map matching algorithm for transport telematics applications. GPS Solut. 7, 157–167.
Sadahiro, Y., Lay, R., Kobayashi, T., 2013. Trajectories of moving objects on a network: detection of similarities, visualization of relations, and classification of trajectories. Trans. GIS 17, 18–40.
Shaheen, S.A., Cohen, A.P., Roberts, J.D., 2006. Carsharing in North America: market growth, current developments, and future potential. Transport. Res. Rec.: J. Transport. Res. Board 1986, 116–124.
Shaheen, S.A., Mallery, M.A., Kingsley, K.J., 2012. Personal vehicle sharing services in North America. Res. Transport. Bus. Manage. 3, 71–81.
Train, K., McFadden, D., 1978. The goods/leisure tradeoff and disaggregate work trip mode choice models. Transp. Res. 12, 349–353.
Vasudevan, M., Wunderlich, K., 2013. Analysis, Modeling, and Simulation (AMS) Testbed Framework for Dynamic Mobility Applications (DMA) and Active Transportation and Demand Management (ATDM) Programs. US Department of Transportation, Research and Innovative Technology Administration.
Won, J.-I., Kim, S.-W., Baek, J.-H., Lee, J., 2009. Trajectory clustering in road network environment. Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on, IEEE, pp. 299–305.
Yelchuru, B., Singuluri, S., Rajiwade, S., 2013. Active Transportation and Demand Management (ATDM) Foundational Research. US Department of Transportation, Research and Innovative Technology Administration.