




# Efficient and robust data augmentation for trajectory analytics: a similarity-based approach

Dan He<sup>1</sup> · Sibow Wang<sup>2</sup>  · Boyu Ruan<sup>1</sup> · Bolong Zheng<sup>3</sup> · Xiaofang Zhou<sup>1,4,5</sup>

Received: 4 April 2018 / Revised: 5 May 2019 / Accepted: 15 May 2019 /  
Published online: 28 May 2019  
© Springer Science+Business Media, LLC, part of Springer Nature 2019

## Abstract

Trajectories between the same origin and destination (OD) offer valuable information for us to better understand the diversity of moving behaviours and the intrinsic relationships between the moving objects and specific locations. However, due to the data sparsity issue, there are always insufficient trajectories to carry out mining algorithms, e.g., classification and clustering, to discover the intrinsic properties of OD mobility. In this work, we propose an efficient and robust trajectory augmentation approach to construct sizeable qualified trajectories with existing data to address the sparsity issue. The high-level idea is to concatenate existing trajectories to reconstruct a sufficient number of trajectories to represent the ones going across the OD pair directly. To achieve this goal, we first propose a transition graph to support efficient sub-trajectories concatenation to tackle the sparsity issue. In addition, we develop a novel similarity metric to measure the similarity between two set of trajectories so as to validate whether the reconstructed trajectory set can well represent the original traces. Empirical studies on a large real trajectory dataset show that our proposed solutions are efficient and robust.

**Keywords** Trajectory sparsity · Trajectory concatenation · Trajectory augmentation · Trajectory set similarity

## 1 Introduction

With the proliferation of GPS-enabled devices, a significant increasing volume of trajectories have been collected, which record the mobility of moving objects, e.g., vehicles. Trajectory data offers valuable information for us to better understand the moving objects and the intrinsic relationship between the moving objects and specific locations. This fosters plenty of applications in location-based social networks and intelligent transportation systems, e.g., personalised routing service. Given an origin and a destination, in general, most existing online navigation services, e.g., Google Map, only provide the shortest path or the fastest route from  $o$  to  $d$ . But the personalised routing service will mine the trajectories

---

✉ Sibow Wang  
swang@se.cuhk.edu.hk

between the same origin and destination, and further helps offer valuable information on the options that travelers might be interested in, e.g., the quiet route, the route with fewer humps/traffic lights. In the literature, there exists a plethora of work [2, 13, 26] focusing on how to do trajectory mining given OD pairs.

However, most of these mining algorithms only focus on the mining part and do not consider the *data sparsity* issue. To explain, given an origin and a destination, the number of trajectories could be found from a trajectory dataset might be notably small, even though the volume of the entire trajectory set is considerably huge, which hinders the effectiveness of the mining algorithms. From a statistic on a real dataset containing 190K location points (190K × 190K pairs) and 2000K trajectories, only 1.5% pairs of points contain trajectories passing from one point to another. Figure 1a summarizes how many OD pairs have a number  $x$  of trajectories going across the OD pair. As we can see, the number of OD pairs drops sharply with the increase of the number  $x$  of trajectories traversing the OD pair, and only a small number of OD pairs contain large sets of trajectories passing by.

In particular, data sparsity is a typical issue in data mining, which will significantly affect the performance of corresponding mining algorithms. In the literature, the research studies [5, 6, 27, 28] on trajectory mining mainly apply trajectory concatenation to address the sparsity problem. They augment OD trajectories with existing sub-trajectories, namely, if there is no/insufficient trajectory directly traversing over an OD pair, they join multiple sub-trajectories to represent the traces from the origin to the destination. For example, in Figure 1b, suppose that there are only four historical trajectories  $T_1, T_2, T_3, T_4$  on this portion of the road network in a dataset, where the OD pair is set to be  $p_1$  and  $p_{14}$ . Clearly, there is no trajectory starting from  $p_1$  to  $p_{14}$  that can be directly extracted from this dataset. A notable solution is to concatenate the trajectory ( $T_2$ ) traveling from  $p_1$  to  $p_{11}$  with the one ( $T_4$ ) traveling from  $p_{11}$  to  $p_{14}$  to represent the one from  $p_1$  to  $p_{14}$ . Here,  $p_{11}$  is identified as the *intermediate transition node (ITN)* for trajectory concatenation. As we can see, the concatenation is a straightforward and feasible solution for trajectory augmentation.

Nevertheless, the major challenge is how to form the concatenation of trajectories, which further consists of threefold issues. 1) Firstly, the retrieval of possible ITN could be complicated. Particularly, the number of intermediate transition nodes required for specific

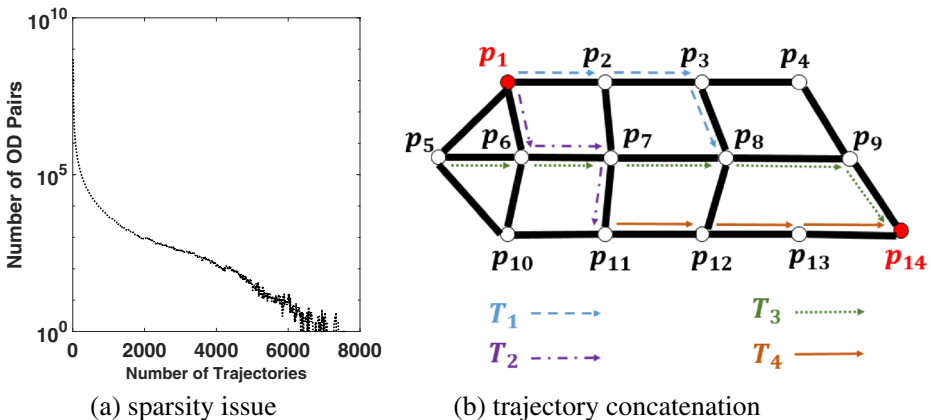


Figure 1 illustration of the sparsity issue and an example of trajectory concatenation

OD pairs to generate concatenation varies according to the location relationship between the OD pairs. For instance, in Figure 1b, given the OD pair  $p_5$  and  $p_{14}$ , we can obtain the OD trajectory  $T_3$  which directly traverses from  $p_5$  to  $p_{14}$  with zero ITN, or we can concatenate the trajectories from  $p_5$  to  $p_7$  ( $T_3$ ), from  $p_7$  to  $p_{11}$  ( $T_2$ ) and then from  $p_{11}$  to  $p_{14}$  ( $T_4$ ), with 2 ITNs ( $p_7, p_{11}$ ). 2) Besides, when given a fixed number of ITNs, there could be multiple possible ITNs for trajectory concatenation. For example, in Figure 1b, by setting the OD pair as  $p_1$  and  $p_{14}$ , and the number of ITN as 1, the possible ITNs for concatenation could be  $p_6, p_7, p_8$ , and  $p_{11}$ . Concatenate trajectories according to all possible ITNs is considerably time-consuming and unnecessary. Thus a challenging issue is to identify which points should be chosen to form the concatenation. 3) Furthermore, after the trajectory concatenation, whether the concatenated trajectories are qualified to represent the one traversing directly from the origin to the destination is unclear, and no existing work on trajectory concatenation provides a robustness validation for the concatenated trajectories.

In our previous work [9], we proposed a novel similarity measure for trajectory sets, which to the best of our knowledge was the first to explore the similarity between two sets of trajectories. All existing trajectory similarity measures only focus on two individual trajectories rather than on two trajectory sets. In this paper, we propose an efficient and robust trajectory augmentation approach to address the data sparsity issue by concatenating the existing sub-trajectories to represent the trajectories over specific OD pairs. In order to retrieve the possible ITN w.r.t. a given OD pair efficiently, we build a transition graph based on the trajectory data, which indicates the connectivity relationship between transition nodes. After obtaining the ITNs candidates, we provide a few criteria for the selection of ITNs to concatenate trajectories. For validating the quality of the concatenated trajectories, we apply our EMD-based trajectory set similarity measure [9] to validate the robustness of our concatenation approach, compared with two existing strategies: *sweep-and-expand* [6] and *popularity-based* [5]. The contributions of our work can be summed up as follow:

- We propose a novel and robust similarity measure [9] for two sets of trajectories, derived from the Earth Mover’s Distance [19] (EMD), which measures the distance of the spatial hits distribution between two trajectory sets. We further extend the EMD so as to cover the spatial, temporal, and sequence properties of trajectory sets.
- We develop a transition graph based on existing trajectory data on account of efficient retrieval of eligible intermediate transition nodes regarding to a specific OD pair to form the concatenation set of trajectories.
- We propose effective heuristic solutions for intermediate transition nodes selection, which not only accelerates the intermediate transition nodes retrieval but also guarantee the quality of the concatenation set of trajectories.

The remainder of this paper is organised as follows. In Section 2, we introduce the related work and their limitations on our studied problem. Next, in Section 3, we present the preliminary concepts and the overview of our framework. The technical details of our approach will be demonstrated in Sections 4 and 5, followed by the experimental study in Section 6. Finally, we conclude and present potential future studies in Section 7.

## 2 Related work

In this section, we present the existing research work related to our study. In Section 2.1, we present existing work on the similarity measures for spatial-temporal data, followed by

the similarity measure for distributions in Section 2.2. Finally, we present existing work on dealing with trajectory sparsity in data mining area and the corresponding solutions in Section 2.3.

## 2.1 Trajectory similarity measures

Measuring the similarity or the distance between two entities is a crucial and fundamental procedure in data mining and knowledge discovering. As for trajectory data, a plethora of research work [3, 4, 17, 20, 23, 30] has studied how to define similarity measures on trajectory data for over decades. Several surveys [21, 24] have provided precise and systematic classification on similarity measures according to diverse concerns on the characteristic of trajectories. In particular, Euclidean Distance (ED) [20] is a straightforward and intuitive similarity measure with low calculation cost, which computes the distance between the corresponding matched points from two trajectories. However, the ED measure is sensitive to noise. Vlachos et al. introduced the The Longest Common Subsequence (LCSS) measure [23], to measure the similarity of two trajectories, which is much robust to noise than the ED measure. Chen et al. proposed two versions of similarity measure for two trajectories based on Edit Distance: Edit Distance on Real Sequences (EDR) [3] and Edit Distance with Real Penalty (ERP) [4], which are widely used in trajectory similarity measure. Kruskal et al. [11] and Kassidas et al. [10] proposed to use Dynamic Time Warping (DTW) [30], traditionally applied on time series analysis since decades ago, as a trajectory measure. Apart from the above similarity measures, there are many other measures introduced for trajectory dataset, e.g., Dynamic Time Warping (DTW) [30], One Way Distance (OWD) [14], etc.

Nevertheless, all the aforementioned similarity measures mainly focus on individual trajectory. No existing similarity measure is introduced for two sets of trajectories. A naive extension from individual trajectory similarity measure to the trajectory set measure is to calculate the pairwise distances between each pair of trajectories from two different sets respectively. However, such similarity measure is not only time-consuming, but also ineffective. For two identical sets of trajectories it might result in a large distance value by calculating the pairwise distances, while intuitively the distance should be zero.

## 2.2 Distribution similarity measure

By extracting some representative features from the corresponding sets and transforming them into distributions, the distribution similarity measure can be easily extended to measure the similarity between two sets of objects. In measuring the similarity between two distributions, which are usually represented by histograms, the Mankowski-Form Distance [22] is the most straightforward one, by simply calculating the  $L_1$ -norm distance between each pair of bins. Kullback-Leibler Divergence [12] and Jeffrey Divergence [18] are two distribution measures derived from information theory, that measure how inefficient on average it would be to code one histogram using the other as the code-book. Most of the distribution measures require that the number of bins in two histograms to be the same, and they do not consider the ground distance between each pair of bin. The Earth Mover's Distance [19] is a well known distribution similarity measure, which is proposed for the image retrieval. Informally, interpreting the distributions as two different ways of piling up a certain amount of dirt over the region, the Earth Mover's Distance is the minimum work flow of moving one pile into the other. It takes the ground distance between different bins into account and the numbers of bins in two histograms are unnecessary to be the same. Thus, we derive the EMD to develop our set-based trajectory similarity measure.

## 2.3 Trajectory sparsity problem

Trajectory data usually spans across a geographical area such as a city, which renders the data concerning a specific region probably to be very sparse. Likewise, although given a trajectory database recording trajectories distributed over an extended period, e.g., one year, trajectories existing within a short period, e.g., one hour, are also likely to be rare. Trajectory sparsity problem would significantly reduce the effectiveness and efficiency of many trajectory applications, e.g., destination prediction [28], travel time estimation [29], and popular route discovery.

Destination prediction aims to find out the potential destination of a user while gives only part of trajectory information. A universal solution is to derive the probability of a location being the destination based on building a Hidden Markov Model (HMM) [7] on top of historical trajectory data [1]. However, data sparsity is an inevitable problem for the general technique. Xue et al. [28] proposed a Sub-trajectory Synthesis (SubSyn) algorithm to address the sparsity problem with the HMM for destination prediction, which first decomposes trajectories into sub-trajectories between two neighbour locations and then concatenates the sub-trajectories into "synthesised" trajectories. The corresponding sparsity issue differs from our work as we take destination as one of the input, while destination is the output of this problem. Regarding the traveling time estimation, which strives to evaluate the time cost of a path from one location to another based on historical trajectory data, the sparsity issue of the trajectory is also a challenging problem. Wang et al. [25] propose an efficient and effective model to estimate the travel time of a path with sparse trajectories. This model uses context-aware tensor decomposition approach to fill the missing data and find out the most optimal concatenation of trajectories for time estimation. The sparsity problem is solved by employing tensor factorization to fill the empty fields (traveling time) based on features of trajectories, which is unsuitable for the scenario that requires retrieving the actual trajectories.

The problem of data sparsity also appears in applications such as popular route discovery [5], which attempts to return the popular route concerning a pair of origin and destination. To solve this problem, it first builds a transfer network based on the trajectory data, and then retrieves the popular route based on the transfer network, which is formed by a sequence of edges whose multiplication popularity is maximum. There is a similar approach to address the sparsity issue (i.e., there is no reference trajectory in between a given OD pair) in personalized routing [6], which builds a global reference graph by concatenating the intermediate edges from the origin to the destination. Then it discovers a route from the reference graph that to the largest extent satisfies the user's preference. In general, the existing strategies for the sparsity problem in trajectory applications mainly focus on concatenating the sub-trajectories along the specific paths and filling up the sparsity with conceptual trajectory based on probability or popularity of the sub-paths. However, no existing work gives the robustness validation for their reconstructing trajectories, i.e., how qualified the reconstructed trajectories are to represent the potential mobility.

## 3 Overview

In this section, we first illustrate some formal concepts and definitions. Then we introduce the problem statement and the framework overview. The frequently used notations in this paper are shown in Table 1.

**Table 1** frequently used notations

Notation	Description
$G$	A road network
$T$	A trajectory
$T_{p_j, p_k}$	The sub-trajectory from $p_j$ to $p_k$
$S_{p_o p_d}$	An origin-destination set of trajectories
$ITN$	The intermediate transition node
$k$	The number of ITNs in one concatenation
$S_c(p_o, p_d, k)$	A concatenation set of trajectories
$sh(p)$	The spatial hit of road segment $p$
$S$	The spatial hit signature
$\delta$	The average delta duration
$\alpha$	The percentage of spatial character in EMDT
$G_t$	A transition graph
$\lambda$	The parameter for $G_t$ construction
$ASL$	The accumulative spatial length
$\tau$	The threshold of ASL for ITNs retrieval

### 3.1 Preliminary concepts

**Definition 1 (Road network)** A road network is defined as a graph  $G = (V, E)$ , where  $V$  is a set of intersection nodes on the road network and  $E$  is a set of road segments  $r \in E$ , such that  $r = (p_s, p_e)$  with  $p_s, p_e \in V$  being the two end nodes of  $r$ , denoted by  $\langle latitude, longitude \rangle$ .

**Definition 2 (Trajectory)** A raw trajectory of a moving object is usually recorded by a sequence of spatial-temporal points. Given a road network  $G = (V, E)$ , each spatial-temporal point from a trajectory can be mapped onto an intersection node of the road network appended with corresponding timestamp. Thus, a trajectory  $T$  can be represented by a sequence of time-ordered road network nodes, i.e.  $T = \{(p_1, t_1), (p_2, t_2), \dots, (p_{|T|}, t_{|T|})\}$ . Note that  $p_i$  here indicates the  $i$ -th nodes the trajectory  $T$  mapped onto the road network. The length of a trajectory  $T$  is the number of constituted nodes, notated as  $|T|$ . A sub-trajectory of  $T$  is a subsequence of road network nodes from  $T$ , denoted by  $T_{p_j, p_k} = \{(p_j, t_j), \dots, (p_k, t_k)\}$ , where  $1 \leq j < k \leq |T|$ .

**Definition 3 (Origin-destination set)** Given a trajectory database  $D$  and a pair of origin and destination nodes  $(p_o, p_d)$ , the set of trajectories/sub-trajectories traversing from  $p_o$  to  $p_d$  are notated as  $S_{p_o p_d}$ . Formally, we have  $S_{p_o p_d} = \{T_{p_j, p_k} | T \in D \wedge p_j = p_o \wedge p_k = p_d\}$ .

**Definition 4 (Concatenation set)** Given a sequence of  $k$  nodes on the road network  $\{p_1^c, p_2^c, \dots, p_k^c\}$ , denoted as Intermediate Transition Nodes (ITNs), and a pair of origin and destination nodes,  $p_o$  and  $p_d$ , we concatenate the sets of trajectories/sub-trajectories traveling from  $p_o$  to  $p_d$  by passing the intermediate transition nodes consecutively into a hybrid set  $S_{con}$  of trajectories. Formally, we have  $S_c(p_o, p_d, k) = S_{p_o p_1^c} \parallel S_{p_1^c p_2^c} \parallel \dots \parallel S_{p_{k-1}^c p_k^c} \parallel S_{p_k^c p_d}$ , where  $S_{p_o p_1^c} \neq \emptyset, S_{p_i^c p_{i+1}^c} \neq \emptyset (1 \leq i \leq k - 1)$  and  $S_{p_k^c p_d} \neq \emptyset$ .

The origin-destination set is the set of trajectories/sub-trajectories starting from  $p_o$  and ending at  $p_d$ , which is equivalent to the concatenation set of trajectories in the case of  $k = 0$ . Given an origin and a destination, to efficiently retrieve the trajectories going across the OD pair, a widely applied index structure is the inverted list, defined as follows.

**Definition 5 (Trajectory inverted list [8])** Each entry in the trajectory inverted list corresponds to a road intersection nodes, with the form of  $\langle p_{id}, T_{list} \rangle$ , where  $p_{id}$  indicates the intersection node and  $T_{list}$  is the trajectory position list that contains the trajectories occur on that node. The trajectory position list is formed by a list of  $\langle T_{info}, pos \rangle$ , where  $T_{info}$  consists of both trajectory id and the trajectory address indicating the block where  $T$  is stored and the *position* is the occurrence of the corresponding node on that trajectory.

Note that the  $T_{list}$  is sorted by the trajectory id and the entry list is sorted by the  $p_{id}$  for efficient search. Besides, all the trajectory sequences are stored consecutively in the ascending order of the trajectory ids on the disk. Also, the inverted index can be built by a single scan on the trajectory dataset. For each road intersection node that occurs on a trajectory, we create an entry consisting of the occurrence and trajectory information and append this entry to the corresponding trajectory position list of that node.

*Problem Statement:* The **Trajectory Augmentation Approach (TAA)** takes as input an origin  $p_o$ , and a destination  $p_d$ . It then returns sizeable Concatenation Sets  $S_c(p_o, p_d)$ , of trajectories, to augment the moving traces traversing from  $p_o$  to  $p_d$ .

### 3.2 Framework overview

Figure 2 demonstrates the framework of our trajectory augmentation approach, consisting of three components: trajectory retrieval, trajectory concatenation and similarity-based validation.

In the trajectory retrieval stage, raw trajectories are transformed into sequences of time-ordered road network intersection nodes by map matching [15], associated with the underlying road network data. Afterwards, an inverted index (Ref. Definition 5) is built based on the mapped trajectories, which develops the linking between road network nodes and trajectories traversing through. When given an OD pair of spatial nodes (if they are arbitrary points, find the nearest intersection nodes respectively), trajectory retrieval is to obtain a set of trajectories/sub-trajectories that start from  $p_o$  and end at  $p_d$ . With the inverted index, we can efficiently return a set of trajectories that traverse across the nodes  $p_o$  and  $p_d$ . Firstly, we load the corresponding trajectory position lists of  $p_o$  and  $p_d$  in the memory and scan both synchronously to find out each intersecting trajectory  $T$  (i.e.  $\{T \in T_{list}(p_o) \cap T_{list}(p_d)\}$ ). Further, the occurrence of  $p_o$  on the intersecting trajectory  $T$  should be earlier than that

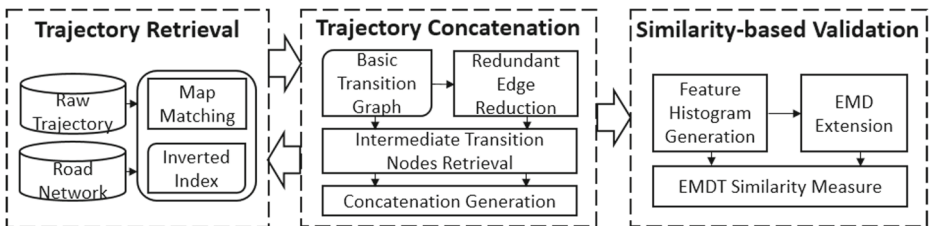


Figure 2 Framework Overview



of  $p_d$  (i.e.  $T(p_o).pos < T(p_d).pos$ ). This helps find out the Origin-Destination Set of trajectories for the given OD pair with the inverted index.

In our trajectory concatenation component, we construct a directed transition graph based on the trajectory data for intermediate transition nodes retrieval. Each edge on the graph indicates the connectivity relationship by the trajectory between two nodes. In order to improve the efficiency and reduce the redundancy, we eliminate some unnecessary edges of the graph. Employing the transition graph, we retrieve the eligible candidates of intermediate transition nodes. Further, we propose several criteria, which are based on the experimental study, to filter unnecessary intermediate transition nodes for concatenation formation.

In the similarity-based validation component, borrowing the idea of the Earth Mover's Distance, we propose an effective and robust similarity measure to calculate the similarity between two trajectory sets. Our proposed measure captures most characteristics of trajectories, e.g., spatial-temporal property, sequence (i.e., order and direction) information in individual trajectories. After obtaining the concatenation set of trajectories, we extract the features of the origin-destination set and each concatenation set respectively, followed by the signature generation for the corresponding feature. Note that the origin-destination set of trajectories is temporarily removed from the database before concatenation formation. Afterwards, we calculate the distance between the two trajectory sets according to our proposed trajectory set similarity measure. In terms of the computation of the Earth Mover's Distance, we utilise the algorithm proposed in [16], which is the fastest in the state of the art.

Next, we present the details of our framework component. We omit the discussion of our trajectory retrieval component, since it mainly applies existing solutions. Also, for the ease of exposition, we first present our similarity-based validation component assuming that two trajectory sets are given, which is the principle content of our previous work [9]. Then, we present the details on how to do trajectory concatenation aiming at finding eligible ITNs to reconstruct concatenation trajectory sets.

## 4 Similarity-based validation

Unlike other existing work that concatenates trajectories without any robustness validation and guarantee, in our work, we provide a trajectory set similarity measure to evaluate the similarity between the concatenated trajectories and the origin-destination ones. The concatenated trajectories with high similarity to the origin-destination set are verified as the ones with good quality to represent the potential traveling mobility from the origin to the destination. In this section, we first introduce a similarity measure to calculate the distance between two set of trajectories based on the Earth Mover's Distance (EMD). Then we extend the EMD-based measure to a more robust and effective measure which considers more trajectory characteristics, i.e, temporal and sequence information. In our paper, we use the solution proposed in [16] to calculate the Earth Mover's Distance.

### 4.1 EMD

The Earth Mover's Distance [19] is widely used in image retrieval, which measures the similarity for the distributions of features from two different sets of objects. Given two signatures, EMD calculates the minimum cost to transform one signature to another with the ground distance between each bin into consideration. To consider the similarity of two trajectory sets, intuitively, two similar sets of trajectories ( $i$ ) have higher probability traversing



across the same or near locations, (ii) go across the same node with similar probabilities. This motivates us to apply the *spatial hits* defined as follows to consider the similarity of two trajectory sets.

**Definition 6 (Spatial hit)** Given a set of trajectories  $\mathbb{T}$  and a node  $p$ , the spatial hit of  $p$  w.r.t.  $\mathbb{T}$ , notated as  $sh(p)$ , is defined as the number of occurrence of  $p$  in  $\mathbb{T}$ . Correspondingly, we build the spatial hit signature denoted by  $S = \{(p_1, w_1), (p_2, w_2), \dots, (p_n, w_n)\}$ , where  $\{p_1, p_2, \dots, p_n\}$  is the union set of road network nodes in  $\mathbb{T}$ , and  $w_i$  is the hit ratio for node  $p_i$ , formulated by

$$w_i = \frac{sh(p_i)}{\sum_{i=1}^n sh(p_i)} \tag{1}$$

The spatial hits captures the above mentioned intuition, and for two similar trajectory sets, the minimum cost to transform one spatial hits signature to another should be small. Hence, based on the spatial hit signature, we introduce the measurement of the distance between two sets of trajectories based on Earth Mover’s Distance as follows.

**Definition 7 (Earth Mover’s Distance (EMD) [19])** Let two spatial hit signature  $S_a$  and  $S_b$  be notated as  $S_a = \{(p_{a,1}, w_1), (p_{a,2}, w_2), \dots, (p_{a,n}, w_n)\}$  and  $S_b = \{(p_{b,1}, u_1), (p_{b,2}, u_2), \dots, (p_{b,m}, u_m)\}$ , with  $p_{a,i}$  (resp.  $p_{b,j}$ ) and  $w_i$  (resp.  $u_j$ ) being the node from the set  $S_a$  (resp.  $S_b$ ) of trajectories and the corresponding hit ratio. Let  $\mathbf{D} = [d_{ij}]$  be the ground distance matrix where  $d_{ij}$  is the ground distance between  $p_{a,i}$  and  $p_{b,j}$ . We aim to find a flow  $\mathbf{F} = [f_{ij}]$  that minimizes the cost to match  $S_a$  to  $S_b$ , formulated as below:

$$EMD(S_a, S_b) = \min \left\{ \sum_{i=1}^n \sum_{j=1}^m d_{ij} f_{ij} \right\} \tag{2}$$

subject to the following conditions:

$$f_{ij} \geq 0 \quad 1 \leq i \leq n, 1 \leq j \leq m \tag{3}$$

$$\sum_{j=1}^m f_{ij} \leq w_i, \quad \sum_{i=1}^n f_{ij} \leq u_j \tag{4}$$

$$\sum_{i=1}^n w_i = \sum_{j=1}^m u_j = 1 \tag{5}$$

$$d_{ij} = \begin{cases} \frac{d(p_{a,i}, p_{b,j})}{d_{max}}, & d(p_{a,i}, p_{b,j}) < d_{max} \\ 1, & otherwise \end{cases} \tag{6}$$

Here,  $d(p_{a,i}, p_{b,j})$  usually refers to the Euclidean distance or the road network distance between two nodes. According to the experiment, the Euclidean distance is sufficient to display the spatial difference between trajectory sets. For simplification and efficiency, we set the default ground distance measure as the Euclidean distance. In addition, we set a large distance value  $d_{max}$  for normalisation, s.t. the EMD is bounded by  $[0, 1]$ .

The flow  $\mathbf{F} = [f_{ij}]$  is a matrix representing the hit ratio flowing from one signature to another for matching two spatial hit signatures. Constraint (3) confines the amount of  $p_{a,i}$  matched to  $p_{b,j}$  to be nonnegative. Constraint (4) limits the

total amount in  $S_b$  matched from  $p_{a,i}$  does not exceed  $w_i$ , and the total amount in  $S_a$  matched to  $p_{b,j}$  does not exceed  $u_j$ . Take the two spatial hit signatures in Figure 3 as example, where  $S_a = \{(p_1, 0.2), (p_2, 0.3), (p_3, 0.2), (p_4, 0.3)\}$  and  $S_b = \{(p_1, 0.3), (p_2, 0.2), (p_3, 0.3), (p_4, 0.2)\}$ . Given the ground distance matrix  $d_{ij}$  shown as Figure 3, we can obtain the flow with  $f_{11} = 0.2, f_{21} = 0.1, f_{22} = 0.2, f_{33} = 0.2, f_{43} = 0.1,$  and  $f_{44} = 0.2$ , such that the  $EMD(S_a, S_b)$  is minimum. Eventually, in this case,  $EMD(S_a, S_b) = 0.1$ .

However, since a trajectory set is a set of spatial-temporal sequences rather than a multi-set of spatial nodes, simply taking EMD as the measure is insufficient. It will discard the temporal / sequence information, and may result in biased results. Take an extreme case as example. Consider two set of trajectories where the first set  $S_1$  contains trajectories going from  $o$  to  $d$  and the second set  $S_2$  contains trajectories going from  $d$  to  $o$ . For each trajectory  $T_i$  in  $S_1$ , there exists a trajectory  $T'_i$  such that the nodes going across in  $T_i$  are the same as the one in  $T'_i$ , and vice versa. In this case, the EMD between these two sets should be zero. However, it is clear that the two sets of trajectories differ significantly and the EMD score cannot capture the difference as it ignores the temporal / sequence (i.e., the order and direction of the spatial nodes) information.

### 4.2 EMDT

To capture the temporal / sequence information of trajectories, we further propose the Earth Mover’s Distance on Trajectory, an improved version of the EMD to capture more characteristics of trajectory set. In particular, we take into account the time duration information and add it into the signatures. More formally, we define the average delta duration for a node as follow.

**Definition 8 (Average delta duration)** Given a signature  $S = \{(p_1, w_1), (p_2, w_2), \dots, (p_n, w_n)\}$  w.r.t. a set of trajectory  $\mathbb{T}$ , for each node  $p$ , we define the average delta duration

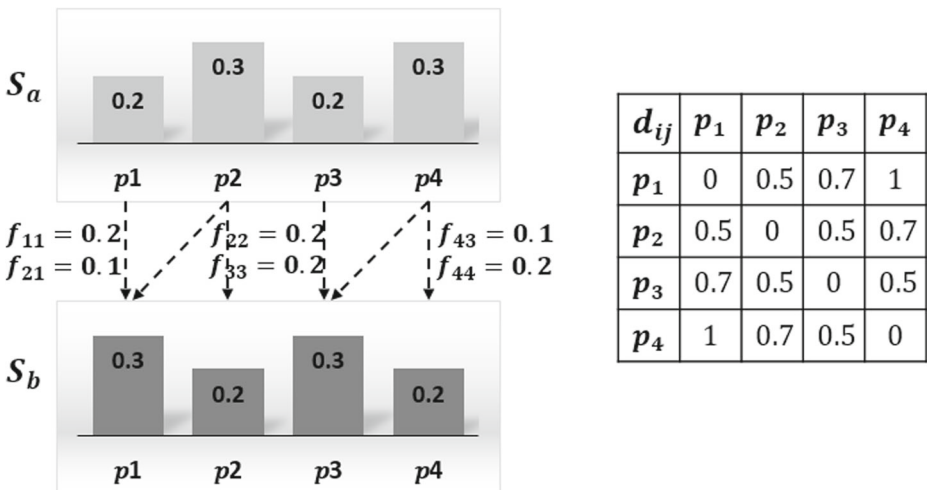


Figure 3 EMD Example

of  $p$ , denoted as  $\delta$ , to be the average time duration of  $p$  to the time of the first node in each trajectory in  $\mathbb{T}_p$  (where  $\mathbb{T}_p \subset \mathbb{T}$  is the set of trajectories containing  $p$ ), formulated by

$$\delta = \frac{1}{|\mathbb{T}_p|} \sum_{i=1}^{|\mathbb{T}_p|} (t_{T_i}(p) - t_{T_i}(p_1)) \tag{7}$$

where  $T_i \in \mathbb{T}_p$  is the trajectory traversed through  $p$ ,  $p_1$  is the first node in  $T_i$  and  $t_{T_i}(p)$  is the corresponding time stamp of  $p$  on trajectory  $T_i$ .

Note that the average delta duration indicates not only the temporal information of a signature but also reflects the sequence characteristic of trajectory nodes. This is because the delta duration represents an accumulative time used from the starting node of the trajectory to the corresponding node. For example, given 3 trajectories  $T_1 = \{(p_{11}, 1), (p_{12}, 3), (p_{13}, 6)\}$ ,  $T_2 = \{(p_{21}, 2), (p_{22}, 5), (p_{23}, 7), (p_{24}, 9)\}$  and  $T_3 = \{(p_{31}, 1), (p_{32}, 5), (p_{33}, 8)\}$ . Suppose  $p = p_{12} = p_{22} = p_{32}$ . Then the average delta duration of  $p$  in these 3 trajectories equals  $(2 + 3 + 4)/3 = 3$ . With the average delta duration information on the signature, we propose the *Earth Mover’s Distance on Trajectory (EMDT)* as follow.

**Definition 9 (Earth Mover’s Distance on Trajectory (EMDT))** Given two trajectory sets  $S_a$  and  $S_b$ , we have the corresponding spatial hit histograms notated as  $S_a = \{(p_{a,1}, \delta_{a,1}, w_1), (p_{a,2}, \delta_{a,2}, w_2), \dots, (p_{a,n}, \delta_{a,n}, w_n)\}$  and  $S_b = \{(p_{b,1}, \delta_{b,1}, u_1), (p_{b,2}, \delta_{b,2}, u_2), \dots, (p_{b,m}, \delta_{b,m}, u_m)\}$ , with  $\delta$  representing the average delta duration. We define the distance between  $S_a$  and  $S_b$  as the Earth Mover’s Distance on Trajectory (EMDT) formulated by the following:

$$EMDT(S_a, S_b) = \min \left\{ \sum_{i=1}^n \sum_{j=1}^m Cost_{ij} f_{ij} \right\} \tag{8}$$

with

$$Cost_{ij} = \alpha \times d_{ij} + (1 - \alpha) \times d(\delta_{a,i}, \delta_{b,j}) \tag{9}$$

$$d(\delta_{a,i}, \delta_{b,j}) = \begin{cases} \frac{dist(\delta_{a,i}, \delta_{b,j})}{dist_{max}}, & dist(\delta_{a,i}, \delta_{b,j}) < dist_{max} \\ 1, & otherwise \end{cases} \tag{10}$$

where the  $dist(\delta_{a,i}, \delta_{b,j})$  is defined as the  $l_1$ -norm between  $\delta_{a,i}, \delta_{b,j}$  and  $\alpha$  is an adjustable linear combination parameter, (which can be defined according to the users’ preference, i.e., how to allocate the proportion of the spatial and temporal characteristics in measuring the similarity). Similarly, a large value  $dist_{max}$  is set for normalisation.

The EMDT differs from the EMD in trajectory set similarity measure by the calculation of the cost for transforming one signature to the other, while the computation of the distance value is similar. As we will see in the experiments in Section 6, by capturing the temporal / sequence characteristics of the trajectories, the proposed EMDT is a more robust and effective measure in distinguishing the difference between trajectory sets.

## 5 Trajectory concatenation

In this section, we propose the algorithms to generate the concatenation set of trajectories based on existing data. Firstly, a transition graph is built for candidate *Intermediate Transition Nodes (ITNs)* retrieval, followed by the details of the corresponding algorithm and strategies to obtain ITNs for trajectory concatenation. Intuitively, in order to form the concatenation, it requires to identify the ITNs such that there exist a sizeable set of trajectories between the origin to an ITN, between two consecutive ITNs, and between an ITN to the destination. Moreover, the number of ITNs to form the concatenation could be various, i.e., a different number of transition nodes can form different concatenations of trajectories. A naive approach (intersection-based) to identify the ITNs is to recursively check from the inverted index that whether the nodes are eligible to form the concatenation. For example, given an OD pair  $p_o$  and  $p_d$ , by setting the number of ITNs to be 1, an eligible ITN  $p_1$  must suffice: there exist a sizeable set of trajectories from  $p_o$  to  $p_1$  and trajectories from  $p_1$  to  $p_d$ , which can be determined by the intersection retrieval of the corresponding trajectory position lists from the inverted index.

However, with the increase of the number of transition nodes, the number of intersection retrieval operations grows exponentially since such ITNs retrieval approach runs in a recursive manner. Clearly, the computational cost would be considerably high as for each node traversed in an iteration, we need to invoke the intersection retrieval to determine whether it is eligible to be one of the ITN. Hence, for the purpose of efficiency, we preprocess the intersection retrieval in advance to build a parameter based transition graph over the trajectory data set. In such way, we can identify the ITNs directly from the transition graph rather than invoking the intersection retrieval constantly.

### 5.1 Transition graph

Given an OD pair, in order to generate the corresponding ITNs candidates, we need to identify the reachability from the origin to the destination and how this OD pair is connected according to the existing trajectories. Thus, we build a transition graph based on the trajectory dataset, such that, all the nodes from the graph are made up of nodes from the road network that contain a certain number of trajectories traversing on. And if there exist a sizeable set of trajectories/sub-trajectories from one node  $p_a$  to another  $p_b$ , then there is a directed edge between  $p_a$  and  $p_b$ . Formally, we define the transition graph as follow.

**Definition 10 (Transition graph)** The transition graph, denoted by  $G_t(V_t, E_t, \lambda)$ , is a directed graph derived from a trajectory database  $D$  that suffices:

- $V_t$  is formed by the union of the road network nodes occurring on the trajectories in  $D$ , where each node is traversed by at least  $\lambda$  trajectories;
- If there is an edge  $(u, v) \in E_t$  directed from  $u$  to  $v$ , there must exist at least  $\lambda$  trajectories/sub-trajectories traversing from  $u$  to  $v$ .
- There is no redundant edge. Given an edge  $(u, v)$ , we say  $(u, v)$  is redundant, if there exists a node  $w \in V_t$ , such that  $(u, w) \in E_t$  and  $(v, w) \in E_t$ , and the underlying sets of trajectories (i.e. the ids of trajectories) traversing on the three edges  $(u, v)$ ,  $(u, w)$ ,  $(v, w)$  are completely the same.

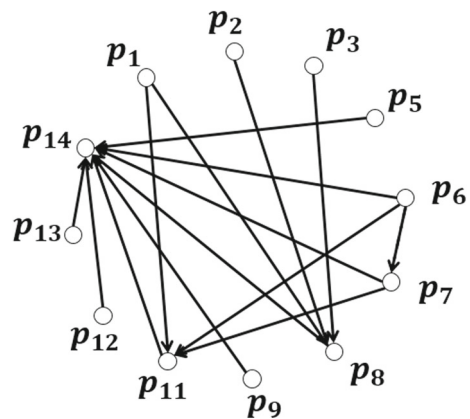
The transition graph is a parameter-based graph, where the variable  $\lambda$  determines the size and the precision of the graph. In particular, a smaller  $\lambda$  indicating a denser graph but more

connectivity relationships between two nodes recorded, while a larger  $\lambda$  depicting a portable graph but only prosperous connectivity relationships reserved (i.e., edges with enough trajectories traversed in between). The  $\lambda$  parameter can be adjustable based on the application demand or according to the density of trajectories on the specific road network. Intuitively, without such a setting, an edge traversed by very few trajectories will also be recorded, while it provides little information on the diversity of traveling behaviours between two nodes. In the definition of the graph, the first condition demonstrates the deriving of the nodes to build the transition graph, and the second condition indicates the connectivity relationships. The last condition, not only guarantees the efficiency in the graph traversal, but also reduces the redundancy for concatenation generation.

Figure 4 shows an example of the transition graph with  $\lambda = 1$ , derived from the trajectories in Figure 1b. The transition graph indicates the reachability from one node to another by trajectory. For instance, there are 2 nodes reachable from nodes  $p_1$ , since there exist two trajectories  $T_1$  and  $T_2$  starting from  $p_1$  and ending at  $p_8$  and  $p_{11}$  respectively. Given two arbitrary nodes  $p_a$  and  $p_b$ , we can check from the transition graph to see whether there are trajectories/sub-trajectories from  $p_a$  to  $p_b$  by checking the out-neighbour list of  $p_a$  to observe whether  $p_b$  is one of  $p_a$ 's out-neighbours. Further, we can find out the intermediate transition nodes by traversing the graph in a depth-first-search manner. For example, to obtain the ITNs for concatenating trajectories from  $p_2$  to  $p_{14}$ , we first get the elements from the out-neighbour list of  $p_2$ , which only contains  $p_8$ . And then we check the neighbour list of  $p_8$ . Since  $p_{14}$  is one of  $p_8$ 's out neighbours,  $p_8$  is an eligible ITN to form the concatenation, which is the concatenation of sub-trajectories from  $T_1$  and  $T_3$ , to represent the trajectories from  $p_2$  to  $p_{14}$ .

With the transition graph, when given an OD pair  $(p_o, p_d)$  and a transition parameter  $k$ , the algorithm to get the candidate ITNs, which works as follow. Line 1: Get the elements in the adjacent list of  $p_o$  from the transition graph. Line 2–5: If  $k = 0$ , check if the neighbours contain  $p_d$ . If yes, add the current transition nodes sequence to the result set  $C_k$ . Line 6–10: Otherwise, for each  $p$  in the neighbour list, the algorithm recursively executes *ITNR* with  $p_o = p$  and  $k = k - 1$ . Line 11: Finally, the algorithm returns all ITNs sequences, each of which contains  $k$  nodes. Take the Figure 1b as example with OD pair  $(p_1, p_{14})$  and  $k = 1$ . *ITNR* first gets the neighbours of  $p_1$  which are  $p_8$  and  $p_{11}$ . Then for each neighbour, *ITNR* checks its out-neighbour list to see whether  $p_{14}$  is included. Since  $p_{14}$  is in the out-neighbour list,  $p_{14}$  is an eligible ITN. Similarly,  $p_8$  is also a ITN. The computational cost

**Figure 4** Example of transition graph



can be analysed in a top-down manner. The execution can be formed into a tree. On the root, it is the point  $p_o$ , and the fanout is the number of its out-neighbours. On level 1, each node is from the out-neighbours of the root and the fanout is the number nodes in the corresponding out-neighbour list. There are in total  $k + 1$  levels, and the execution flow follows a DFS from the root to each node. Thus, in conclusion, the complexity is bounded by  $O(d^{k+1})$ , where  $d$  is the maximum out-degree in the transition graph.

---

**Algorithm 1** ITN Retrieval Algorithm (ITNR).

---

**Input:** An OD pair  $(p_o, p_d)$ , the Transition Graph  $TG$ , and a transition parameter  $k$ .

**Output:** a set of ITNs:  $C_k$ , each item contains  $k$  nodes.

```

1: Get the neighbor list  $L_n$  of  $p_o$  from  $TG$ ;
2: if  $k = 0$  then
3:   for each  $p \in L_n$  do
4:     check if  $p_d \in L_n, p \rightarrow C_k$ 
5:   end for
6: else
7:   for each  $p \in L_n$  do
8:      $ITNR(p, p_d, k - 1)$ 
9:   end for
10: end if
11: return  $C_k$ ;

```

---

Compared with the naive approach, to retrieve the ITNs based on the transition graph, we can not only improve the efficiency (since there is no need to perform intersections between two inverted lists), but also reduce the redundancy of concatenation on trajectories. We analyse how the redundant edge can generate the redundant concatenation set of trajectories to verify the importance of the third condition in the definition of the transition graph. For example, from Figure 1b, we can identify that  $(p_1, p_6)$ ,  $(p_1, p_7)$  are both redundant edges. If we keep these edges, by setting OD pair as  $(p_1, p_{14})$ , and  $k = 1$ , the ITNs returned by  $ITNR$  are  $p_6, p_7, p_8, p_{11}$ . However, after retrieving the concatenated trajectories based on these ITNs, we note that taking the  $p_6$  and  $p_7$  as the intermediate transition node respectively, the concatenated trajectories are the same, which are the sub-trajectory from  $T_2$  and  $T_3$  in both cases. Moreover, setting  $k = 2$ , one of the ITNs will be  $\{p_6, p_7\}$ , such that the concatenated trajectories are the same as the aforementioned ones with  $k = 1$ . In addition, according to the complexity analysis of the ITN retrieval algorithm, the computational cost mainly depends on the average out-degree of each node in the transition graph. Thus, the elimination of redundant edges is both efficient and effective for trajectory concatenation.

Next we introduce the construction of the transition graph consisting of two major steps: the basic graph generation and redundant edge reduction. The basic transition graph is the graph that satisfies the first two conditions in the definition, which can be constructed in a straightforward manner. In particular, for each trajectory  $T = (p_1, p_2, \dots, p_j)$ , we add an edge  $(p_i, p_j)$  with weight 1 if  $i < j$  and there does not exist edge  $(p_i, p_j)$  yet. Otherwise, we increase the weight of the edge by one. Finally, we remove all the edges with weight no larger than  $\lambda$ . This can be finished in  $O(n \cdot l^2)$  time where  $n$  is the number of trajectories, and  $l$  is the maximum number of nodes among all the trajectories. After constructing the basic transition graph, we reduce the redundant edges by a traversal of the current graph and eliminate the redundant edges based on a particular strategy. Before the introduction of that strategy, we first illustrate a lemma that is important to edge reduction.

**Algorithm 2** Redundant Edge Elimination Algorithm (REEA).

**Input:** The basic Transition Graph  $BTG$ , the inverted basic Transition Graph  $IBTG$ , the Inverted Index  $II$ .

**Output:** The Transition Graph  $TG$ .

```

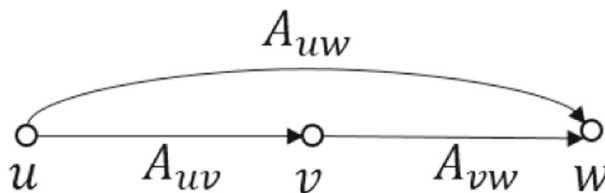
1:  $i \leftarrow 0$ ;
2: while  $i < |V_t|$  ( $V_t \in BTG$ ) do
3:   for each  $p_k \in L_{p_i}$  do
4:     get the neighbor list of  $p_i$  in the  $IBTG$ ,  $IL_{p_i}$ 
5:     if  $\exists p_j \in IL_{p_i}$ , s.t.  $A_{p_j p_i} = A_{p_j p_k} = A_{p_i p_k}$  then
6:       get the  $sh(p_i)$  from  $\mathbb{T}_{p_j p_k}$ 
7:       if  $sh(p_i) = A_{p_j p_k}$  then
8:         delete edge  $(p_j, p_i)$ 
9:       end if
10:    end if
11:  end for
12: end while
13: return  $TG = BTG$ ;

```

**Lemma 1** Let  $\mathbb{T}_{uv}$  be the set of trajectories traversing from  $u$  to  $v$  and  $|\mathbb{T}_{uv}| = A_{uv}$ . If edge  $(u, v)$  is redundant, there must exist a node, such that  $A_{uv} = A_{uw} = A_{vw}$  holds. In addition, if  $A_{uv} = A_{uw} = A_{vw}$  holds and any two sets of trajectories from  $\mathbb{T}_{uv}, \mathbb{T}_{uw}, \mathbb{T}_{vw}$  are the same, then the other set is the same as the two.

*Proof* The first portion of Lemma 1 is obvious. It indicates that if the quantities of trajectories traversing on these three edges are not the same, the exact trajectories/sub-trajectories can not be from the same set of trajectories. As for the second part, given three nodes  $u, v, w$  with edges  $(u, v), (u, w)$  and  $(v, w)$ , shown as Figure 5, we assume that  $\mathbb{T}_{uv} = \mathbb{T}_{vw}$ , then  $\mathbb{T}_{uv} \subset \mathbb{T}_{uw}$  holds. Since  $A_{uv} = A_{uw}$ , we have  $\mathbb{T}_{uv} = \mathbb{T}_{uw}$ .  $\square$

Based on Lemma 1, we introduce the strategy to eliminate the redundant edges, shown in Algorithm 2. Before that, we generate an inverted transition graph by scanning the basic transition graph once, where each neighbour list stores the in-neighbours and the weights of the corresponding node. The Redundant Edge Elimination algorithm works as follow. Line 1 – 2: The algorithm traverses the basic transition graph node by node. For each node  $p_i$ , it first gets the out-neighbour list, denoted by  $L_{p_i}$ . Line 3 – 5: For each node  $p_k$  in the out-neighbour list, it checks the inverted in-neighbour list of  $p_i$  from the inverted basic Transition Graph  $IBTG$ , to see if there exists a node  $p_j$ , such that the numbers of trajectories traversing these three edges are the same, i.e.  $A_{p_j p_i} = A_{p_j p_k} = A_{p_i p_k}$ . Line 6:



**Figure 5** Trajectory quantity relationship



If so, it retrieves the trajectories with respect to two edges  $(p_j, p_k)$ , denoted as  $\mathbb{T}_{p_j p_k}$ , and get the corresponding spatial hit  $sh(p_i)$  of  $p_i$  in  $\mathbb{T}_{p_j p_k}$ . Line 7 – 8: If  $sh(p_i)$  is equal to  $A_{p_j p_k}$ , namely, all the trajectories traversing from  $p_j$  to  $p_k$  go through  $p_i$ . Since  $A_{p_j p_i} = A_{p_i p_k} = A_{p_j p_k}$ , the trajectories traversing this three edges must be the same. Then we confirm that  $(p_j, p_i)$  is a redundant edge, and delete it from the graph. Line 13: Finally, we return the reduced basic transition graph.

### 5.2 Concatenation strategy

The transition graph provides the support to efficiently retrieve the candidate ITNs, i.e., the nodes connecting the origin and the destination by a sizeable set of trajectories. However, it is obvious that not all the candidate ITNs can form the concatenated trajectories that are able to well reveal similar mobility over an OD pair. In this subsection, we introduce the strategies to select the ITNs that the concatenation sets of trajectories are supposed to be similar to the origin-destination ones. We propose several heuristics that can effectively prune the ITNs whose constructed trajectory sets are with high probability dissimilar to the origin-destination set. Hence, it can save the running time of the trajectory concatenation phase, and also guarantee the robustness.

**Pruning with accumulative spatial length** Our first intuition is that if the ITNs can form trajectory set similar to the original-destination set, then if we take the shortest distance from the origin to the ITNS and then to the destinations should be similar to that from the origin to the destination. Hence, we first define the accumulative spatial length as follow, using it as the first pruning parameter.

**Definition 11 (Accumulative spatial length)** Given an OD pair  $(p_o, p_d)$  and a corresponding intermediate transition node sequence  $ITN = \{p_1, p_2, \dots, p_k\}$ , we define the Accumulative Spatial Length (ASL) of ITNs as follow:

$$ASL(p_o, p_d, ITN) = dist(p_o, p_1) + \sum_{i=1}^{k-1} dist(p_i, p_{i+1}) + dist(p_k, p_d) \tag{11}$$

where  $dist(p, p')$  refers to the Euclidean distance between two nodes  $p$  and  $p'$ .

Thus the first criterion is defined as follow.

- Given a sequence of ITN w.r.t. an OD pair  $(p_o, p_d)$ , it is a qualified candidate if it satisfies the below inequality, where  $\tau$  is a tunable parameter ( $\tau > 0$ ).

$$\frac{ASL(p_o, p_d, ITN)}{dist(p_o, p_d)} - 1 < \tau \tag{12}$$

**Pruning with direction** Further, we introduce the direction criterion. The intuition of the direction criterion is that the trajectories traversing from the origin to the destination follow the direction towards from the origin and destination. We choose the ITNs that the direction of the sub-trajectories is close to the ideal direction from the origin to the destination. Thus we define the second criterion as follow.

- Given a sequence of ITN w.r.t. an OD pair  $(p_o, p_d)$ , denoted as  $\{p_1, p_2, \dots, p_k\}$ , the projection points of the intermediate transition nodes regarding the line segment  $\overline{p_o p_d}$  must be on the line,  $\angle p_i p_{i-1} p_d \leq 90^\circ$ , and  $\angle p_i p'_{i-1} p_d \leq 90^\circ$ , where  $1 \leq i \leq k$  (Note

that when  $i = 1$ ,  $p_{i-1}$  refers to the  $p_o$ ) and  $p'_{i-1}$  is the projection point of  $p_{i-1}$  on  $\overline{p_o p_d}$ .

Figure 6 shows an example of how to choose the ITNs based on the direction criterion. Here, the number of transition nodes is set to be 2, and the first node can only locate in the area of the spanning region of angle  $\alpha$ , i.e., on the right side of the vertical line cross  $p_o$ . After the selection of  $p_1$ , the second node can only lie on the sector regarding angle  $\beta$ .

**Pruning with the number of ITNs** Finally, the last pruning criterion refers to the number of ITNs required to form the concatenation. Intuitively, the more ITNs we have, the more chances we have that it goes away from the expected route from the origin to the destination. Hence, we set an integer threshold  $k$  as the maximum ITNs to form the concatenation. As we will see in our experiments, on average when we enlarge the number of ITNs to form the concatenation, the similarity between the concatenation set and the origin-destination set decrease. In addition, according to the complexity analysis of the ITN retrieval algorithm, the computational cost increases exponentially on the number of intermediate transition nodes. Thus, a limitation on the maximum ITNs to form the concatenation is necessary.

From the above, the overall strategy to select the ITNs works as follow. We augment the criteria verification in the ITN Retrieval Algorithm to filter the unnecessary nodes directly. For each node in the neighbour list we visit in an iteration, shown in Algorithm 1, both on Line 3 and 7, we first check whether this node satisfies the direction criterion and compute the current ASL to see whether it exceeds the desired threshold. Only if the node satisfies both requirements, will it be kept for the further processing. We invoke  $k$  times to generate  $k$  sets of ITNs. Eventually, we extract the corresponding concatenation set of trajectories for each ITN sequence. The concatenation strategy can not only guarantee the robustness of the concatenated trajectories but also can accelerate the retrieval of the intermediate transition nodes. This is because if a candidate node is unqualified, the retrieval of the further nodes can be early terminated in the corresponding iteration.

### 6 Experimental study

In this section, we conduct the experimental study to evaluate the efficiency and effectiveness of the proposed framework. Firstly, we present the experimental settings in Section 6.1. Afterwards, we evaluate the effectiveness of our proposed EMDT on trajectory similarity measure in Section 6.2. Next, we further examine the efficiency of our trajectory concatenation component in Sections 6.3 and 6.4. Finally, we demonstrate that the ITNs

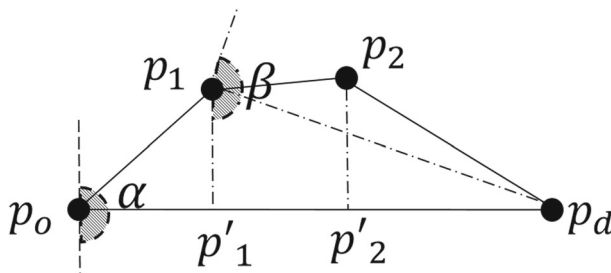


Figure 6 Direction criterion

generated in the trajectory concatenation component indeed derive concatenation set similar to the origin-destination set, and derives better result and is more robust than both the *sweep-and-expand* and the *popularity-based* strategies in Section 6.5.

## 6.1 Experiment setup

The trajectory dataset we use in the experiment is collected from 50K taxis in Beijing for 5 days, which consists of more than 2 million trajectories, each of which is represented by a sequence of GPS records. A map-matching [15] algorithm is employed to transform the raw trajectories into a sequence of time-order road segments (the mapped trajectory set is about 1.6 GB), where the time is estimated by the first GPS point assigned on the corresponding road segment. We consider the road network of Beijing city that composes of 302,364 intersections and 387,588 road segments (60MB).

The constructions of the inverted index and transition graph are executed off-line, and the transition graph is maintained by adjacency lists. We vary the important parameters used in the experiment, which is listed in Table 2, where the default values are shown in bold-face. The parameter  $k$  is the number of ITNs in one concatenation. A larger  $k$  results in smaller similarity between the concatenation set and the origin-destination set of trajectories. Beside, from a statistic on the given dataset, over 85% OD pairs can be concatenated with less than 3 intermediate node, while only 1.5% OD pairs contains trajectories directly traversing in between. Thus, we set that  $k$  is no larger than 3. The  $\lambda$  is a parameter for transition graph generation, which indicates the number of trajectories traversing across each edge. Parameter  $\tau$  is the threshold for the accumulative spatial length of a sequence intermediate transition node. The smaller  $\tau$  makes, the tighter restriction on the candidate ITNs is returned. The parameter  $\alpha$  denotes the percentage of spatial characters for trajectory set similarity measure in EMDT, where  $\alpha = 1$  indicates only the spatial distance is considered for the cost computation, while  $\alpha = 0$  indicates that only average delta duration is taken into account. In the calculation of similarity measure, we follows the algorithm proposed in [16].

All of our algorithms are implemented in Java, and we run all the experiments on a Dell R720 PowerEdge Rack Mount Server with two Xeon E5 – 2690 2.90GHz CPUs, 192GB memory running Ubuntu Server 14.04 LTS operating system.

## 6.2 Rationale of EMD

In this subsection, we design a series of experiments to verify the rationality of the EMD-based trajectory set similarity measure. The experiments contain three perspectives. First, we evaluate multiple distribution similarity measures for spatial hit histogram, which include Murkowski Form Distance (MFD) [22], Jeffrey Divergence Distance (JDD) [18], and EMD. We aim to verify that among those distribution measures, EMD is the most

**Table 2** Parameter settings

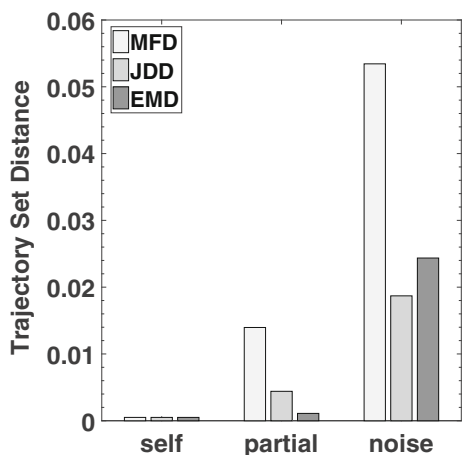
Parameter	Values
$k$	1, 2, <b>3</b>
$\tau$	<b>0.001</b> , 0.01, 0.1
$\alpha$	1, <b>0.95</b> , 0.5, 0.05, 0
$\lambda$	4, 16, <b>64</b> , 128

suitable one for trajectory similarity measure. Secondly, we evaluate the EMD with two different ground distances, Euclidean distance, and road network distance, in order to show the different impact by utilizing different ground distances. Finally, we discover the difference between EMD and EMDT to show that how EMDT can capture the temporal / sequence information.

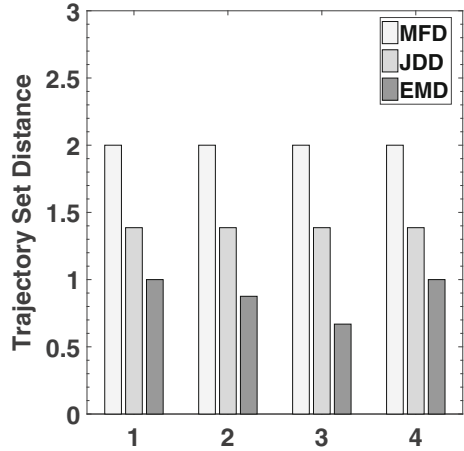
For the comparison of different distribution similarity measures, we randomly select multiple pairs (100 pairs) of OD points, extract the corresponding set of OD trajectories as the sample dataset, and estimate the average distance values. We first compare different distribution similarity measures for the same set of OD trajectories, where each set is divided into two subsets based on different strategies. In particular, we have the self comparison which compares two identical sets, partial comparison that arbitrarily divides a set into two parts, and noise augmentation test which adds noise into a set and compare the noise-added set with the one without noise. In addition, to measure the distance between two different sets of OD trajectories, we divide the sample dataset into 4 subsets and in each subset we arbitrarily pick two OD sets for multiple rounds, calculate the distribution distance, and show the average distance scores. In self-comparison, as the contrastive two subsets are derived from the same set, the distance scores are supposed to be very small. From Figure 7, we can observe that these three distribution similarity measures are robust to noise in self-comparison, as the distance scores are all very small. Also, the self-self distances are all zero, which is superior to the pairwise individual trajectory similarity measure (discussed in Section 2.1). However, when comparing the similarity between two different sets of trajectories, as shown in Figure 8 apart from the EMD, the other two distribution similarity measures cannot distinguish the difference between two different sets of trajectory, as the distance scores of the MFD and JDD are both identical in different sets' comparison, which is in line with our analysis, since the EMD will take the ground distance of the signatures into consideration.

Regarding the different ground distance comparison, we use the Euclidean distance and road network distance respectively in the calculation of the EMD. The data settings in this part are the same as the one for the comparison of different distribution similarity measures. From the Figure 9, we can find out that, the variation of two ground distances is insignificant, which indicates that changing the ground distance from road network distance to Euclidean distance exert little effect on the measurement of EMD. Thus, for the rest of

**Figure 7** Self comparison



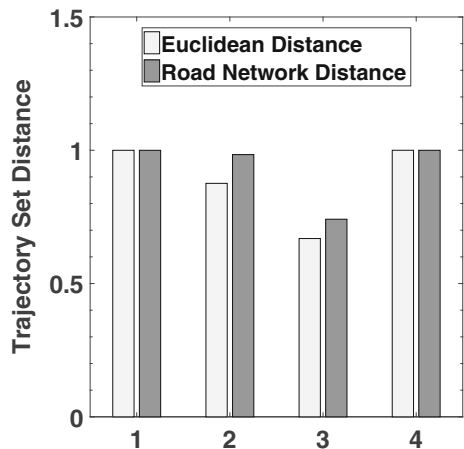
**Figure 8** Different set comparison



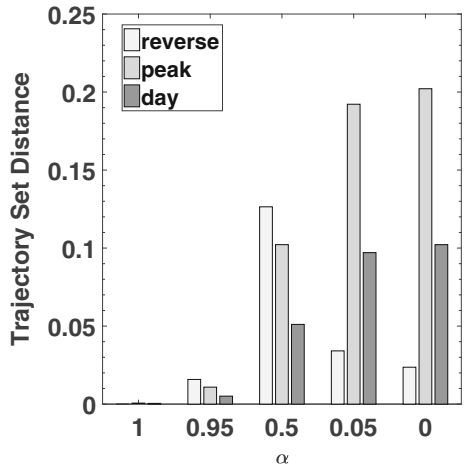
our experiment, we use the Euclidean distance as the ground distance for EMD calculation. Applying the Euclidean distance, we can either reduce the time on ground distance calculation or eliminate the space consumption to store the ground distance matrix for road network distance.

Finally, we test the parameter  $\alpha$  for EMDT calculation. The datasets used in this experiment are from a single set of trajectories (i.e., the trajectories from the same origin to the same destination). We evaluate the average distance scores from 100 OD pairs, where each pair corresponds to a set of trajectories. For each set, we divide the trajectories into two subsets as follow. We first reverse the sequence of trajectories (as discuss in Section 4) and compare it with the origin set (denoted as reverse). Furthermore, we partition the set of trajectories based on temporal information, namely peak time (7:00–9:00am, 5:00–7:00pm) and off-peak time, weekday and weekend. We compare the peak (resp. weekday) set with the off-peak (resp. weekend) set, denoted as peak (resp. day) in Figure 10. And the parameter  $\alpha$  varies from 1 – 0 indicating the spatial character contribution for cost computation.

**Figure 9** Ground distance



**Figure 10** Temporal contribution



The result is shown in Figure 10. As we can see, when  $\alpha = 1$ , which means we only consider the spatial character, the distance between the reverse set and origin set is zero. When the contribution of temporal character increases, the EMDT values also increase. For the comparison between the reverse set and the original set, taking the spatial and temporal / sequence evenly results in largest distinction since considering only one type of characteristics, the difference between reverse set and original set of trajectories is insignificant. In terms of the other two scenario, the distance scores increase with the higher proportion of the temporal characteristics. By intuition, the speed of trajectory during the peak (resp. weekday) time might be lower than the one during the off-peak time (resp. weekend). Thus, we can infer that EMDT can capture the speed information of trajectories sharing the same spatial routes, as the distances between peak set and off-peak set, weekday set, and weekend set, are notable. Consequently, we conclude that 1) EMDT can display the sequence information since the original set is different from the reversed set. 2) EMDT can distinguish the difference between two set of trajectories with different speed, and indicates the temporal feature in trajectory sets.

### 6.3 Transition graph study

To retrieve the intermediate transition nodes efficiently, we build a parameter-based transition graph on top of the trajectory dataset, which consists of two procedures: the basic graph construction and redundant edge reduction. In this set of experiments, we provide a study on the properties of the transition graph with various parameters, as shown in Table 3.

**Table 3** Transaction graph properties

$\lambda$	Number of nodes		Maximum degree		Average degree		Size	
	Basic	Reduced	Basic	Reduced	Basic	Reduced	Basic	Reduced
4	216954	216954	9519	7332	475	395	963.16MB	807.95MB
16	151504	151504	3336	3290	190	188	291.11MB	288.88MB
64	104673	104673	1171	1171	76	76	87.01MB	86.89MB
128	82116	82116	671	671	48	48	46.21MB	46.20MB

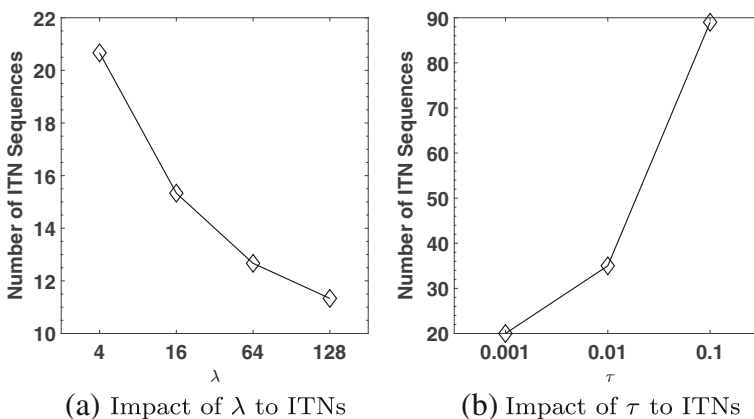
We evaluate four basic properties of the transition graph: the number of nodes (each refers to the road segment on the road network), the maximum degree (the degree indicates the number of out neighbours of a node), the average degree, and the size. We record the properties of different transition graph with parameter  $\lambda$  to be 4, 16, 64, 128 and show the differences between the basic graph and the reduced one. As we can see, the values of those 4 properties decrease gradually with the increase of  $\lambda$ , which is expected, since the parameter  $\lambda$  indicates the minimum number of trajectories traversing on the edges. After the redundant edge elimination, mostly the number of nodes are still the same, but the maximum and average degree decrease by 23% and 17% for  $\lambda = 4$ , 1.4% and 1.1% for  $\lambda = 16$  respectively. As for  $\lambda = 64$  and  $\lambda = 128$ , the differences between the basic graph and reduced graph are insignificant (difference shown by the size). Hence, the redundant edge elimination can significantly reduce the degree and size of the graph with a small parameter.

Besides, we calculate the number of concatenation options (i.e., the number of ITN sequences) retrieved over different parameters. From Figure 11a, we observe that, on average, the number of ITN sequences w.r.t. a specific OD pair can be retrieved from different transition graphs decrease significantly with the increase of parameter  $\lambda$ . Thus, a moderate value of  $\lambda$  is more desirable for concatenation generation considering various factors, including the size of the graph, concatenation options, and the efficiency (discussed in the next subsection). The number of ITN sequences returned based on the transition graph can be affected by the parameter  $\tau$ , which is the threshold parameter for the accumulative spatial length of the ITN sequence. Figure 11b shows the number of ITN sequences increase correspondingly to the increase of  $\tau$ , which is reasonable as the larger  $\tau$  offers a looser restriction on the candidate ITN sequence to be returned.

## 6.4 Efficiency of ITN retrieval

In this subsection, to study the efficiency of the ITN retrieval algorithm based on the transition graph, we compare the computational time of the graph-based algorithm with the basic intersection-based one. In addition, we analyse the running time of graph-based algorithm with different parameters and conditions.

The intersection-based algorithm retrieves the candidate ITN sequences by recursively checking whether there is a sizeable (equals to the parameter  $\lambda$ ) set of trajectories traversing



**Figure 11** Impact of  $\lambda$  and  $\tau$  to ITNs



from one node to another, where the intersection operation scans the two trajectory position lists from the inverted index belonging to the corresponding nodes. In order to compare the computational cost of these two algorithms, we vary the number of ITNs in one concatenation sequence,  $k$  from 1 – 3 to observe the execution time of different algorithms. We randomly select 100 OD pairs and measure the average running time for each algorithm to retrieve the candidate ITN sequence. As shown in Figure 12a, the computational cost of each algorithm increases correspondingly with the increase of  $k$ , which is in line with the complexity analysis. Moreover, apparently, the cost of the graph-based algorithm is dramatically smaller than that of the intersection-based algorithm. The comparison of the computational cost of these two algorithms shows the importance of the employment of the transition graph in concatenation formation, since the running time of the intersection-based algorithm is considerably huge.

In terms of the graph-based algorithm, we first evaluate the running time to retrieve the candidate ITNs without the consideration of our proposed concatenate strategy. For each transition graph with different  $\lambda$ , we randomly pick multiple 50 pairs of nodes to execute the ITN Retrieval algorithm with  $k$  rising from 1 – 3, and show the average computational time. Figure 12b illustrates the running time varying according to different parameter settings. The result suggests that, on average, the computational cost grow significantly with the increase of  $k$ , while the larger  $\lambda$  leads to the smaller running time in each case regarding

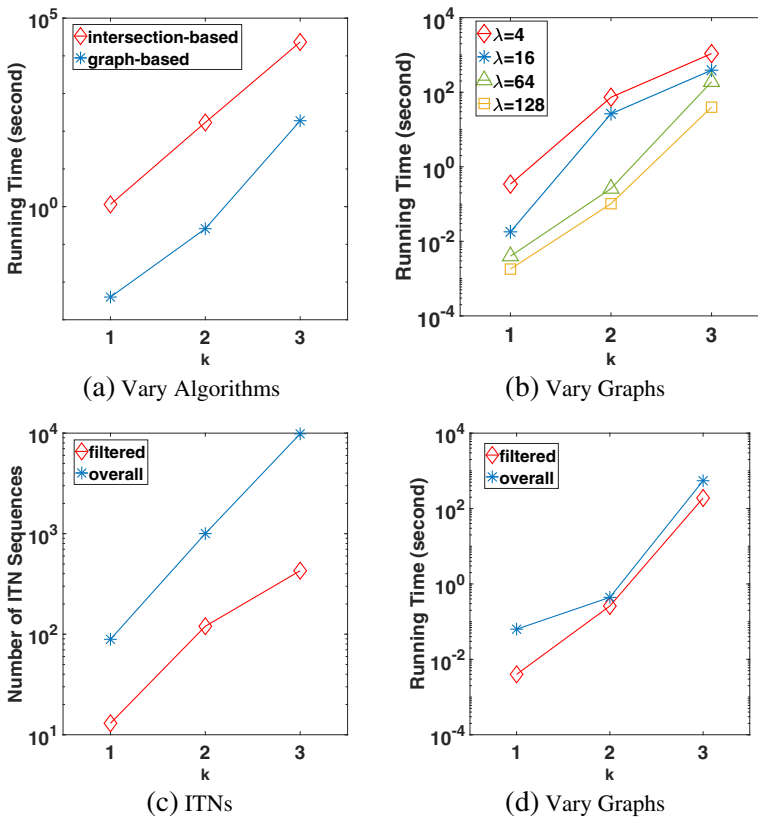


Figure 12 Experimental evaluation on ITN Retrieval

$k$ . The variation regarding to  $k$  is consistent with the complexity analysis in Section 5. The reason for the variation as for  $\lambda$  is that the smaller  $\lambda$  generates a larger transition graph, thus it gives rise to more concatenation options that require more retrieval time.

Furthermore, we also compare the running time in graph-based ITN Retrieval with and without our proposed concatenation strategy. On the one hand, we retrieve all possible ITNs candidates, recording the number of ITN sequences and the corresponding running time (denoted as the overall scenario). On the other hand, we retrieve the ITNs based on our concatenation strategy which filters the unnecessary ITNs based on the accumulative spatial length threshold and direction criterion (denoted as filtered scenario). Figure 12c and d, show both the number of ITN sequences and the running time for the overall scenario are in excess of the ones for the filtered scenario. From this result, we verify the effectiveness of the concatenation strategy that it can filter a large number of the unnecessary ITN candidates, and correspondingly reduce the computational time.

## 6.5 Robustness analysis

In this subsection, we focus on the robustness analysis of the trajectory concatenation based on the proposed trajectory set similarity measure. To begin with, we study the variations of distances between the concatenation set of trajectories and the origin-destination set based on the number of ITNs (i.e.,  $k$ ) and the parameter for accumulative spatial length (i.e.,  $\tau$ ). Further, we compare the robustness between our approach and two other concatenation strategies: *sweep-and-expand* [6] and *popularity-based* [5].

Without special concatenation strategy, we perform a statistic study by retrieving all the possible ITNs w.r.t. to the sample OD pairs, along with the computation of the distance between corresponding concatenation sets and the origin-destination sets. In that way, we can gain an overall understanding of how distance varies by different settings of important parameters. Figure 13a suggests that the distance between the concatenation and origin-destination sets goes up with the increase of  $k$ , which supports the claim that the number of ITNs to form the concatenation set is unnecessary to be large. Similarly, we evaluate the variation of distance according to the accumulative spatial length of the ITN sequence. The

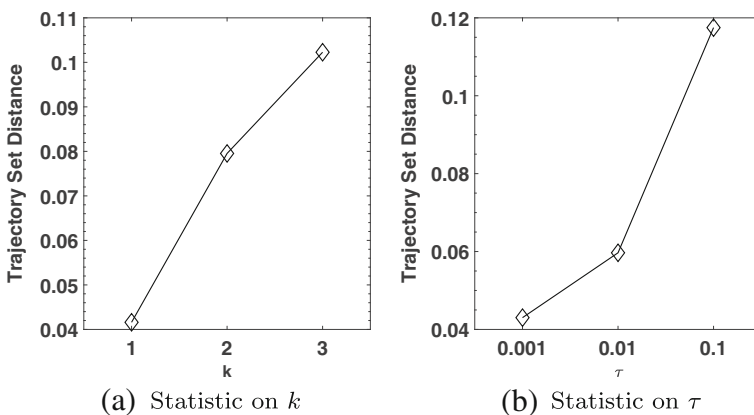


Figure 13 Robustness statistic

results are reported in Figure 13b. For all the candidate ITNs we retrieved w.r.t. a specific OD pair, we calculate the corresponding accumulative spatial lengths. By setting the  $\tau$  to be 0.001, 0.01, and 0.1, we observe that the average distance appears a linear growth trend. This also indicates that the distance between the concatenation set and the origin-destination set increases with the rise of the accumulative spatial length. Next, we estimate the average distance regarding different parameters for the transition graph. Figure 14a shows that with the smaller  $\lambda$ , the similarities between the concatenation sets and the origin-destination set tends to be higher, and grow with the increase of  $k$ .

Finally, we evaluate the robustness of our proposed approach. For comparison, we implement two concatenation strategies. The *sweep-and-expand* strategy concatenates the intermediate transition nodes with a sweeping from the origin to the destination on a grid index, and it does not matter how many transition nodes needed to form the concatenation. We compare the *sweep-and-expand* strategy with our graph-based one and record the average trajectory sets distance, as well as the cases with minimum and maximum distance values. Figure 14b shows that on average our graph-based strategy defeats the *sweep-and-expand* one, and it is more reliable since the *sweep-and-expand* strategy may result in large distance between the concatenation trajectories and the origin-destination ones. The *popularity-based* strategy considers only the popularity of the edges to form concatenation, which selects the ITN sequence that tends to maximise the product of popularity on each edge (where the popularity represented by the number of trajectories traversing on it). In terms of our approach, by setting the parameters to be the default ones, we obtain the ITNs returned by our concatenation algorithm. For each result w.r.t. to a specific OD pair, we pick up the one with the minimum distance, the one with the highest popularity, and calculate the average distance, followed by the computation of the average values on these three distances for all OD pairs. Then we compare these three categories of distance obtained by our approach with the one only considering the popularity. Figure 14c shows the result which suggests that the robustness of our approach is superior to the one only considering the popularity. Moreover, even the one obtained by our algorithm that with the highest popularity reveals a lower robustness. Consequently, concatenating trajectory only based on the popularity probably leads to a result that is ineligible to represent the mobility from the origin to the destination, and our approach can provide both robustness guarantee and sufficient number of concatenated trajectories for further mining.

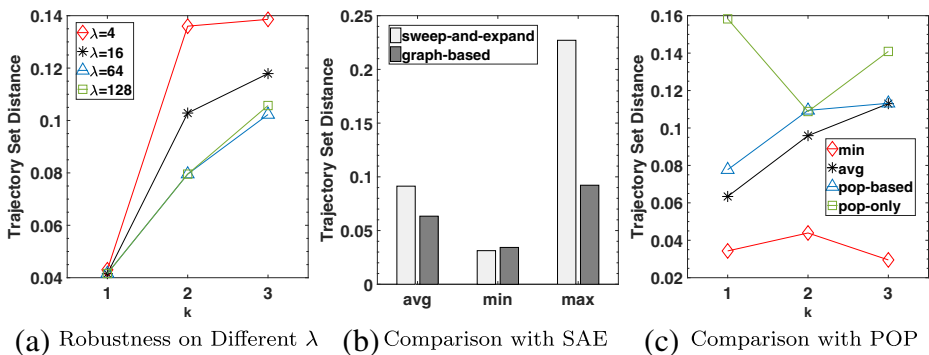


Figure 14 Robustness analysis

## 7 Conclusion

In this paper, we propose an efficient and robust trajectory augmentation approach to address the sparsity issue prevalent in trajectory mining applications that aim to find out the intelligence of the moving behaviours w.r.t. a specific origin and destination pair. The basic idea of the proposed framework is to concatenate the existing sub-trajectories to represent the ones traversing across the OD pair. We first develop a transition graph to support efficient sub-trajectories concatenation. Then, we further propose efficient and effective heuristics to identify good intermediate transition nodes which provide concatenation trajectory set similar to the ones going across the OD pair. We also provide a novel trajectory set similarity measure for the robustness evaluation. The experimental study shows the superiority of our approach. In future work, we will consider the sparsity issue regarding to origin and destination regions.

**Acknowledgments** Sibio Wang was supported by CUHK Direct Grant No. 4055114. He was also supported by the CUHK University Startup Grant No. 4930911 and No. 5501570.


## References

1. Alvarez-Garcia, J.A., Ortega, J.A., Gonzalez-Abril, L., Velasco, F.: Trip destination prediction based on past GPS log using a hidden Markov model. *Expert Syst. Appl.* **37**(12), 8166–8171 (2010)
2. Castro, P.S., Zhang, D., Chen, C., Li, S., Pan, G.: From taxi GPS traces to social and community dynamics: a survey. *ACM Comput. Surv. (CSUR)* **46**(2), 17 (2013)
3. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pp. 491–502. ACM (2005)
4. Chen, Z., Shen, H.T., Zhou, X., Zheng, Y., Xie, X.: Searching trajectories by locations: an efficiency study. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 2010, pp. 255–266. ACM (2010)
5. Chen, Z., Shen, H.T., Zhou, X.: Discovering popular routes from trajectories. In: *2011 IEEE 27th International Conference on Data Engineering (ICDE)*, pp. 900–911. IEEE (2011)
6. Dai, J., Yang, B., Guo, C., Ding, Z.: Personalized route recommendation using big trajectory data. In: *2015 IEEE 31st International Conference on Data Engineering (ICDE)*, pp. 543–554. IEEE (2015)
7. Eddy, S.R.: Hidden markov models. *Curr. Opin. Struct. Biol.* **6**(3), 361–365 (1996)
8. He, D., Ruan, B., Zheng, B., Zhou X.: Origin-destination trajectory diversity analysis: efficient top-k diversified search. In: *2018 19th IEEE International Conference on Mobile Data Management*, pp. 135–144. IEEE, MDM (2018)
9. He, D., Ruan, B., Zheng, B., Zhou, X.: Trajectory set similarity measure: an emd-based approach. In: *Australasian Database Conference*, pp. 28–40. Springer (2018)
10. Kassidas, A., MacGregor, J.F., Taylor, P.A.: Synchronization of batch trajectories using dynamic time warping. *AIChE J.* **44**(4), 864–875 (1998)
11. Kruskal, J.B.: An overview of sequence comparison: time warps, string edits, and macromolecules. *SIAM Rev.* **25**(2), 201–237 (1983)
12. Kullback, S.: *Information Theory and Statistics*. Courier Corporation (1997)
13. Lee, J.G., Han, J., Li, X., Gonzalez, H.: Traiclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proce. VLDB Endow.* **1**(1), 1081–1094 (2008)
14. Lin, B., Su, J.: Shapes based trajectory queries for moving objects. In: *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, pp. 21–30. ACM (2005)
15. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 336–343. ACM (2009)
16. Pele, O., Werman, M.: Fast and robust earth mover’s distances. In: *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467. IEEE (2009)

17. Pelekis, N., Kopanakis, I., Marketos, G., Ntoutsis, I., Andrienko, G., Theodoridis, Y.: Similarity search in trajectory databases. In: 14th International Symposium on Temporal Representation and Reasoning, pp. 129–140. IEEE (2007)
18. Puzicha, J., Hofmann, T., Buhmann, J.M.: Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In: 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings, pp. 267–272. IEEE (1997)
19. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.* **40**(2), 99–121 (2000)
20. Sanderson, A.C., Wong, A.K.: Pattern trajectory analysis of nonstationary multivariate data. *IEEE Trans. Syst. Man Cybern.* **10**(7), 384–392 (1980)
21. Su, H.: Quality-aware trajectory processing using significant locations. University of Queensland (2015)
22. Swain, M.J., Ballard, D.H.: Color indexing. *Int. J. Comput. Vis.* **7**(1), 11–32 (1991)
23. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: 18th International Conference on Data Engineering, 2002. Proceedings, pp. 673–684. IEEE (2002)
24. Wang, H., Su, H., Zheng, K., Sadiq, S., Zhou, X.: An effectiveness study on trajectory similarity measures. In: Proceedings of the Twenty-Fourth Australasian Database Conference-Volume, vol. 137, pp. 13–22. Australian Computer Society Inc. (2013)
25. Wang, Y., Zheng, Y., Xue, Y.: Travel time estimation of a path using sparse trajectories. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 25–34. ACM (2014)
26. Wei, L.Y., Zheng, Y., Peng WC: Constructing popular routes from uncertain trajectories. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 195–203. ACM (2012)
27. Wei, L.Y., Chang, K.P., Peng, W.C.: Discovering pattern-aware routes from trajectories. *Distrib. Parallel Databases* **33**(2), 201–226 (2015)
28. Xue, A.Y., Zhang, R., Zheng, Y., Xie, X., Huang, J., Xu, Z.: Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 254–265. IEEE (2013)
29. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proc. VLDB Endow.* **6**(9), 769–780 (2013)
30. Yi, B.K., Jagadish, H., Faloutsos, C.: Efficient retrieval of similar time sequences under time warping. In: 14th International Conference on Data Engineering, 1998. Proceedings, pp. 201–208. IEEE (1998)

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Affiliations

Dan He<sup>1</sup> · Sibow Wang<sup>2</sup>  · Boyu Ruan<sup>1</sup> · Bolong Zheng<sup>3</sup> · Xiaofang Zhou<sup>1,4,5</sup>

Dan He  
d.he@uq.edu.au

Boyu Ruan  
b.ruan@uq.edu.au

Bolong Zheng  
bolongzheng@hust.edu.cn

Xiaofang Zhou  
zxf@itee.uq.edu.au

<sup>1</sup> University of Queensland, Brisbane, Australia

<sup>2</sup> The Chinese University of Hong Kong, Shatin, Hong Kong

<sup>3</sup> Huazhong University of Science and Technology, Wuhan, China

<sup>4</sup> Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan, China

<sup>5</sup> Guangzhou University, Guangzhou, China