Distributed Human Trajectory Sensing and Partial Similarity Queries

Haotian Wang
haotwang@cs.stonybrook.edu
Computer Science Department
Stony Brook University

Jie Gao
jg1555@cs.rutgers.edu
Computer Science Department
Rutgers University

ABSTRACT

Advances in wireless communication technology have allowed for the collection of large-scale human motion trajectories by recording the appearance of mobile devices within the neighborhood of wireless base stations. Such city-scale datasets pose new challenges on efficient data collection, analysis and similarity based queries. In this paper, we propose new partial similarity measures, categorized as time-sensitive, order-sensitive and order-insensitive ones, and show with real data that these partial similarity measures are more robust than classical measures and more suitable for generating meaningful query results in near-neighbor type of data mining applications. Further, the power of the partial similarity persists even with significant down-sampling. We presented rigorous analysis of the performance of partial similarity measures with subsampling. Our evaluation using real data shows high recall and precision ($\geq$ 90%) with samples only in the order of 1% of the original data size.

1 INTRODUCTION

The ubiquitous availability of mobile devices and wireless coverage has allowed for the collection of a huge volume of human motion trajectories through wireless connections between devices or from devices to base stations. Periodically the mobile agents exchange information with nearby base stations regarding the agent identification (ID), time and location. These records, put together, produces a good representation of the trajectory. Different from previous approaches in which users voluntarily upload their GPS trajectories, the distributed framework of trajectory sensing has allowed for a massive scale of trajectory collection. Examples of trajectories collected in this manner include human mobility traces of over $100,000$ mobile phone users [27] and about $95,000$ users on a large university campus [47]. Such big human mobility data has motivated human trajectory analysis and mining with a wide range of applications including anomaly detection [37, 39], crime investigation [30], city planning [35], traffic analysis and optimization [40].

Trajectory Sensing and Analysis One of the most fundamental questions is to understand properties of natural human trajectories. Prior work on mining and analysis of human trajectories has revealed two important aspects: *individuality* and *commonality*.

On the aspect of individuality, it has been confirmed in multiple datasets [19, 21] that a handful of randomly chosen spatial-temporal data points – the timestamp and location of a user – could uniquely identify the user from a large trajectory database, even with coarse-grained resolution. This suggested that human mobility trajectories, in the space-time domain, are very sparse. It is unlikely that two different users are at the same location at randomly chosen timestamps. The individuality of human trajectories suggests important security-related applications such as crime investigation (locate the individual who appeared in a few locations at certain times). It also reveals interesting interplay with social structures – individuals who frequently visit the same location (even if not in the same order and not at the same time) are strongly correlated with social ties [8, 42]. The individuality of human trajectories also raises potential privacy concerns for releasing the whole trajectory database to the public.

On the other hand, data mining applications have been trying to identify commonalities among the trajectories, such as convoys (groups of users moving together continuously) [31, 32, 36, 41, 44], frequent sequential patterns (frequently occurring subsequences) [38, 46], or popular paths (paths travelled by a significant fraction of users, possibly at different time, among all users that appears on the paths) [20, 33]. These common patterns and features of trajectories suggest important patterns that could be useful for traffic analysis and civil planning.

Understanding both individuality and commonality of trajectories requires efficient similarity search, possibly with different similarity measures, which is the topic of this paper.

Dataset and Motivation. The research problem in this paper is motivated by a trajectory data set from electric motorbikes collected in two different cities over a month. This is a unique dataset – it is large both in the number of users and the time scale, and the motorbikes are individually owned, which makes the dataset to be directly related to individual mobility patterns. In prior work, there have been a lot of studies on mobility datasets of taxis [40, 43, 45], which is suitable for traffic analysis but is less useful for understanding human mobility properties.

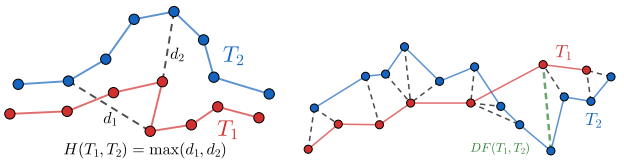
City	Zhengzhou	Wenzhou
Sensing	GPS location	Appearance
Geo-range	$27km \times 33km$	$110km \times 70km$
# of Agents (n)	42,380	633,194
# of Records	19,802,231	50,873,192
Avg # of Records/Agent	467	80

Table 1: Dataset Description

Some basic information from one day's data is shown in Table 1. As could be seen in the table, there are two immediate challenges in data collection and analysis. First, the dataset is huge. There are over 600K agents in Wenzhou with over 50 million records for one day only. For Zhengzhou there are fewer agents (still over 40K) but the data is collected with much higher resolution and the number

of records for one day is beyond 19 million. Collection, storage, and management of the big data is a great challenge. The immediate questions are: 1) whether one could use sparse sampling to reduce data size; yet 2) whether one could still perform meaningful similarity analysis and query for data mining tasks.

Trajectory Similarity and Query. Comparing two trajectories is a heavily studied problem. There has been many commonly used measures to calculate the distance or similarity of two trajectories, including the Hausdorff distance in Figure 1(a) (the maximum of the distance from any vertex on one trajectory to the other trajectory), the Fréchet distance in Figure 1(b) (the min max distance for a traversal along the two trajectories without backtracking) or Dynamic Time Warping (DTW) distance (min sum instead of min max compared to the Fréchet distance)[1]. The Hausdorff distance ignores the order of locations visited by the agent while the Fréchet and DTW distance are sensitive to the orders.



(a) Hausdorff Distance (b) Fréchet Distance

Figure 1: Illustration of the Hausdorff and discrete Fréchet distance of two trajectories.

Our datasets are at a city scale and contains user activities in a long period of time. Using these classical measures to analyze the similarity of two users faces two problems.

First, many of these distance measures are *extreme* measures (of the min-max or max-min type) and capture the worst-case scenario. On a dataset at a city-scale, the distances often come out to be too large to be interesting. It is nearly impossible for any two individuals to be always staying close to one another in a day. In Figure 2, we randomly selected $1,000$ agents in the Zhengzhou dataset and compared the discrete Hausdorff and discrete Fréchet distance (without the timestamp information). About half of the Hausdorff distance and Fréchet distance are more than $20km$, which is the radius of the city. If we want to answer near-neighbor queries (return trajectories similar to a given one), these measures are too sensitive to outlier points and are not very informative.

Second, the computational cost of these distance measures becomes a non-trivial challenge, especially on data at this scale. First, computing the distance of two trajectories could take a quadratic running time (e.g., for Fréchet distance and Dynamic Time Warping distance) on the number of data points of each trajectory. For similarity-based queries, i.e., return the trajectories in the database that are similar to a given one, even a linear running time algorithm (as for the case of Hausdorff distance) becomes infeasible, if we need to scan and compare with the entire database. We need to design super efficient algorithms to answer such queries.

[1]We focus on discrete versions of curve similarity, in which the trajectories are given in the form of a sequence of (possibly time-stamped) points.

Our Results. In this paper, we propose efficient methods for sensing, processing and answering queries of massive city-scale trajectory datasets.

We develop and use new concepts of *partial* trajectory similarity measures. The motivation is two-folds. First, we would like to reduce sensitivity to extreme values and develop measures that generate more differentiating results in similarity queries. Second, by using different amount of information in human trajectories, we have measures for different interesting features in terms of individuality and commonality.

- *Time-Sensitive partial similarity*: Each trajectory is a sequence of time-stamped locations. Two trajectories are α-Time-Sensitive similar if at least an α fraction of samples on the short trajectory are the same ones (or nearby with a specified distance/time threshold) on the other trajectory, including time stamp and location.
- *Order-Sensitive partial similarity:* Each trajectory is an *ordered sequence* of locations visited by the agent. Two trajectories are α-order-sensitive similar, if an α fraction of samples on the shorter trajectory are matched (exactly or with a given distance threshold r) with the samples on the other trajectory in the corresponding order. This could be considered as a partial measure for Fréchet distance.
- *Order-Insensitive partial similarity:* Similar to the previous case, but ignore the order of locations an agent visits. This could be considered as a partial measure for Hausdorff distance.

The partial similarity/distances are more robust and differentiating than the classical distances. It is well known that the trajectories of the same agent in different days are similar. The plot using partial similarities clearly demonstrates this while classical measures fail to catch this pattern. See Figure 2 and Section 3.

The three partial measures, Time-Sensitive, Order-Sensitive, and Order-Insensitive ones, keep a decreasing amount of information (temporal dimension, traversal order, or appearance only, respectively) from the original trajectory data. Therefore, the focus of (partial) similarity in these three measures transitions from individuality to commonality in trajectory data. We empirically evaluated this dimension. By using a random sample from a given query trajectory, we evaluate the minimum number of samples such that there is a unique agent whose trajectory matches the given samples according to each of our defined similarity measures. We find that the temporal property and traversal order are important features to identify an agent with a small number of samples, suggesting these properties to be differentiating and sensitive features for user privacy.

Next, we use random sampling to create sketches for answering partial similarity queries efficiently. We consider a network setting in which the mobile agents communicate with nearby checkpoints/base stations with their appearances and possibly timestamps (exactly the same setting in which the two datasets were collected). Each checkpoint uses a probability p to collect the time-stamped location from the agents nearby. We build hashes using these samples, for the different partial similarity measures. Here the collected records for the same mobile agent can be treated as a sequence, sensitive in the temporal order, or as a set, insensitive to

the temporal order. In particular, for the time-sensitive partial similarity, we keep the samples as a set of tuples (t, c), with timestamp t and location c; for the order-sensitive/order-insensitive partial similarity, we keep the samples as a *sequence*/set of locations visited, ignoring the timestamps.

We show that with a small sample probability p, this reduces the data rate and communication cost from mobile agents to base stations, yet the sketches allow efficient partial similarity queries. Intuitively, two trajectories of high similarity are still likely to share a lot of samples even with significant down-sampling. Answering queries for partially similar trajectories of a given query trajectory becomes a simple table lookup by using a properly chosen bucket size - simply report the trajectories that are hashed to the same bucket. In all three cases, we analyze rigorously the true positive rate (a trajectory that is α-similar is retrieved) of this random hash, and empirically evaluated the precision and recall for different choices of parameters. Specifically, to guarantee a good recall ratio, we just need to maintain very small samples of the original trajectory. For example, for α-Time-Sensitive similarity, with $\Omega(\frac{1}{\delta^2}\log(1/\eta))$ samples, trajectories that are β-similar to the query trajectory, with $\beta \in (\alpha-\delta, \alpha+\delta)$ can be retrieved with probability $1-\eta$. For order-sensitive and order-insensitive cases, we need $\Omega(\frac{1}{\delta}\sqrt{l\log(1/\eta)})$ samples to obtain a good recall ratio, where l is the length of the trajectory in the dataset. In practice, we do not need such many samples.

Last, we implemented our framework and algorithms to evaluate similarity queries for both datasets. Precision and recall are two factors that we mainly focus on. There are three parameters that have affect on the performance, including similarity parameter α, sample probability p and threshold coefficient k, which is related to δ, setting the similarity threshold for retrieved instances. In our evaluation, we find that for trajectories that cover a time scale of a week, the sample probability could be less than $0.01 \sim 0.1$, for both recall and precision to be higher than 90%, with an appropriate threshold coefficient k. This leads to a substantial amount of saving in both communication and data processing costs. The running time can be reduced by 70% approximately for Time-Sensitive similarity and Order-Insensitivity similarity. For Order-Sensitivity similarity, the running time can be reduced by more than 90%.

In the following, we start by reviewing the related work in Section 2. In Section 3, we introduce existing measurements in curve similarity and discuss the reason why they are not suitable for real applications. Then, we proposed our framework and concept of partial similarity in Section 4. At the same time, we talk about how to compare two trajectories and analyze the performance guarantee in this section. The simulations are presented in Section 5 and Section 6 concludes this paper.

2 RELATED WORK

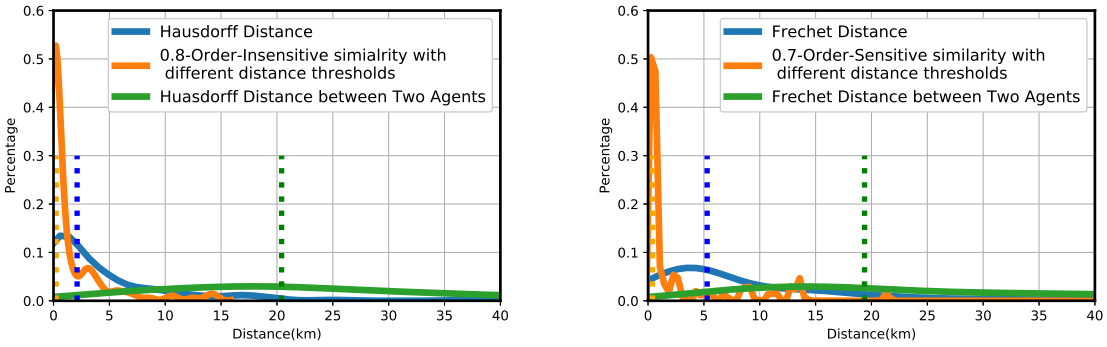
Distances between Two curves. There has been a lot of work on computing distances between two trajectories or curves. For two discrete trajectories of length l_1, l_2 respectively, the Hausdorff distance can be computed in time $O(l\log l)$, with $l = l_1 + l_2$. For the Fréchet distance, Alt and Godau presented an $O(l_1 l_2 \log(l_1 l_2))$ algorithm for two polygonal curves with l_1 and l_2 vertices respectively [5, 26]. Over the decades, studies on the Fréchet distance have developed into a rich field of research, in which many generalizations and variants are studied [4, 14, 17, 24]. However, the quadratic worst-case complexity of Alt and Godau's algorithm had only a log-factor improvement [13]. Bringmann recently presented strong evidence that the Fréchet distance has no strongly subquadratic algorithms, by proving that any such algorithm would yield a breakthrough for the Satisfiability problem (specifically it would break the Strong Exponential Time Hypothesis) [11]. Similar lower bounds exist for Dynamic Time Warping distance [12]. Thus, in a data set of n trajectories, finding the near neighbors of a given query, implemented in the naive manner, will run in $O(nl^2)$ time. This is computational quite heavy for the kind of dataset we consider.

Curves of Special Types. Due to the high computational cost of the Fréchet distance for general curves, researchers have thus focused on curves of special properties, such as backbone curves [7], k-straight curves [6], ϕ-low density curves [24], and c-packed curves [24]. While these notions improved our understanding of characteristics of curves that make Fréchet distance computationally hard, Julian and Karl [10] found that real-world trajectories may not have these properties, for example, in the data set of the ACM SIGSPATIAL GIS Cup 2017, derived from GPS traffic data in San Francisco Bay Area.

Similarity and Hashing. A useful algorithm in practice is to efficiently answer similarity-based queries to a trajectory database. For example, given a query curve q, the k-nearest neighbor query asks for finding the curves that are the top k most similar ones to q; the r-range query asks for all trajectories within distance r from q [3, 18].

A commonly used idea is Locality Sensitive Hashing (LSH) framework [29]. Indyk used the product metrics [28] to achieve an approximation of $O(\log l + \log\log n)$ for the trajectory with nearest neighbor discrete Fréchet distance search with query time of $O(l^{O(1)}\log n)$, where l is the length of the trajectory and n is the number of trajectories. To improve the storage of data structure and query time, a new LSH scheme was proposed by Drimel and Silvestri [25]. Their idea is to snap trajectories to a grid placed with random shift, Or apply a fixed sequence of random perturbations to the vertices of a trajectory before snapping them to nearest grid points, which achieves a constant approximation factor. This is tested on real world data sets in [16]. Similarly, the recent work by Maria et al. [8] proposed to snap trajectories to randomly placed disks. When two trajectories have Hausdorff or Fréchet distance less than $\frac{2R}{a}$, the lower bound of probability that they have the same hash value is $1 - O(\frac{R^2 Dl}{A})$, where R is the radius of the disk, D is the number of disks, A is the area of the region and a is a constant. This probability bound is good when the trajectories are long.

Addressing outliers. There was also work in the literature trying to address the issue of outliers in distance calculation. In *partial Fréchet distance* [14], given a threshold δ, the similarity of two curves is defined as the total length of longest subcurves that are matched with Fréchet distance of at most δ. It can be computed in $O(l^3\log l)$ time under the L_1 and L_∞ norm, but computation under the L_2 norm is still a problem. Alternatively, *Fréchet distance with shortcuts* [23] considers traversals allowing shortcuts [15] or forward jumps [9]. The first one means that the detours are replaced by line segments. They need to decide whether the *shortcut Fréchet*



(a) Density of Hausdorff Distance

(b) Density of Fréchet Distance

Figure 2: Distribution of distances and partial similarities in Zhengzhou. The blue curve and the orange curve describe the distance of two trajectories of two consecutive days of the same agent. The green line show the distance between the trajectories of two randomly chosen agents in one day. For partial similarity measures, we consider two sample points (without timestamps) match if they are within distance r, with r to be the x-axis of the plot. We mark the 50 percentile by the vertical dotted line.

distance is within the threshold δ. In the general case, the shortcuts can start and end at any point on the input curve. They provided a 3-approximation algorithm to solve it. For the vertex-restricted case, the shortcuts have to start and end at input vertices. There is an exact algorithm. Both algorithms run in $O(l^3 \log l)$ time. The forward jump simply skip the detour and and gave an $O(l^3 \log^3(l))$-time deterministic algorithm, but the forward jump is only allowed in one curve. However, all of these algorithms have high computation times which are not suitable to our applications.

Curve Simplification. In managing large trajectory datasets curve simplification is often adopted, where the goal is to construct an approximate curve that is close to the original. The Douglas-Peucker algorithm [22] is probably the most popular algorithm of this type. A recent work uses topological persistence to simplify trajectories, and find significant turns at different resolutions [34].

Compared to these work, our work is closely tied to the setup of trajectory sensing scenario. Our random sampling could be considered as curve simplification as well, with guarantee of preserving partial similarity measures to other trajectories. Also our hashing could be considered as locality sensitive hash for partial curve similarity with a better probability bound than the previous solution [8]. If the length of trajectories are almost the same, our work improve the lower probability bound to $1 - 1/e^{O(l)}$, where l is the length of trajectories.

3 CHALLENGES

Many data mining applications require a similarity measure as a basic routine. We first introduce the classical similarity measurements, Hausdorff and Fréchet distance. We assume a discrete setting in which each trajectory is a sequence of points in the plane.

Definition 3.1. ***(Hausdorff Distance):*** *Consider two trajectories $T_1 = \{u_1, u_2, \cdots, u_{l_1}\}$ and $T_2 = \{v_1, v_2, \cdots, v_{l_2}\}$ of length (number of vertices) l_1 and l_2, respectively. The Hausdorff distance is the maximum of the distance from vertices on one trajectory to the nearest vertex on the other trajectory:*

$$H(T_1, T_2) = \max\{\max_{u \in T_1} \min_{v \in T_2} d(u, v), \max_{v \in T_2} \min_{u \in T_1} d(u, v)\}$$

where $d(u, v)$ is the distance between vertex u and v.

The Hausdorff distance ignores the traversal order of vertices on a trajectory. The Fréchet distance explicitly considers the order of vertices on a trajectory. We first define the concept of a traversal:

Definition 3.2. ***(Traversal):*** *Given two trajectories T_1, T_2, of size l_1, l_2 respectively, a traversal $\tau = \{(i_1, j_1), (i_2, j_2), \cdots, (i_l, j_l)\}$ is a sequence of pairs of indices referring to a pairing of vertices from two trajectories with the properties:*

(1) $i_1 = 1, j_1 = 1, i_l = l_1$ *and* $j_l = l_2$
(2) $\forall (i_k, j_k) \in \tau$: $(i_{k+1} - i_k) \in \{0, 1\}$ *and* $(j_{k+1} - j_k) \in \{0, 1\}$
(3) $\forall (i_k, j_k) \in \tau$: $(i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$

Definition 3.3. ***(Discrete Fréchet distance):*** *Let $\mathcal{T}$ be the set of all traversals between two trajectories T_1 and T_2. The discrete Fréchet distance $DF(T_1, T_2)$ between them is:*

$$DF(T_1, T_2) = \min_{\tau \in \mathcal{T}} \max_{(i_k, j_k) \in \tau} d(u_{i_k}, v_{j_k})$$

Both measurements, taking min-max or max-min values, are sensitive to outliers. In Figure 2, we show the distribution of Hausdorff distance and Fréchet distance in our trajectory datasets. The green lines describe Hausdorff distance and Fréchet distance between the trajectories of two random chosen agents in one day. About half of the pairs have distance more than $20km$, which is more than the radius of the city. The blue lines consider the trajectories of the same agent in two consecutive days. Indeed the trajectories of the same agent in different days are shown to be more similar to each other. By relaxing to partial similarities (shown by the orange curves), the distance thresholds of achieving α-similarity shows much higher level of differentiation of the trajectory pairs.

4 TRAJECTORY SENSING, ANALYSIS AND QUERY

We consider the setting in which n mobile agents move within a geographical region with m wireless checkpoints $C = \{c_1, c_2, \cdots, c_m\}$. The checkpoints, being roadside units, WiFi access points or cellular base stations, are assumed to be connected to each other by existing infrastructure. The mobile agents periodically communicate with the checkpoints with their IDs (tagged with timestamps) to nearby checkpoints. The data records may also be *appearances* only (with or without timestamps) or may carry detailed GPS locations, if the mobile agents have GPS data.

The trajectory set of these mobile agents is represented as $T = \{T_1, T_2, \cdots, T_n\}$. Each trajectory is a sequence composed of pairs of time stamp and location (either the location of the checkpoint visited by the mobile agents or the GPS location of the mobile agent). The length of trajectory T_i can be represented as l_i.

In the query process, given a query trajectory T_q, which might be a trajectory of one agent in the data set or a sequence of checkpoints and possibly appearance timestamps, we wish to find trajectories in the dataset T that are similar to T_q. There are many different ways to define the similarity between two trajectories. We will now define rigorously our similarity measure.

4.1 Partial Similarity

The motivation for partial similarity is to avoid the measure being overly sensitive to outliers.

First, we talk about Time-Sensitive similarity. A trajectory records the appearance of a mobile agent in the neighborhood of checkpoints and its appearance time. It is a set of tuples (t, c), which means that the agent visited the checkpoint c at time slot t. Time-Sensitive similarity measures the degree of two agents appearing near the same checkpoint at the same time slot. We denote the number of common tuples in two trajectories by $CC(T_q, T_i)$.

Definition 4.1. ***Time-Sensitive Similarity:*** *Given a query trajectory T_q and a similarity threshold α, $0 \leq \alpha \leq 1$, the trajectory T_i is α-Time-Sensitive similar to the query trajectory T_q if $\frac{CC(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$.*

Next, we relax the time dimension to consider only the order that an agent visits different checkpoints, termed the Order-Sensitive similarity. A trajectory only records the appearances of a mobile agent in the neighborhood of checkpoints as a sequence of checkpoints visited. The Order-Sensitive similarity is to capture the shared sequential pattern in two trajectories by finding the longest common subsequence between two sequences. It is similar to the definition of Fréchet distance. We denote the length of the longest common subsequence as $LCS(T_q, T_i)$.

Definition 4.2. ***Order-Sensitive Similarity:*** *Given a query trajectory T_q and a similarity threshold α, $0 \leq \alpha \leq 1$, the trajectory T_i is α-Order-Sensitive similar to the query trajectory T_q if $\frac{LCS(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$.*

Last, we only record the checkpoints visited by agents, ignoring timestamp or traversal order. A trajectory is a set of checkpoints visited. The Order-Insensitive similarity is to consider the common checkpoints visited by two agents, denoted as $CN(T_q, T_i)$. Now we provide the definition of Order-Insensitive similarity:

Definition 4.3. ***Order-Insensitive Similarity:*** *Given a query trajectory T_q and a similarity threshold α, $0 \leq \alpha \leq 1$, the trajectory T_i is α-Order-Insensitive similar to the query trajectory T_q if $\frac{CN(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$.*

These partial similarities are more robust to noises or perturbations than the discrete Hausdorff distance and Fréchet distance. In the sensing process, it is common to miss some points or add some noise points in the trajectory, which might increase Hausdorff distance and Fréchet distance, but may not influence the partial similarity much.

4.2 Random Collection

The current system of trajectory sensing collects location records on the order of every $10 - 15$ seconds. This produces a huge volume of data, resulting in a heavy burden for communication, storage, and analysis. A natural question we ask is, can we reduce data sampling rate/size and develop an efficient query scheme with performance bounds for data mining application?

First, we propose our sampling sketches to collect the trajectories from the checkpoints. The sample probability p is set to reduce the data size. In each collection time slot, each checkpoint decides to collect the information from the agents with probability p and do nothing otherwise. Then, for different settings, we store the trajectories and compute the similarity in different ways:

(1) Time-Sensitive Similarity: For any trajectory T_i as a set of tuples, each tuple can be selected with probability p according to the checkpoint and the time slot. If a tuple is selected, it is stored in subtrajectory H_i, with the same data structure with T_i. Then we can set a hash table for each tuple, i.e. $< t, c >$ as the keys of hash tables. To compute Time-Sensitive similarity between two trajectories, we just need to go through the shorter trajectory and check how many tuples are collide with the other trajectory in the hash tables. It is a linear algorithm.

(2) Order-Sensitive Similarity: The trajectory T_i is stored as a sequence or a vector in the form of $< c_1^i, c_2^i, \cdots, c_l^i >$. To randomly collect this trajectory, we also store the subtrajectory as a sequence or a vector. As computing Order-Insensitive similarity between two trajectories, we have to use dynamic programming to solve the longest common subsequence between them, which is a quadratic-time algorithm.

(3) Order-Insensitive Similarity: The trajectory is stored as a set, containing all the checkpoints visited, as $\{c_1^i, c_2^i, \cdots\}$. With a sample probability p, we collect the subtrajectory as a set. Using these sets, we maintain a hash table, with the checkpoints as the keys. For each checkpoint, it stores all the agents who visit it. To compute the Order-Insensitive similarity, we take all the trajectories of agents and check whether they visit the checkpoints which are along the query trajectory T_q. We can get the satisfied trajectories quickly. It is a linear algorithm, depended on the number of samples in the dataset and the number of checkpoints.

The process of our framework is shown in Figure 3. The nodes are checkpoints in the city. The red nodes represent when the agent arrives, it records the appearance at that time. The sample probability for the checkpoints is 40%. Through the random collection, according to the query case, we store the trajectories in different

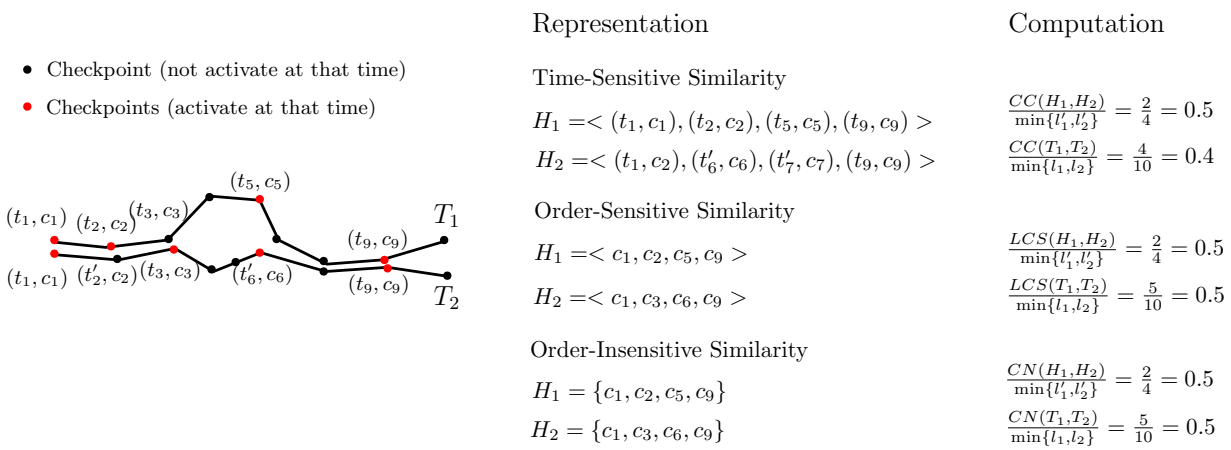


Figure 3: Process of sampling and computation: There are two agents who visit 10 checkpoints. When the agent arrive at a checkpoint, with probability p, the checkpoint collects the information from all nearby agents. The sample probability p is set as 0.4. Then, for different query cases, we represent the sampled trajectories in different ways. According to the definitions, we compute the similarities between the sampled trajectories and the original trajectories.

ways, as the representations in Figure 3. We can see the lengths of trajectories are compressed. Then, we compute the similarity using two sampled trajectories H_1 and H_2. The result is close to the ground truth.

4.3 Performance Analysis

In this section, we analyze the relationship between the similarity of the sampled sub-trajectories and that of the original trajectories. In the following analysis, we mainly use Chernoff Inequality to bound the variance and probability error. Thus, we first introduce Chernoff Inequality.

Theorem 4.1. ***Chernoff Bound**: Let $X_1, X_2, \ldots, X_t$ be i.i.d. random variables with range $[0, c]$ and expectation μ. Then, if $X = \frac{1}{t}\sum_i X_i$ and $0 < \delta < 1$,*

$$P[|X - \mu| \geq \delta\mu] \leq 2\exp(\frac{-\mu t\delta^2}{3c})$$

In this part, we mainly focus on Time-Sensitive similarity. Given a query trajectory T_q and another trajectory T_i, both of them are sequences of tuples, including the appearance time and checkpoints visited. The parts of trajectories collected by the checkpoints are denoted by H_q and H_i respectively. We use the methods in Section 4.2 to compute the Time-Sensitive similarity between H_q and H_i.

Theorem 4.2. *Given a query trajectory T_q and another trajectory T_i with time-sensitive similarity at least α, with sampling probability p, the sampled trajectories are H_q and H_i with length l'_q and l'_i respectively. If $pl \geq \frac{3\log(2/\eta)}{\alpha\delta^2} = \Omega(\frac{1}{\delta^2}\log(1/\eta))$, where l is the shorter length of T_q and T_i, by Chernoff inequality, we have*

$$Prob\{|\frac{CC(H_q, H_i)}{\min\{l'_q, l'_i|\}} - \alpha| \geq \delta\alpha\} \leq 2\exp(-\frac{pl\alpha\delta^2}{3}) \leq \eta$$

Proof. Suppose the query trajectory is the shorter one and its length is l, the expected number of samples in H_q is pl. Since $\frac{CC(T_q,T_i)}{\min\{l_q,l_i\}} \geq \alpha$, the number of tuples shared in trajectories T_q and T_i are at least αl. Each of these tuples has a probability p to be selected into the sampling sketch – the checkpoint with probability p to take samples of nearby agents and if one tuple is taken by a nearby checkpoint, the one on the other trajectory must be collected within the same time slot as well. Thus, the expected number for $CC(H_q, H_i)$ is pl. Let $X_j = 1$ if the j-th pair of H_q are matched with the pair in H_i, and 0 otherwise. Let $X = \sum_{j=1}^{pl} X_i$. We have

$$\begin{aligned} & Pr\{|\frac{|(V_q, V_i)|}{\min\{|V_q|, |V_i|\}} - \alpha p| \geq \delta\alpha p\} \\ = \; & Pr\{|\frac{X}{pl} - \alpha p| \geq \delta\alpha p\} \\ \leq \; & 2\exp(\frac{-\mu p^2 l\delta^2}{3}) \\ \leq \; & \eta \end{aligned}$$

By keeping subtrajectories of length $\Omega(\frac{\log(1/\eta)}{\delta^2})$, we can use $\frac{|CC(H_q,H_i)|}{\min\{l'_q,l'_i\}}$ to estimate $\frac{|CC(T_q,T_i)|}{\min\{l_q,l_i\}}$ with probability at least $1 - \eta$, where δ is the variance bound and η is the probability bound. □

We have similar theorems for Order-Sensitive similarity and Order-Insensitive similarity.

Theorem 4.3. *Given a query trajectory T_q and another trajectory T_i with order-sensitive similarity at least α, with sampling probability p, the sampled trajectories are H_q and H_i with length l'_q and l'_i respectively. If $pl \geq \sqrt{\frac{3l\log 2/\eta}{\alpha\delta^2}} = \Omega(\frac{1}{\delta}\sqrt{l\log(1/\eta)})$, where l is the shorter length of T_q and T_i, by Chernoff inequality,*

$$Prob\{|\frac{LCS(H_q, H_i)}{\min\{l'_q, l'_i\}} - \alpha p| \geq \delta\alpha p\} \leq 2\exp(-\frac{p^2 l\alpha\delta^2}{3}) \leq \eta$$

Theorem 4.4. *Given a query trajectory T_q and another trajectory T_i with order-insensitive similarity at least α, with sampling probability p, the sampled trajectories are H_q and H_i with length l'_q and*

l'_i *respectively. If* $pl \geq \sqrt{\frac{3l\log 2/\eta}{\alpha\delta^2}} = \Omega(\frac{1}{\delta}\sqrt{l\log(1/\eta)})$*, where* l *is the shorter length of* T_q *and* T_i*, by Chernoff inequality,*

$$Prob\{|\frac{LCS(H_q, H_i)}{\min\{l'_q, l'_i\}} - \alpha p| \geq \delta\alpha p\} \leq 2\exp(-\frac{p^2 l\alpha\delta^2}{3}) \leq \eta$$

For Order-Sensitive similarity and Order-Insensitive similarity, the only difference with Time-Sensitive similarity is that, for matched samples in both trajectories, they may be not within the same time slot. Thus, these matched samples are collected in the subtrajectories H_q and H_i with probability p^2. That is the reason that we need more random samples for the same error bound. In our simulations, the number of samples needed is not increased by too much. One reason is that a trajectory may repeatedly visit the same checkpoint and the matched samples in the subtrajectories are not necessarily coming from the samples at the same time slot.

5 EVALUATION BY SIMULATION

In this section, we present experimental evaluation of our distributed trajectory collection framework. Section 5.1 is about the setup of our experiments including the hardware, dataset and baseline algorithms used as references. In Section 5.2, we analyze the performance of the sampling based sketches, in particular, we investigate how the sampling probability p and the threshold k affect the performance (recall and precision) in similarity based queries. In Section 5.3, the running time of our framework is compared with the framework without sampling. In Section 5.4, we include some advice for practitioners, on how to choose the parameters for the best performance and running time. In Section 5.5, we use the sketches and similarity measures to analyze the *individuality* property of human trajectories. We use Time-Sensitive similarity, Order-Sensitive similarity and Order-Insensitive similarity as the agents' signatures and test how many samples are enough to identify a special agent.

The main take-away messages from our experiments are:

- For a given α, by choosing a proper parameter of sampling probability p around $1\% \sim 10\%$, we could achieve high recall and precision ($> 90\%$) by adjusting the threshold coefficient k (Section 5.2).
- The sampling probability could be taken as a very small number ($\sim 1\%$) when the trajectories are long while still achieving comparable performance. This means significant savings in running time for computing partial similarity. For Time-Sensitive similarity and Order-Insensitive similarity, the running time can be reduced by more than 70%. For Order-Sensitive similarity, the running time can be reduced by 99% for long trajectories. (Section 5.3).
- We randomly select several samples from the trajectory as its signature. Then we check whether these samples are enough to identify a unique agent. We found that for Time-Sensitive similarity and Order-Sensitive similarity, only a constant number of samples are enough to identify the agent. For Order-Insensitive similarity, much more samples (proportional to the length of trajectories) are needed (Section 5.5).

Dataset	Period	# of Agents	Length		
			min	median	max
Zhengzhou	Day	36, 951	20	423	5, 907
Zhengzhou	Week	2, 085	10, 000	11, 477	56, 592
Wenzhou	Day	274, 307	50	110	6, 175
Wenzhou	Week	241	10, 026	15, 499	84, 225

Table 2: characteristics of the datasets used in our experiments

5.1 Experimental Setup.

Hardware. We implemented our algorithm in C++ with OpenMP, using GCC 7.4.0. We ran the experiments on a Ubuntu 18.04.03 machine equipped with Intel(R) Core(TM) i7-8559U @2.70GHz (4C8T), and 32GB of RAM.

Datasets. The trajectories are collected from electric motorbikes in two cities, Zhengzhou and Wenzhou, China, shown in Table 2. The trajectories includes a list of points with vehicle ID, system time, longitude and latitude. For each city, we have more than one-month of data. In the pre-processing step, we remove agents whose trajectories in one day are too sparse (fewer than 20 samples in Zhengzhou and fewer than 50 samples in Wenzhou). Given that each agent may not travel for a long time in one day, we concatenate the trajectories of 7 days to form a trajectory of one-week long for each agent. Then we only keep trajectories that contain more than 10, 000 samples.

The dataset from Zhengzhou is *appearance*-based. The location in a vertex on the trajectory is taken as the location of the checkpoint that has recorded such appearance. There are 5, 228 checkpoints in Zhengzhou. For Wenzhou, the dataset is *GPS location*-based. Thus, it is very unlikely that two agents report exactly the same location. We generate 900 checkpoints in Wenzhou and each checkpoint has a surveillance radius of $0.5km$. This way we transform the GPS-based trajectory data to appearance-based.

Baseline Algorithm. For Time-Sensitive similarity, Order-Sensitive similarity, and Order-Insensitive similarity, we have implemented the classical algorithms to obtain the ground truth similarity. We enumerate all pairs of trajectories in the dataset and calculate the similarity value α For the Time-Sensitive similarity, given that all the checkpoints are stored in a temporal order, we can compute the similarity in linear time. For Order-Sensitive similarity, we compute the longest subsequence between two trajectories. For Order-Insensitive similarity, we just count the number of common checkpoints in these two trajectories.

Evaluation. We implemented the proposed algorithms based on C++ and our source code is publicly shared on Github [1]. We provide a tool-chain of our framework, including data collection, similarity computation and performance analysis. Here are the main steps in our tool-chain.

(1) Process Dataset: The data points are collected with the agents' ID, system time, monitoring time, latitude and longitude. These entries of data are not necessarily ordered. Thus, we construct the trajectory for each agent in a temporal order and remove outliers (with unrealistic speed or at impossible locations).

Index	City	Measure	α	p	k
Fig. 4(a)	Zhengzhou	TS	0.3	0.05%~5%	1
Fig. 4(b)	Zhengzhou	OS	0.8	0.05%~5%	1
Fig. 4(c)	Zhengzhou	OI	0.8	0.05%~30%	1
Fig. 4(d)	Zhengzhou	TS	0.3	3%	0.5~1.4
Fig. 4(e)	Zhengzhou	OS	0.8	1%	0.5~1.2
Fig. 4(f)	Zhengzhou	OI	0.8	1%	0.4~1.2
Fig. 4(g)	Wenzhou	TS	0.5~0.8	0.5%~20%	1
Fig. 4(h)	Wenzhou	OS	0.5~0.8	0.5%~20%	1
Fig. 4(i)	Wenzhou	OI	0.6~0.9	0.5%~20%	1

Table 3: Parameters used in Figure 4.

(2) Parameter Setting: There are two datasets, Zhengzhou and Wenzhou, that can be selected. Then, we need to select the measurement method to compare two trajectories, including Time-Sensitive similarity (TS), Order-Sensitive similarity (OS) and Order-Insensitive similarity (OI). According to the corresponding measurement, the sample probability p should be given.

(3) Comparison Algorithm: We output the similarity measures among all agent pairs in the dataset as the ground truth. We calculate the similarity between the sampled trajectories, and then compare them with the ground truth. Now, the threshold coefficient k can be set and we can retrieve the satisfied instances and analyze the performance (recall and precision).

5.2 Precision and Recall

In this part, we analyze how the parameters (sample probability p, threshold coefficient k, and similarity parameter α) affect the performance of similarity based queries. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total number of relevant instances that are retrieved. The parameters of these experiments are shown in Table 3. We only execute the algorithms on the dataset of one-week long trajectories, as the saving is more significant on long trajectories. For different cases, we set an appropriate value for α such that we retrieve less than 5% of all trajectories.

Sample probability. First, Figure 4(a), 4(b) and 4(c) show the effect of sample probability p on precision and recall. These simulations are all based on Zhengzhou. In general, recall goes up when sample probability p goes up. As the number of samples increases, by Chernoff Inequality, the variance bound δ and the probability bound η can be reduced. In these cases, the variance bound δ does not change. Thus, the probability bound η drops and recall is higher. On the other hand, precision varies in different ways, because precision mainly depends on the dataset and the parameter setting. Take Figure 4(c) as an example: since we may under estimate the similarity parameter between two sampled trajectories, some trajectories with a low similarity (about 0.7) may also be retrieved. The retrieved instances contain all true positive instances but also false positive instances, leading to low precision.

For Time-Sensitive similarity, when the sample probability is about 0.5%, recall is higher than 95%. For Order-Sensitive similarity, the sample probability is close to 1% for the same recall. For Order-Insensitive similarity, when the sample probability is about 10%, recall is close to 1. According to our analysis in Section 4.3, the number of samples needed for Time-Sensitive similarity is the fewest compared with the other two similarity measures. The results in experiments confirm this as well. To achieve a good recall performance, we just need fewer than 1% of samples for Time-Sensitive similarity and Order-Sensitive similarity, and fewer than 10% of samples for Order-Insensitive similarity. In either case the data size can be reduced significantly.

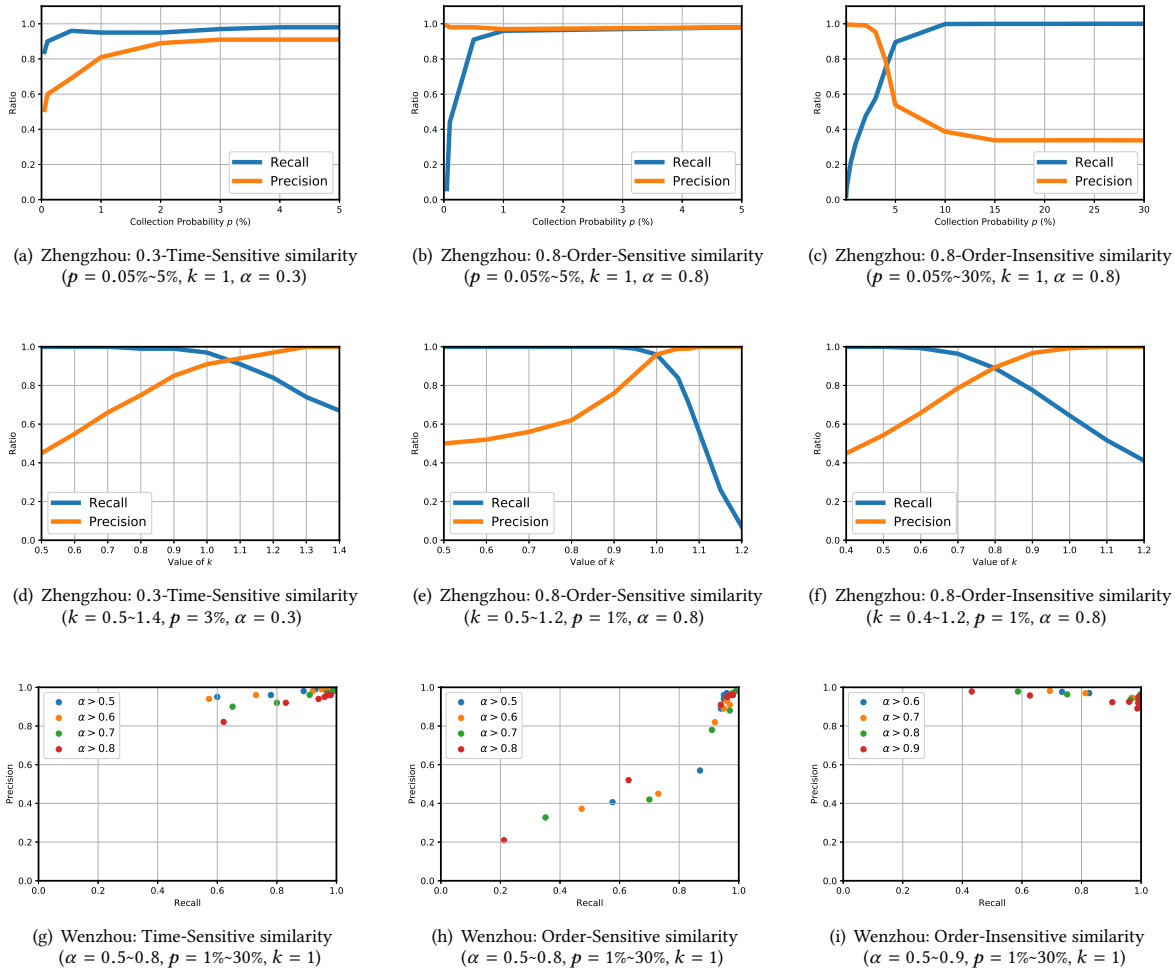
Threshold Coefficient. According to the analysis in Section 4.3, the parameter δ is to set the variance bound for the retrieved instances. If we want to retrieve all trajectories with high similarity with the query trajectory, we set a similarity threshold as the multiplication of the threshold coefficient k and the expected similarity parameter. Trajectories with similarity to the given query higher than the threshold are retrieved.

The effect of this threshold coefficient k on the performance is shown in Figure 4(d), 4(e) and 4(f). In general, with k increasing, recall is decreased and precision is increased. A higher threshold results in fewer retrieved instances. Thus, when k is small, the retrieved instances contain most of the true positive instances as well as some false positive instances. It makes high recall and low precision. When the value of k is large, most of the retrieved instances are true positive instances, but the number of retrieved instances is fewer than the number of true positive instances. Thus, it leads to low recall and high precision.

The threshold coefficient can be used to balance the trade-off between recall and precision. The curves for recall and precision intersect when k is around $0.8 \sim 1.2$. At this point, both recall and precision are higher than 90%. The optimal point for k is depended on the dataset and the parameter setting.

Similarity parameter. For different similarity parameters α, the relationship of precision and recall is shown in Figure 4(g), 4(h) and 4(i). Each color represents a similarity parameter. Nodes of the same color correspond to different sample probabilities (from 1% to 30%). Both precision and recall range from 0 to 1, with 1 being the best. Thus, the closer a node is to the top right corner, the better the performance. In general, most nodes are close to the top right corner, with high recall and high precision. The other nodes not in the top right corner have very small collection probabilities and relatively poor result.

Comparison with LSH. We compared with Locality Sensitive Hashing (LSH) [25] to find the (c, r)-near neighbor trajectories. The experiment is carried out on Shenzhen taxi dataset as it is publicly available [2]. Each trace is for one taxi for one day sampled at a rate of 30 seconds. We split each trace into trajectories, such that each trajectory is a continuous occupied trip for 1 hours. We apply their algorithm to build hash function families and set the pair of trajectories whose Fréchet distance is within $1km$ as the ground truth. We compare the performance between their algorithm and our solution. As shown in Figure 5, we can see that the performance of LSH is slightly better than our solution. However, LSH requires to store all the appearance of the agents, which might need much larger storage capacity and is not suitable for applications.



(a) Zhengzhou: 0.3-Time-Sensitive similarity (p = 0.05%~5%, k = 1, α = 0.3)

(b) Zhengzhou: 0.8-Order-Sensitive similarity (p = 0.05%~5%, k = 1, α = 0.8)

(c) Zhengzhou: 0.8-Order-Insensitive similarity (p = 0.05%~30%, k = 1, α = 0.8)

(d) Zhengzhou: 0.3-Time-Sensitive similarity (k = 0.5~1.4, p = 3%, α = 0.3)

(e) Zhengzhou: 0.8-Order-Sensitive similarity (k = 0.5~1.2, p = 1%, α = 0.8)

(f) Zhengzhou: 0.8-Order-Insensitive similarity (k = 0.4~1.2, p = 1%, α = 0.8)

(g) Wenzhou: Time-Sensitive similarity (α = 0.5~0.8, p = 1%~30%, k = 1)

(h) Wenzhou: Order-Sensitive similarity (α = 0.5~0.8, p = 1%~30%, k = 1)

(i) Wenzhou: Order-Insensitive similarity (α = 0.5~0.9, p = 1%~30%, k = 1)

Figure 4: Performance in terms of precision and recall of our framework on Zhengzhou and Wenzhou. the effect of sample probability p, threshold coefficient k and similarity parameter α on the performance.

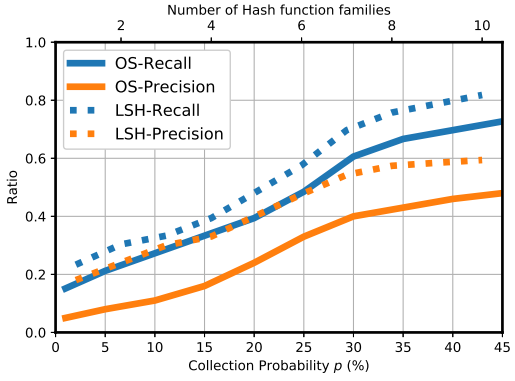


Figure 5: Shenzhen: Order-Sensitive similarity compared with result by LSH

5.3 Time Efficiency

Now, we consider the running time of our algorithms for different query cases, shown in Table 4. This process contains the sampling process (collecting samples from trajectories) and computing the similarity between all pairs of agents. Thus, the running time is affected by the sample probability p, which determines the data size. We ran simulations on two datasets, Zhengzhou and Wenzhou, with three similarity measures. The sample probability p varies from 1% to 30%.

For Time-Sensitive similarity and Order-Insensitive similarity, the running time grows linearly with the sample probability p. Generally, we set the sample probability p less than 10%, achieving recall more than 95%. It means that the running time could be reduced by about 70% approximately, which is a significant improvement. Another interesting observation is that the running time between Zhengzhou and Wenzhou is different in Time-Sensitive similarity and Order-Insensitive similarity. Although the number of agents in Zhengzhou is greater than that in Wenzhou, the average length

Similarity	Dataset	sample probability p							
		1%	5%	10%	15%	20%	25%	30%	100% (Ground Truth)
Time-Sensitive	Zhengzhou	$6ms$	$17ms$	$24ms$	$37ms$	$47ms$	$61ms$	$67ms$	$237ms$
	Wenzhou	$17ms$	$53ms$	$108ms$	$164ms$	$216ms$	$254ms$	$321ms$	$1,142ms$
Order-Sensitive	Zhengzhou	$4s$	$147s$	$251s$	$517s$	$842s$	$1,584s$	$3,125s$	$35,713s$
	Wenzhou	$0.4s$	$21s$	$42s$	$90s$	$180s$	$376s$	$735s$	$6,179s$
Order-Insensitive	Zhengzhou	$10ms$	$27ms$	$43ms$	$61ms$	$74ms$	$87ms$	$100ms$	$361ms$
	Wenzhou	$8ms$	$12ms$	$14ms$	$24ms$	$28ms$	$35ms$	$39ms$	$137ms$

Table 4: Running time of different query cases with varied collection probabilities

of trajectories in Wenzhou is longer than that in Zhengzhou. For Time-Sensitive similarity, we enumerated all pairs and compare them in linear time, which depends on the length of trajectories. For Order-Insensitive similarity, we just need to go through all the samples once. There are more samples in Zhengzhou because there are more agents. Thus, the running time of Zhengzhou is larger than Wenzhou's.

For Order-Sensitive similarity, the running time grows quadratically with the sample probability p. Without subsampling, we compute the Order-Sensitive similarity between all pairs of agents in about 10 hours for Zhengzhou dataset, and about 2 hours for Wenzhou. With sample probability p as 10%, we can compute these similarities within several minutes, which is a big improvement for real applications.

5.4 Advice to Practitioners

We have some tips for practitioners on how to adjust parameters to obtain desired precision and recall. The similarity parameter α is typically determined by applications. A high similarity parameter α leads to a low number of retrieved instances and true positive instances. We empirically found the distribution of pairs of trajectories by the similarity parameter α based on Zhengzhou data set, shown in Figure 6. The highest similarity parameter appears to be around 0.6 and there are only a few such pairs suggesting 0.6 to be a good threshold to test.

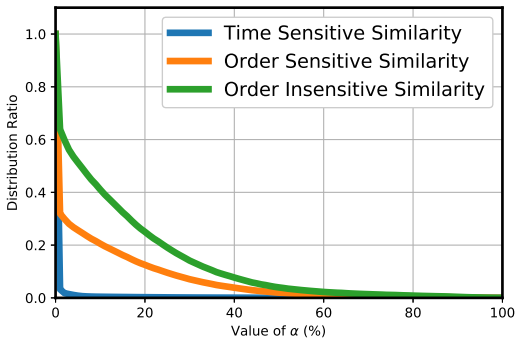


Figure 6: Zhengzhou: Distribution of the trajectory pairs based on similarity parameter

	One-Day	One-Week
Time-Sensitive similarity	4**(0.8%)**	4**(0.03%)**
Order-Sensitive similarity	46**(15%)**	52**(0.5%)**
Order-Insensitive similarity	167**(56%)**	1032**(9.4%)**

Table 5: The number of samples to uniquely identify a trajectory

With the similarity parameter α fixed, we determine sample probability p. From our experience, in most cases, the sample probability p calculated by the inequality from the Chernoff Inequality is large enough to achieve recall $\geq 95\%$.

Now, with a good sample probability p and a good recall rate, the precision might be poor, as in Figure 4(c). In this case, we can adjust the threshold coefficient k. By increasing k, precision could be increased significantly with a slight reduction to recall. According to our simulations, we can adjust the value of k from 0.8 to 1.2 to obtain a rate of at least 90% for both recall and precision.

5.5 Human Mobility Pattern

We ran experiments on using different partial similarity measures in identifying the unique agent. Human movement trajectories over a long period of time are surprisingly unique with strong personal traits. For example, four spatio-temporal points are enough to uniquely identify 95% of the individuals out of a million other [19]. This signature is a special case of our partial similarity, namely, the time-sensitive similarity. We would like to re-examine our data sets using different similarity measures in terms of how much individuality they capture.

Here, we use the dataset of Zhengzhou. In one day, there are $36,951$ agents and each one has about 500 samples on average. We use their one-week dataset, with $2,085$ agents and each trajectory has about $11,000$ vertices on average. In the experiments, we randomly select several samples from the trajectory as the signature of this agent. We want to check whether these samples and other order information could help us to identify this agent uniquely.

For example, we use Time-Sensitive similarity to identify this unique agent. The similarity parameter α is set to 1. We enumerate all the agents in the dataset and check whether any agent has the same signature. For Order-Sensitive similarity, we just check whether any agent visits all of these checkpoints in the same order as in the signature. For Order-Insensitive similarity, we test whether the agent visits all of these checkpoints in the signature.

For different signatures, we run 1,000 times to get the average successful rate for uniquely identifying an agent. Then we set the threshold as 90% and find the minimum number of samples needed. The result is shown in Table 5.

For time-Sensitive similarity, 4 spatio-temporal samples are enough to uniquely identify more than 90% individuals, in both the dataset of daily trajectories and the weekly trajectories. For order-sensitive similarity, 46 and 52 ordered samples are enough to identify most of the agents in the one-day dataset and the one-week dataset. For Order-Insensitive similarity, we need many more samples, 167 and 1,032, for the one-day dataset and the one-week dataset respectively. It is a large fraction, 56% and 9.4%, of the full trajectory length.

These findings show that human trajectories, even with significant down sampling, can still be very sparse and unique with timestamped locations. The visiting order of the locations helps, but not nearly as much. Thus removing the temporal information of a trajectory is crucial for privacy protection.

6 CONCLUSION

In this paper, we proposed a distributed trajectory collection framework to understand the properties of natural human trajectories, *individuality* and *commonality*. Due to the large dataset and high computation time of original similarity measurement methods, we design three partial similarities, which are suitable for real-world trajectory analysis. In our framework, we use random sampling to reduce the data size. Through analysis and simulation, we show that with less than 10% of the original trajectories we can get a recall more than 90% and reasonably high precision. Thus, our framework provides an efficient solution for processing massive trajectory data.

Acknowledgement The authors would like to acknowledge supports from NSF CNS-1618391, DMS-1737812, OAC-1939459.

REFERENCES

[1] Code. https://github.com/SBUhaotian/Partial_Similarity.

[2] CRAWDAD data set. http://crawdad.cs.dartmouth.edu.

[3] Peyman Afshani and Anne Driemel. 2018. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 898–917.

[4] Helmut Alt and Maike Buchin. 2010. Can we compute the similarity between surfaces? *Discrete & Computational Geometry* 43, 1 (2010), 78.

[5] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications* 5, 01n02 (1995), 75–91.

[6] Helmut Alt, Christian Knauer, and Carola Wenk. 2004. Comparison of distance measures for planar curves. *Algorithmica* 38, 1 (2004), 45–58.

[7] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. 2006. Fréchet distance for curves, revisited. In *European Symposium on Algorithms*. Springer, 52–63.

[8] Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. 2018. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 279–288.

[9] Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J Katz, and Micha Sharir. 2014. The Discrete FréChet Distance with Shortcuts via Approximate Distance Counting and Selection. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry (SOCG'14)*. ACM, New York, NY, USA, 377:377–377:386.

[10] Julian Baldus and Karl Bringmann. 2017. A fast implementation of near neighbors queries for Fréchet distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 99.

[11] Karl Bringmann. 2014. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 661–670.

[12] Karl Bringmann and Marvin Kunnemann. 2015. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, 79–97.

[13] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. 2014. Four soviets walk the dog—with an application to Alt's conjecture. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1399–1413.

[14] Kevin Buchin, Maike Buchin, and Yusu Wang. 2009. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 645–654.

[15] Maike Buchin, Anne Driemel, and Bettina Speckmann. 2014. Computing the Fréchet distance with shortcuts is NP-hard. In *Proceedings of the thirtieth annual symposium on Computational geometry*. ACM, 367.

[16] Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. 2019. FRESH: Fréchet Similarity with Hashing. In *Algorithms and Data Structures*. Springer International Publishing, 254–268.

[17] Erin Wolf Chambers, Eric Colin De Verdiere, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. 2010. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry* 43, 3 (2010), 295–311.

[18] Mark de Berg, Cook, Atlas F., and Joachim Gudmundsson. 2013. Fast Fréchet queries. *Comput. Geom.* 46, 6 (Aug. 2013), 747–755.

[19] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the Crowd: The privacy bounds of human mobility. *Sci. Rep.* 3 (2013), 1376.

[20] J Ding, C Ni, M Zhou, and J Gao. 2017. MinHash Hierarchy for Privacy Preserving Trajectory Sensing and Query. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 17–28.

[21] J Ding, C C Ni, and J Gao. 2017. Fighting Statistical Re-Identification in Human Trajectory Publication. *Proceedings of the 25th ACM SIGSPATIAL* (2017), 1–4.

[22] David H Douglas and Thomas K Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization* 10, 2 (1973), 112–122.

[23] A Driemel and S Har-Peled. 2013. Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts. *SIAM J. Comput.* 42, 5 (Jan. 2013), 1830–1866.

[24] Anne Driemel, Sariel Har-Peled, and Carola Wenk. 2012. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry* 48, 1 (2012), 94–127.

[25] A Driemel and F Silvestri. 2017. Locality-sensitive hashing of curves. In *33rd International Symposium on Computational Geometry, SoCG 2017*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 37:1–37:16.

[26] Michael Godau. 1991. A natural metric for curves—computing the distance for polygonal chains and approximation algorithms. In *Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 127–136.

[27] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *nature* 453, 7196 (2008), 779.

[28] Piotr Indyk. 2002. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the eighteenth annual symposium on Computational geometry*. 102–106.

[29] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 604–613.

[30] Wesley G Jennings, Chris L Gibson, Jeffrey T Ward, and Kevin M Beaver. 2008. "Which group are you in?": A preliminary investigation of group-based publication trajectories of criminology and criminal justice scholars. *Journal of Criminal Justice Education* 19, 2 (2008), 227–250.

[31] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. 2008. Convoy Queries in Spatio-Temporal Databases. *2008 IEEE 24th International Conference on Data Engineering (ICDE 2008)* (2008), 1457–1459.

[32] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. 2008. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment* 1, 1, 1068–1080.

[33] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. 2005. *Advances in Spatial and Temporal Databases: 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005. Proceedings*. Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter On Discovering Moving Clusters in Spatio-temporal Data, 364–381. https://doi.org/10.1007/11535331_21

[34] Panagiota Katsikouli, Rik Sarkar, and Jie Gao. 2014. Persistence based online signal and trajectory simplification for mobile devices. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 371–380.

[35] Qingquan Li, Zhe Zeng, Bisheng Yang, and Tong Zhang. 2009. Hierarchical route planning based on taxi GPS-trajectories. In *2009 17th International Conference on*

Geoinformatics. IEEE, 1–5.

[36] Faisal Orakzai, Thomas Devogele, and Toon Calders. 2015. Towards distributed convoy pattern mining. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 50.

[37] Claudio Piciarelli, Christian Micheloni, and Gian Luca Foresti. 2008. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for video Technology* 18, 11 (2008), 1544–1554.

[38] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceeding of the 5th international conference on extending database technology (EDBT'96)*. 3–17.

[39] Naohiko Suzuki, Kosuke Hirasawa, Kenichi Tanaka, Yoshinori Kobayashi, Yoichi Sato, and Yozo Fujino. 2007. Learning motion patterns and anomaly detection by human trajectory analysis. In *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 498–503.

[40] Mohammed M Vazifeh, P Santi, G Resta, SH Strogatz, and C Ratti. 2018. Addressing the minimum fleet problem in on-demand urban mobility. *Nature* 557, 7706 (2018), 534.

[41] Florian Verhein. 2009. Mining complex spatio-temporal sequence patterns. In *Proceedings of the 2009 SIAM international conference on data mining*. SIAM, 605–616.

[42] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1100–1108.

[43] Zuchao Wang, Min Lu, Xiaoru Yuan, Junping Zhang, and Huub Van De Wetering. 2013. Visual traffic jam analysis based on trajectory data. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2159–2168.

[44] Hyunjin Yoon and Cyrus Shahabi. 2009. Accurate Discovery of Valid Convoys from Moving Object Trajectories. In *2009 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 636–643.

[45] Yang Yue, Yan Zhuang, Qingquan Li, and Qingzhou Mao. 2009. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In *2009 17th international conference on geoinformatics*. IEEE, 1–6.

[46] Mohammed J Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine learning* 42, 1-2 (2001), 31–60.

[47] Mengyu Zhou, Kaixin Sui, Minghua Ma, Youjian Zhao, Dan Pei, and Thomas Moscibroda. 2016. Mobicamp: A campus-wide testbed for studying mobile physical activities. In *Proceedings of the 3rd International on Workshop on Physical Analytics*. ACM, 1–6.