MinHash Hierarchy for Privacy Preserving Trajectory Sensing and Query

Jiaxin Ding, Chien-Chun Ni
Stony Brook University
{jiading,chni}@cs.stonybrook.edu

Mengyu Zhou
Tsinghua University
zhoumy12@mails.tsinghua.edu.cn

Jie Gao
Stony Brook University
jgao@cs.stonybrook.edu

ABSTRACT

In this work, we study privacy preserving trajectory sensing and query when *n* mobile entities (e.g., mobile devices or vehicles) move in an environment of *m* checkpoints (e.g, WiFi or cellular towers). The checkpoints detect the appearances of mobile entities in the proximity, meanwhile, employ the MinHash signatures to record the set of mobile entities passing by. We build on the checkpoints a distributed data structure named the MinHash hierarchy, with which one can efficiently answer queries regarding popular paths and other traffic patterns. The MinHash hierarchy has a total of near linear storage, linear construction cost, and logarithmic update cost. The cost of a popular path query is logarithmic in the number of checkpoints. Further, the MinHash signature provides privacy protection using a model inspired by the differential privacy model. We evaluated our algorithm using a large mobility data set and compared with previous works to demonstrate its utilities and performances.

CCS CONCEPTS

•Security and privacy →Privacy protections; •Networks →In-network processing;

KEYWORDS

MinHash Hierarchy, Privacy protection, Trajectory sensing, Path queries

ACM Reference format:
Jiaxin Ding, Chien-Chun Ni, Mengyu Zhou, and Jie Gao. 2017. MinHash Hierarchy for Privacy Preserving Trajectory Sensing and Query. In *Proceedings of The 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, Pittsburgh, PA USA, April 2017 (IPSN 2017),* 12 pages.
DOI: http://dx.doi.org/10.1145/3055031.3055076

1 INTRODUCTION

The technology development in localization mechanisms and the wide spread of mobile devices have enabled the capability to gather an enormous amount of real-world mobility data. Such data can be valuable in a variety of ways. Human mobility patterns can be used to aid civil planning such as improving transportation systems and city construction. How people move around in the living space can be used for activity recognition, energy management, health monitoring, just to name a few.

With such enormous trajectory data, we have new challenges on the front of data management, analysis, and applications. The motivation of this paper is two folds. The first issue is *real-time* mobility sensing and queries, and the second issue is privacy protection. The two topics are closely related. In almost all past works, mobility trajectories are first gathered into a trajectory database before any analysis is done, which does not meet real-time requirements. It is a waste of efforts to collect and archive all trajectory data as traffic congestion or frequent traffic patterns may change fast and soon become outdated. In addition, user queries for traffic information generally exhibit spatial and temporal locality – one cares mostly about the traffic situation near the current location and cares less about the traffic condition faraway. With both data locality and query locality in consideration, it is desirable to have a distributed system to keep data near where it is generated.

Collecting all mobility traces for a long time scale at a central place may also raise privacy concerns. Human movement trajectories over a long period of time are surprisingly unique with strong personal traits. Simply removing the personal identifiers of the records is far from enough. Frequently visited locations or repeated motifs can be related to activities that reveal important locations such as home or work locations [19], which results in the high predictability of individual locations [39]. Frequent co-location events can be used to infer social tie structures [44]. Natural motion trajectories have fairly distinctive signatures – even the coarse knowledge of someone's whereabouts can be used to identify a single trajectory out of a million others [12]. Sanitization of long-term mobility traces is a non-trivial task. In this work, we do not archive long-term trajectory traces but only keep, in a distributed manner, mobile entity appearance/traffic situation in recent history.

Our results. We would like to address the problem of finding popular paths in real time, with respect to the current trajectory flow and recent history. Since the trajectories are naturally spatially spread out, we develop a distributed sensing framework which makes use of a set of spatially located *checkpoints*, (e.g., the roadside units in vehicular network setting, WiFi hotspots or cellular towers). These checkpoints can detect and record the mobile entities (e.g., the vehicle with GPS or people with mobile devices) passing by. Instead of collecting all trajectories through these checkpoints first, we would like to directly work with the checkpoints and build a distributed data structure that 1) is compact, of modest size; 2) allows efficient and continuous updates when new traffic data comes in; 3) answers real-time queries of popular patterns; 4) protects user privacy. Our idea is to collect the trajectory data at the 'checkpoints' by using the MinHash mechanism. In particular, each mobile entity is assumed to have a unique ID. At each checkpoint, all mobile agents that pass by would report the hashed values of their IDs. Each checkpoint only keeps the *minimum* hash value of all agents that pass by in a recent time frame. This is repeated by using k independent hash functions at all checkpoints. The min-hash values at these checkpoints now form a *MinHash signature*.

Albeit the simplicity and compactness of the MinHash signature, it has a number of attractive properties. Namely, the signature at one checkpoint can be used to estimate the number of mobile agents that have recently appeared – the larger this number, the smaller the MinHash value is. For two physically adjacent checkpoints, the intersection of their MinHash signatures can be used to estimate the Jaccard coefficient, i.e., the similarity of the mobile entities they have seen. This is useful for mining popular trajectories. Last but most importantly, the MinHash signature protects personally identifiable information, formulated using ε-differential privacy. If any particular trajectory is removed from a set of trajectories, the MinHash signature remains the same with high probability over the random choices made in the hash functions. Thus, with such signatures personally relevant information is protected, while cumulative traffic patterns can still be effectively extracted.

Our most important contribution is the *MinHash hierarchy* on the checkpoints, with which one can efficiently answer popular path queries. The MinHash hierarchy built in levels takes randomly sampled checkpoints in a recursive manner, where a checkpoint on level i appears in level $i+1$ with fixed probability. At each level, the checkpoints record whether they are 'neighbors' on this level – defined as whether there is a popular path connecting them without other checkpoints of the same level in between. The intersection of all the MinHash signatures along this popular path is also stored at the two end checkpoints. It turns out that the total storage requirement of the hierarchy would be near linear in the number of checkpoints. The depth of the hierarchy is logarithmic in the number of checkpoints. With such a hierarchy, which is stored in the checkpoints in a distributed manner, one can issue a number of queries of all the popular paths between two locations, the popularity of a given path, as well as all the popular paths starting from one point. The first two queries can be answered in time only logarithmic in the number of checkpoints while the last one in polylogarithmic time. Both the MinHash signature and the hierarchy can be made time-evolving.

In this paper, we prove rigorously the above claims. We also evaluate the MinHash hierarchy with large trajectory sets. Compared with previous frequent pattern mining algorithms (SPADE[51], PrefixSpan [24]), our MinHash hierarchy outperforms them in both computation and storage complexity. As for the communication complexity of popular path queries, we show our algorithm scales well with heavy tasks compared to the brute force method.

In the following we start by reviewing the related work. Then we define the settings, introduce MinHash for trajectory analysis, and prove that the MinHash signature is privacy preserving. We thereafter propose the MinHash hierarchy and efficient algorithms to answer the popular path queries. The performance evaluation is presented at the end.

2 RELATED WORK

Frequent pattern mining. The study of spatial temporal patterns that summarize the collective behaviors of moving objects has been a subject of study under the name of *sequential pattern mining*. The frequent sequential pattern (FSP) problem [2] is defined over a database of sequences D, where each element is a time-stamped set of items called an itemset. And the problem is to find all sequences that are frequent in D, i.e., appearing as subsequences in a large percentage of all sequences of D. Here a sequence α is a subsequence of β if all the elements of α appear in the same order in β, though not necessarily to be consecutive. This definition has been widely adopted in mining of a variety of data sets, from DNA sequences to e-commerce. A number of well known algorithms have also been proposed (e.g., Generalized Sequential Patterns (GSP) [41], PrefixSpan [24], SPAM [3], IncSpan [8], and SPADE [51]). Handling trajectories as discrete items is missing one important aspect of location proximity. Thus in a number of previous work [17, 28], approximation or clustering is used to group nearby locations together. In the work above, the temporal dimension is not considered and instead the trajectories from different time window are put together to generate the frequent patterns. This has merit in certain applications that are neutral or blind to the time dimension. But our setting is different.

Some other works related to frequent pattern mining are called *Convoy Pattern Mining*[25, 26]. Convoys are groups of objects that are density–connected in continuous trajectory snapshots. In [25, 26], Jeung et al. proposed a method to find the convoy by interpolate an object's location for snapshots and filter-and-refine the trajectory points. Their method has later been discovered by Yoon and Shahabi[49] to have the accuracy problem. Yoon and Shahabi[49] redefined the term to be *valid* convoy, and proposed a method called VCoDA to solve the problem. In recent work, Orakzai et al.[36] suggested a distributed solution of this convoy problem.

There are also some grid-based works for frequent pattern mining. In [27], Uehara et al. proposed a MapReduce method of a hierarchical grid with quad-tree search that can detect different levels of granularity. In [43], Verhein proposed k-STARs, which is also a hierarchical lattice based method to mine the trajectory patterns in different resolution.

Existing frequent sequential pattern mining mainly take two general approaches: the bottom-up Apriori-based approach [41, 51], and the top down FP-growth based approach [23, 24]. The way that the trajectory data is stored is more similar to the Apriori-based approach such that frequent paths appear and bubble up in the hierarchy. While queries to the hierarchy take a more top-down approach and directly start from paths kept on high levels of the hierarchy.

Frequent trajectory mining. There is also prior work that considers the temporal relations between events, or patterns that contain both spatial and temporal information [18, 32]. Giannotti *et al.* [18] handled GPS trajectories and identify popular regions and then connected them with the temporal patterns. Liu *et al.* [32] focused on trajectory pattern mining in noisy RFID data. The mined frequent trajectories are used for prediction or classification of the trajectories [31], labelling of locations [52], or deriving mobile user behaviors [33]. MinHash was used to measure the similarity of two trajectories in [20, 45]. There was no prior work in mining trajectory patterns in a distributed setting as what is done in this paper.

Protecting trajectory privacy. Plenty of work has been conducted to protect the privacy of a *single* location, when the user submits a query to location-based services (LBS) on mobile phones. A nice survey can be found in [9, 38]. For query privacy (i.e., not identifying the user issuing the query), the most common approach is to use k-anonymity measure [42] in which the user location is indistinguishable from at least $k-1$ other users. Cloaking boxes were used [21, 34] in which one user query is packed in a box with $k-1$ other users in the neighborhood. Only the box is sent to the LBS. For location privacy (i.e, not identifying the location of a user), one common metric used is location entropy, which characterizes the uncertainty of the location information an adversary can extract from LBS queries. Most methods use perturbations of the true locations to confuse the adversary (for example, see [48]). Notice that the reported location cannot be too far away from the true location as otherwise the location based query will become useless. There is a tradeoff of location privacy versus query utility.

Protecting the privacy of the trajectory data is not investigated as much. Previous work is along the following directions: spatial temporal cloaking [46, 47], which often suffers from growing size of the boxes; mixed zones [4, 15] in which users' pseudonyms are mixed inside the zone. In the case of protecting privacy during publication of trajectories to a third party, three ideas have been mainly used: clustering based [1, 16], generalization based [35] and adding dummies [30, 50]. In all of them the k-anonymity is used as a measure of privacy protection. While k-anonymity protects against identifying an individual in a group of k entries, it does not provide sufficient protection against attribute – i.e., the link of an attribute to an individual may disclose sensitive information.

3 SETTINGS AND MODELS

3.1 Network and Privacy Models

Checkpoints. We consider the setting in which n mobile entities move within a geographical region in which m checkpoints are spatially distributed. Typically $n \gg m$. Each checkpoint can detect the appearances of mobile entities in close proximity and maintain a signature as the sketch of the ID set of mobile entities passing by. We assume that the checkpoints have sufficient density to distinguish between different trajectories. In particular, we assume that whenever two different paths meet at v, diverge and then meet at w again, they each visit at least one other different checkpoint.

Network Model. The checkpoints, being roadside units, WiFi access points or cellular base stations, are naturally connected to each other by the Internet. We exploit the existing communication infrastructure and assume that any checkpoint can communicate with any other checkpoint with unit cost. When we discuss communication cost we count the number of message transmissions among these checkpoints.

Privacy Model and Adversaries. For privacy protection, we use the idea inspired by differential privacy [13], which is proposed for protecting the privacy of a query database. Given a database and user queries on the database, the queries are answered by a randomized algorithm which is ε-differentially private if for both datasets, D_1 and D_2, that differ on a single element (i.e., data of one person), and all $S \subseteq \mathrm{Range}(\mathcal{A})$,

$$\Pr[\mathcal{A}(D_1) \in S] \le e^{\varepsilon} \times \Pr[\mathcal{A}(D_2) \in S],$$

where the probability is taken over the coins of the algorithm and $\mathrm{Range}(\mathcal{A})$ denotes the output range of the algorithm $\mathcal{A}$.

Our model is a bit different. In the database setting, all data is stored on a trusted server and noise is added to the query result to ensure that one cannot infer the information of any particular element. However, if the server is hacked, all the information is revealed. In our case, trajectory data is stored in a distributed manner, locally at the checkpoints of detection. We assume that the checkpoints are trusted when they collect data and generate signatures. Even if an adversary collects the signatures from all checkpoints, the adversary still cannot infer information about any single trajectory with high probability. This is a stronger condition as any information inferred from the signature will automatically have the differential privacy property.

As will be described later, we use hash functions in the data structure. We assume that the mobile entities, as well as checkpoints, have the same set of hash functions so they are able to compute the hash values themselves. If needed, the hash functions can be taken as cryptographic one-way hash functions [37] so that adversaries knowing the hashed values cannot inverse the functions and recover the IDs of the mobile entities.

Time Evolving Signatures. Trajectories in a long period of time can reveal personal information, such as frequently visited locations [12]. This concern is substantially reduced if only short trajectories are kept [40]. Therefore, in addition to spreading trajectory information spatially on the checkpoints, we work on real-time settings and partition trajectories by short time intervals. At the beginning of each time interval, we re-start the hash signatures (by choosing random seeds). We keep the signatures for the latest t time intervals. Any information older than t slots away is removed. As time evolves, the oldest signature is overwritten. Both the length of the interval and the value t are determined by applications.

3.2 Popular and Consistent Paths

In this work we are mostly interested in paths that are travelled by a large number of mobile entities (either in absolute value, or a fractional amount) in a recent period of time. There are a number of motivations for supporting this type of queries. First, the popular paths represent an important class of significant patterns in the trajectory set. If two sets of trajectories are similar, then the popular paths in these sets must be similar too. Thus comparing popular paths in two different trajectory sets can be a quick detector of anomalies. Second, from a user's perspective, it is often of interest to know what is the popular (most used) path to go to a given destination. Thirdly, learning about the paths that are travelled a lot can help with many other city planning applications, e.g., deciding on the location of gas stations, convenient stores, improving public transportation system, optimizing the road types (one-way, two-ways) etc. Last but not least, popular paths, by definition, are typically of less concerns from a privacy perspective.

Assume there is a path P that goes through the sequence of checkpoints $v_1, v_2, \cdots, v_\ell$. Let I_i be ID set of mobile entities that visited checkpoint v_i for $1 \le i \le \ell$. We introduce two definitions, ψ-popular path and ϕ-consistent path, that capture different features in traffic patterns.

Definition 3.1. P is ψ-popular if there are at least ψ mobile entities that travel along P, i.e., $|\cap_{i=1}^{\ell} I_i| \ge \psi$.

This definition is similar to the definition of flocks [22] and convoys [25], which refer to groups of mobile entities travelling together during consecutive time slots.

We use the Jaccard similarity coefficient of $I_1, I_2, \cdots, I_\ell$ to define consistent paths. For two sets A and B, the *Jaccard similarity coefficient* is defined as: $J(A, B) = |A \cap B| \,/\, |A \cup B|$. We denote the Jaccard similarity coefficient of P to be $J(I_1, I_2, \cdots, I_\ell)$, the Jaccard similarity coefficient of the ID sets along the sequence P:

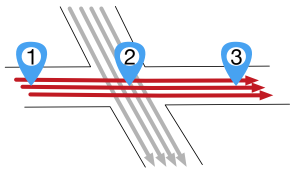
$$J(I_1, I_2, \cdots, I_\ell) = \frac{|I_1 \cap I_2 \cap \cdots \cap I_\ell|}{|I_1 \cup I_2 \cup \cdots \cup I_\ell|}.$$

Definition 3.2. If the Jaccard similarity coefficient of a path P is at least ϕ, we define P as a *ϕ-consistent path*.

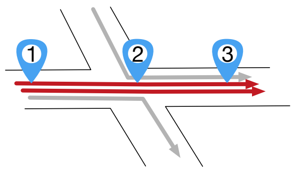
This is inspired by the idea of moving clusters [29], which is a sequence of spatial clusters on different time slots, where the Jaccard similarity of clusters in two consecutive time slots is at least ϕ.

Discussions. The ψ-popular path and ϕ-consistent path capture different features of traffic patterns. ψ is an absolute value and ϕ is a fraction. For example, take $\psi = 3, \phi = 1/2$ and consider the path $P = [1, 2, 3]$ in Figure 1, where each line represents the trajectory of a vehicle. In Figure 1(a), the checkpoint 2 is located at a crossing that sees a lot of vehicles traveling in a different direction such that the fraction of mobile entities on P, $3/7$, does not reach ϕ. Thus P is a ψ-popular path, but not a ϕ-consistent path. On the other hand, there could be paths with light traffic such that they do not meet the criteria for a ψ popular path, as shown in Figure 1(b). But if almost all traffic on P follow the same path, this is an interesting pattern that is captured by the ϕ-consistent path. ϕ-consistent path captures the pattern where a large fraction of traffic follows the same path, no matter whether the traffic is heavy or light. The two definitions have their respective merits.

Almost all theoretical results for the ϕ-consistent path in our work also apply to the ψ-popular path. Therefore in the following discussion we mainly focus on ϕ-consistent paths.



(a) ψ-popular, not ϕ-consistent.



(b) ϕ-consistent, not ψ-popular.

Figure 1: ψ-popular path versus ϕ-consistent path. $\psi = 3$, $\phi = 1/2$. $P = [1, 2, 3]$.

A useful observation is that any subset of a ϕ-consistent path P is also ϕ-consistent. For any subpath, the intersection of the sets (i.e., numerator of the Jaccard coefficient) always gets no smaller while the union of the sets (i.e., the denominator) always gets no greater. Thus the Jaccard coefficient of a subset of P is no smaller than the Jaccard coefficient of P.

Thus it is only interesting to consider the *maximal* ϕ-consistent path, which cannot be extended further. In the following we show a useful property on the maximum number of maximal ϕ-consistent paths that start from one checkpoint.

LEMMA 3.3. *The number of maximal ϕ-consistent paths starting from any checkpoint v is bounded by $1/\phi$.*

PROOF. For a maximal ϕ-consistent path P that goes through the sequence of checkpoints $v_1, v_2, \cdots, v_\ell$, we define the set of supporters $U(P)$ as the ID set of the mobile entities which travel along P. Precisely, $U(P) = \cap_{i=1}^{\ell} I_i$, where I_i is the ID set of v_i.

We consider all the maximal ϕ-consistent paths $\{P_1, P_2, \cdots, P_r\}$ that start at v_1. Since any two such ϕ-consistent paths are different for at least some of the checkpoints, the sets of supporters for them are disjoint (as one trajectory cannot visit two different paths simultaneously). Since each set of supporters must have at least $|I_1|\phi$ mobile entities, the total number of such ϕ-consistent paths is bounded by $r \le 1/\phi$. □

If the maximum number of mobile entities passing by any checkpoint is n^*, with the same proof, we have the similar lemma that the number of maximal ψ-popular paths starting from any checkpoint v is bounded by n^*/ψ.

In this paper we consider the following three types of popular path queries:

- ϕ-consistent/ψ-popular paths for a source-destination pair.
- All ϕ-consistent/ψ-popular paths starting from a checkpoint.
- The Jaccard similarity coefficient/the mobile entity number of a path.

4 MINHASH

4.1 MinHash Definition

Given a set T, let $h : T \to [0, 1]$ be a hash function that maps members of T to distinct numbers drawn uniformly at random from interval $[0, 1]$. For any set $D \subset T$, define the value $\hat{h}(D)$ to be the *MinHash* value of D, where $\hat{h}(D) = \min\{h(x)|x \in D\}$ [6, 7]. MinHash can be used to estimate set cardinality and Jaccard similarity coefficient.

Estimate set cardinality. For any set $D \subset T$, MinHash can be applied to estimate cardinality $|D|$ of D [10, 14]. Suppose we use k different hash functions $h_1, h_2, \cdots, h_k$ and $\hat{h}_1(D), \hat{h}_2(D), \cdots, \hat{h}_k(D)$ are the k corresponding MinHash values, it is shown in [10] that $(k-1)/\sum_{i=1}^{k} \hat{h}_i(D)$ is an unbiased estimator of $|D|$. If $k \ge 2 + 1/(\delta^2\eta)$, with probability at least $1 - \eta$,

$$\left| \frac{k-1}{\sum_{i=1}^{k} \hat{h}_i(D)} - |D| \right| < \delta |D|.$$

Estimate Jaccard similarity coefficient. Denote by $\hat{h}(A)$ and $\hat{h}(B)$ the MinHash values of A and B with hash function h respectively. $\hat{h}(A) = \hat{h}(B)$ if and only if the element with the minimum hash value with respect to h lies in the intersection $A \cap B$. That is,

$$\Pr\{\hat{h}(A) = \hat{h}(B)\} = J(A, B).$$

Given k hash functions, $h_1, h_2, \cdots, h_k$, we get k MinHash values for A and B respectively. Let $X_1, X_2, \cdots, X_k$ be the 0/1 random indicator variables. $X_i = 1$ iff $\hat{h}_i(A) = \hat{h}_i(B)$. Take $X = \sum_i^k X_i$, $E(X/k) = J(A, B)$. Thus X/k is an unbiased estimator of the Jaccard similarity coefficient [6].

In our scenario, we are interested in Jaccard coefficients that are at least ϕ. If $k \geq 3 \log \frac{2}{\eta}/(\phi\delta^2)$, by Chernoff inequality,

$$\Pr\left\{\left|\frac{X}{k} - J(A, B)\right| \geq \delta J(A, B)\right\} \leq 2\exp(-\frac{k\phi\delta^2}{3}) \leq \eta.$$

That is, with the probability of at least $1 - \eta$,

$$\left|\frac{X}{k} - J(A, B)\right| < \delta J(A, B).$$

Estimate cardinality of set intersections. In the definition of ψ-popular path, we are interested in the estimation of the cardinality of set intersections. Combined with the above analyses, for set A and B, denote $J^*(A, B)$ as the estimation of $J(A, B)$, $|A \cup B|^*$ as the estimation of $|A \cup B|$. We can use $J^*(A, B)|A \cup B|^*$ to estimate $|A \cap B|$, such that for $k \geq \max\{2 + 1/(\delta^2\eta), 3 \log \frac{2}{\eta}/(\phi\delta^2)\}$, with probability at least $1 - \eta$,

$$\left| J^*(A, B)|A \cup B|^* - |A \cap B| \right| < 3\delta|A \cap B|.$$

All above, we would like to remark: 1) MinHash estimation is robust to the number of appearances of the same element and produces a count of distinct elements; 2) the error bounds of the above analyses can be applied to the estimations of the multiple sets; 3) with MinHash, we can get a good estimation of the Jaccard similarity as well as the cardinality of the set intersections, with space requirement of $O(k)$ instead of $O(n)$, where k only depends on the approximation error bounds.

The evaluations of the errors in the experiments can be found in [6, 10]. In the following , we would not explicitly write down the approximation factor whenever we use MinHash to approximate the Jaccard coefficient.

4.2 MinHash Signature

Given the set $C = \{1, 2, \cdots, n\}$ consisting of the IDs of n mobile entities, we have k hash functions $h_1, h_2, \cdots, h_k$, each of which maps mobile entity $i \in C$ to distinct uniform random numbers drawn from $U(0, 1)$, denoted as $[h_1(i), h_2(i), \cdots, h_k(i)]$. Therefore, each mobile entity i carries k hash values as its signature.

Denote $V = \{1, 2, \cdots, m\}$ as the ID set of m checkpoints. Each checkpoint $j \in V$ also keeps a signature of k values corresponding to the k hash functions denoted as $S_j = [s_{j1}, s_{j2}, \cdots, s_{jk}]$. When a mobile entity i passes by a checkpoint j, the mobile entity's signature is passed to the checkpoint. The checkpoint j compares i's signature with its current signature; maintains the smaller values for each hash function as its new signature, to be precise, updates $s_{jl} = h_l(i)$ only if $h_l(i) < s_{jl}$. That is, each checkpoint maintains the k MinHash values as its signature, as illustrated in Figure 2. We define this signature to be the *MinHash signature* of the checkpoint.

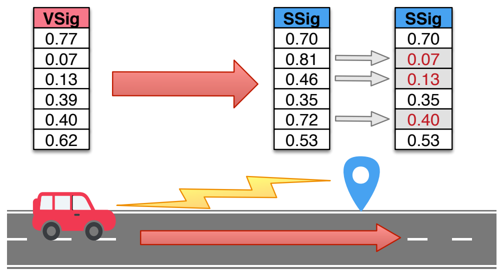


Figure 2: When a vehicle passes by a checkpoint, it sends its signature *VSig* to the checkpoint. The checkpoint compares the received *VSig* with its signature *SSig* and updates wherever *VSig* is smaller.

Since the checkpoint updates the MinHash values corresponding to each hash function only if it receives a smaller one, the chance of this update event decreases when the number of mobile entities passing by increases. The total update cost is only logarithmic in the number of mobile entities appearing, as is proved in [11].

4.3 MinHash Operations

The operations on synopses for distinct value estimation, including MinHash, has been studied in [5, 11]. Similarly, we also introduce union and intersection operations in our work. Given the MinHash signatures of two checkpoints $S = [s_1, s_2, \cdots, s_k]$, $\tilde{S} = [\tilde{s}_1, \tilde{s}_2, \cdots, \tilde{s}_k]$, we define the union and intersection as following.

Union. We define the *union* operation of the MinHash signature S and $\tilde{S}$ as $S \cup \tilde{S} = [x_1, x_2, \cdots, x_k]$ where

$$x_i = \min\{s_i, \tilde{s}_i\} \quad \text{for } i = 1, 2, \cdots, k.$$

The union operation can be used to estimate set cardinality. This can be generalized to multiple checkpoints along a road, or in a certain geographical region to estimate the total number of mobile entities that have visited this region.

Intersection. The *intersection* of the MinHash signature S and $\tilde{S}$ is defined as $S \cap \tilde{S} = [y_1, y_2, \cdots, y_k]$ where

$$y_i = \begin{cases} s_i & \text{if } s_i = \tilde{s}_i \\ \perp & \text{if } s_i, \tilde{s}_i = \perp \text{ or } s_i \neq \tilde{s}_i \end{cases}$$

for $i = 1, 2, \cdots, k$. $\perp$ is a symbol meaning undefined.

The intersection of MinHash signatures offers an estimate for the Jaccard similarity. Let ω be the number of elements which are not $\perp$ in the result. ω/k approximates the Jaccard similarity coefficient.

Further, we define the MinHash signature of path P that goes through $v_1, v_2, \cdots, v_\ell$, as the intersection of the MinHash signatures of the checkpoints on path P, that is, $\cap_{i=1}^{\ell} S_{v_i}$. Thereafter, we can calculate the Jaccard similarity of P to tell whether P is a ϕ-consistent path. Besides, we can take the union of the signatures of checkpoints on P to estimate the cardinality of all the mobile entities passing by any of the checkpoints on P, and multiply the result with the Jaccard similarity of P to find the number of mobile entities sharing P. Hence, we can tell whether P is a ψ-popular path. The Figure 3 illustrates a toy example of the ϕ-consistent and ψ-popular paths.

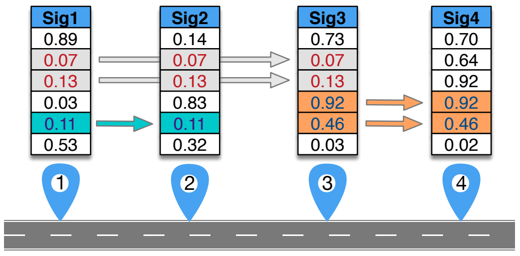


Figure 3: If $\phi = 2/6$, the path $[1,2,3]$ and $[3,4]$ are both ϕ-consistent paths. We can estimate the cardinality of mobile entities visiting at least one checkpoint on path $[1,2,3]$ as $(6-1)/(0.14+0.07+0.13+0.03+0.11+0.03) \approx 10$. Multiplied by the Jaccard similarity of the path, $2/6$, the number of mobile entities sharing the path is 3. Hence the path is a ψ-popular path if $\psi = 3$.

5 PRIVACY PROTECTION

Assume we want to conduct the trajectory pattern mining within a time interval $[t_1, t_2]$, where t_2 could be the current time. Each checkpoint $j \in V$ sets all elements in its MinHash signature S_j to ∞ at time t_1 and starts to update S_j until time t_2. In this interval, the trajectory of a mobile entity is formed as the sequence of time-stamped checkpoints it passes by contiguously. Trajectory set D is the collection of all the n trajectories. The MinHash signature of the trajectory set D consists of the MinHash signatures of all m checkpoints in V, denoted by $S(D) = [S_1, S_2, \cdots, S_m]$.

Now we prove that the MinHash signature of a trajectory set provides a decent method to protect individual user privacy. We show that for two sets D and D' of trajectories that only differ by one, they have the same signature with high probability. Here we require that the total number of mobile entities seen by a checkpoint is beyond a minimum requirement. We do not include the checkpoints before they see at least n' mobile agents, $n' = \Omega(km)$. The checkpoints that only see a small number of IDs have a good chance to reveal information about these mobility traces. In the extreme setting, if only one mobile entity is present, then the entire mobility trace can be plotted easily. But this is not an interesting setting in practice. Typically we have a large number of mobile entities moving around in the environment. The number of mobile entities n is typically much larger than the number of checkpoints m and k (which is typically a small constant). In this setting, the MinHash signature satisfies ε-differential privacy. It means adding one more trajectory to a trajectory set almost changes nothing in the MinHash signature of the trajectory set, therefore the chance of identifying a specific trajectory is low.

Theorem 5.1. *For two trajectory set $D, \tilde{D}$ where $\tilde{D}$ contains D and one extra trajectory, the probability of the output of the MinHash signature $S(D), S(\tilde{D})$ satisfies:*

$$\Pr\{S(D) = S^*\} \le e^{\varepsilon} \times \Pr\{S(\tilde{D}) = S^*\},$$

where $\varepsilon = km/n'$. Here each checkpoint sees at least n' mobile entities.

Proof. For the new mobile entity in $\tilde{D}$, the probability of its ith hash value of its signature appearing in the MinHash signature of a specific checkpoint p is at most $1/n'$. Therefore the probability that the MinHash signature of $\tilde{D}$ being the same as that of D is at least

$$\prod_{i=1}^{m}(1-\frac{1}{n'})^k = (1-\frac{1}{n'})^{km} \ge \exp(-\frac{km}{n'}).$$

Therefore the claim is true. □

The differential privacy property is stronger if each checkpoint has fewer hash functions, while the accuracy of estimating cardinality or Jaccard similarity is worse. This is a tradeoff between the privacy and the accuracy.

6 MINHASH HIERARCHY

With the MinHash signature now we propose the MinHash hierarchy with which we can answer popular path queries efficiently. The process of constructing a hierarchy structure for the ψ-popular path is almost the same as the process for the ϕ-consistent path. We can make two small modifications to change a hierarchy for ϕ-consistent paths to a hierarchy for ψ-popular paths. 1) For ϕ-consistent path, we take the intersections of the signatures of the checkpoints on a path to get the Jaccard similarity, while for ψ-popular path, we also need to take the union of the checkpoint signatures to estimate the cardinality. Multiplying the cardinality with the Jaccard similarity, we can tell whether the path is a ψ-popular path. The additional steps don't increase the bound of complexity. 2) In the following analyses for the complexity, if we change the ϕ to $\frac{\psi}{n^*}$, where n^* is the maximum number of mobile entities seen by any checkpoint, we get the bound for the ψ-popular paths.

In the following, we focus on the ϕ-consistent path. We model the checkpoints as a *directed* graph $G(V, E)$. The node set is the checkpoint set V. If there is a mobile entity that visits checkpoint v and w consecutively, there is a directed edge from v to w in E.

To organize the ϕ-consistent paths, we first define two functions Path and Sig,then we define the MinHash hierarchy.

Definition 6.1. For any nodes v, w, Path(v, w) contains the set of all ϕ-consistent paths from v to w. For a ϕ-consistent path $P = [v = v_1, v_2, \cdots, w = v_\ell] \in$ Path(v, w), Sig(P) refers to the MinHash signature of P, i.e., Sig$(P) = \cap_{i=1}^{\ell} S_{v_i}$.

Definition 6.2. The *MinHash hierarchy* consists of vertex sets $V_q \subseteq V_{q-1} \cdots, \subseteq V_0 = V$, and a directed graph G_i defined on V_i. The vertex set V_i on level i is a set randomly sampled with probability β from V_{i-1} on the level $i-1$. The edge set E_i contains an edge from v to w if there is at least one ϕ-consistent path from v to w without passing through any other node in V_i. Further, we will store with each edge vw on level i two records, Path(v, w) (the set of all ϕ-consistent paths from v to w) and their signatures.

We remark that a path $P \in$ Path(v, w) can be represented using only the sequence of the level $i-1$ nodes on P, instead of using all the checkpoints on P. If there are multiple ϕ-consistent paths between two nodes of level $i-1$ on P, we can index those ϕ-consistent paths and keep the index of the ϕ-consistent path on P. This method can save storage, while it needs to recursively visit the nodes in the lower levels when reconstructing the ϕ-consistent paths.

Lemma 6.3. *The total number of levels in the MinHash hierarchy is $O(\log m/\log(1/\beta))$, where m is the total number of checkpoints.*

Proof. On level 0, there are m nodes. The chance for each node on level $i-1$ to be selected to level i is β. The expected number of nodes on level i is $m\beta^i$. On the highest level of the hierarchy, the number of nodes is $O(1)$. Therefore, the total number of levels of the hierarchy is $O(\log m/\log(1/\beta))$. □

To construct the MinHash hierarchy, on the lowest level, level 0, we generate a graph $G_0(V_0, E_0)$. V_0 is the same as the set V. Any two checkpoints v, w that are adjacent to each other on a trajectory would evaluate the intersection of their signatures $S_v \cap S_w$. If the Jaccard similarity of path $[v, w]$ is at least ϕ, we add an edge (v, w) to E_0. $[v, w]$ is added to set $\text{Path}(v, w)$ and we update $\text{Sig}([v, w]) = S_v \cap S_w$.

Once the bottom level is constructed, recursively we can construct $G_i(V_i, E_i)$ on level i. Given $G_{i-1}(V_{i-1}, E_{i-1})$, we need to find the edges on E_i. From each $v \in V_i$, we start a depth first search to find all the neighbors of v on this level, i.e., the nodes directly connected to v by a ϕ-consistent path. In particular, for each neighbor w of v in G_{i-1}, we start the depth first search with each $P \in \text{Path}(v, w)$. We denote the current path and the MinHash signatures of the current path as $\tilde{P}$ and $\tilde{S}$; at the beginning of the search, $\tilde{P} = P$, $\tilde{S} = \text{Sig}(P)$. Now, the depth search goes to node w. If $w \in V_i$, the search terminates and we add an edge (v, w) to E_i with $\text{Path}(v, w)$ set as $\tilde{P}$ and $\text{Sig}(\tilde{P}) \leftarrow \tilde{S}$. Otherwise, for each path $P' \in \text{Path}(w, u)$ where $(w, u) \in E_{i-1}$, calculate $\tilde{S} \cap \text{Sig}(P')$. If the Jaccard similarity is greater than ϕ and $\tilde{P} \cap P'$ has no loop, assign the result to $\tilde{P}$, assign $\tilde{S} \cap \text{Sig}(P')$ to $\tilde{S}$ and the depth first search goes to node u. This process will repeat until all neighbors of v on level i are found. Otherwise, this search terminates. An example of the hierarchy structure can be found in Figure 4.

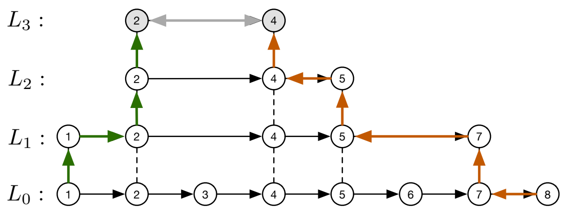


Figure 4: A MinHash hierarchy with 4 levels.

In the following, we will analyze the hierarchy in terms of computational cost and storage cost. First, recall that for each edge vw in G_i we save $\text{Path}(v, w)$, which has at least one ϕ-consistent path P in G_{i-1}. Such paths are represented by a sequence of checkpoints in V_{i-1}. Below we show that there are only a constant number of them.

Lemma 6.4. *The expected number of nodes of level $i-1$ on a ϕ-consistent path $P \in \text{Path}(v, w)$, for $vw \in E_i$, is at most $1/\beta$.*

Proof. Take the path P from v to w. Suppose that there are b nodes of V_{i-1} on P. None of these b nodes are on V_i and v is the first node that appears on V_i. Since each node of V_{i-1} is selected to V_i independently with probability β. The expectation of b is $E(b) \leq \sum_{j=1} j(1-\beta)^{j-1}\beta = 1/\beta$. □

Recursively, the number of checkpoints on the ϕ-consistent path $P \in \text{Path}(v, w)$, for $vw \in E_i$, can be bounded.

Lemma 6.5. *The expected number of checkpoints on P is at most $1/\beta^i$.*

Proof. By the same argument, let α be the number of checkpoints on P. The last checkpoint is selected to level i but not any of the earlier ones. The probability for any checkpoint on P being selected to level i is β^i. Therefore, $E(\alpha) \leq \sum_{j=1} j\beta^i(1-\beta^i)^{j-1} = 1/\beta^i$. □

Lemma 6.6. *For each node $v \in V_i$, the expected number of edges of v in G_i is at most $1/\phi$ to become ϕ-consistent.*

Proof. Recall from Lemma 3.3 that the number of maximal ϕ-consistent path starting from any node v is at most $1/\phi$. This immediately follows from that. □

Theorem 6.7. *For the MinHash hierarchy for ϕ-consistent paths, the expected computation complexity of building the MinHash hierarchy is $O(m/\phi^2)$; the expected communication complexity is $O(m/\phi^2)$; the expected storage complexity is $O(m\log(m)/\phi)$, where m is the number of checkpoints.*

Proof. For any $v \in V_i$, we conduct the depth first search to find its neighbors in G_i. The depth first search terminates at the time either the neighbor of v on V_i is found or the path is not a ϕ-consistent after adding one node. According to Lemma 6.6, there are at most $1/\phi$ ϕ-consistent paths originating from v. Meanwhile, the expected number of nodes to traverse on level $i-1$ for each ϕ-consistent path is at most $1/\beta$ according to Lemma 6.4. Therefore, there are at most $1/(\phi\beta)$ nodes in V_{i-1} on all the ϕ-consistent paths starting from $v \in V_i$. Since every time we traverse to a node, it has at most $1/\phi$ neighbors to explore, the computation complexity of the depth first search for each node $v \in V_i$ is at most $1/(\phi^2\beta)$. In the distributed setting, for each node visited during the depth first search, there is one unit communication cost. The communication complexity of the depth first search for each node v is also at most $1/(\phi^2\beta)$. On level i, the expected number of nodes in V_i is $m\beta^i$. Therefore, the expected computation complexity and communication complexity to construct edges on level i are at most $\beta^{i-1}m/\phi^2$.

With Lemma 6.5, the expected length of a ϕ-consistent path to store is at most $1/\beta^i$. Therefore, the storage cost on level i is m/ϕ.

Now, since the MinHash hierarchy can have at most $\log m$ levels, summing up everything we get that the expected computation complexity and communication cost of building the whole hierarchy is $\sum_{j=1}^{\log m} \beta^{i-1}m/\phi^2 = O(m/\phi^2)$. The expected storage cost is $\sum_{j=1}^{\log m} m/\phi = O(m\log(m)/\phi)$. [1] □

The MinHash hierarchy provides a multi-resolution structure on the ϕ-consistent paths. We can also show that a long ϕ-consistent path is more likely to appear on a high level of the hierarchy.

[1]Remark that for ϕ-consistent paths on level i if we just store the checkpoints on level $i-1$, the storage cost is $O(m/\phi)$.

For a ϕ-consistent path P on the sequence of checkpoints $p_1, p_2, \cdots, p_{\ell+1}$, we define the *length* of P as ℓ, the number of hops of P. For $p_x, p_y \in V_i$, we define that the subpath of P between p_x, p_y, $[p_x, p_{x+1}, \cdots, p_y]$, is *covered* by p_x, p_y, on level i. We are most interested in the maximal subpath covered by all the checkpoints of P on level i. Take the example in Figure 4, on level 0, the length of the maximal subpath covered is 7, on level 1, it is 6, and on level 2, it's 3.

THEOREM 6.8. *For any ϕ-consistent path P of length ℓ, the expected length of the maximal subpath covered on level i is at least $\ell/2$, for $i \leq \log_{1/\beta}(\ell/4)$.*

PROOF. Assume we have j nodes on path P selected to level i. Denote by ℓ_j the length of the maximal subpath covered on level i. If $\ell_j = b$, there are $(\ell + 1 - b)\binom{b-1}{j-2}$ different combinations for the j nodes. Therefore,

$$\Pr\{\ell_j = b\} = \frac{(\ell + 1 - b)\binom{b-1}{j-2}}{\sum_{d=j-1}^{\ell}(\ell + 1 - d)\binom{d-1}{j-2}}.$$

$$E(\ell_j) = \frac{\sum_{d=j-1}^{\ell} d(\ell + 1 - d)\binom{d-1}{j-2}}{\sum_{d=j-1}^{\ell}(\ell + 1 - d)\binom{d-1}{j-2}} = \frac{j-1}{j+1}(\ell + 2).$$

Any checkpoint on P is selected to V_i with probability $q = \beta^i$. The probability that exactly j nodes on P are in V_i is $\binom{\ell+1}{j} q^j (1-q)^{\ell+1-j}$, where $2 \leq j \leq \ell + 1$. Let $\tilde{\ell}$ be the length of the subpath covered on level i with respect to all possible j,

$$\begin{aligned} E(\tilde{\ell}) &= \sum_{j=2}^{\ell+1} \binom{\ell+1}{j} q^j (1-q)^{\ell+1-j} E(\ell_j) \\ &= \ell + 2 - \frac{2}{q} + \frac{(q\ell + 2)(1-q)^{\ell+1} - \ell q^{\ell+2}}{q} \\ &> \ell + 2 - \frac{2}{q} = \ell + 2 - \frac{2}{\beta^i} \end{aligned}$$

Therefore, on the level $i \leq \log_{1/\beta}(\ell/4)$, the expected length of the maximal subpath covered by the nodes on level i is at least $\ell/2$. □

The above theorem means that a ϕ-consistent path of length ℓ has a significant portion of nodes covered on levels below the level $\log_{1/\beta}(\ell/4)$.

7 POPULAR PATH QUERY

Now we explain how to use the MinHash hierarchy for answering ϕ-consistent path queries. We first discuss a number of useful structures and operations.

Each ϕ-consistent path P from v to w has a *canonical representation*, enabled by the MinHash hierarchy. Consider the checkpoints of the highest level (say i) on P. Assume that there are b of them: $\{v_1, v_2, \cdots, v_b\}$. Then we replace the subpath between v_1 and v_b on P by the path $v_1, v_2, \cdots v_b$ on level i. This also partitions the path P to three parts, the first part P_1 from v to v_1, the 'shortcut' from v_1 to v_b on level i, and the last part P_2 from v_b to w. Recursively, we also change P_1 and P_2 to their canonical representations. In other words, the canonical representation contains the first *upward* path in which the checkpoints have increasing levels (including the highest level) and followed by the *downward* path in which the checkpoints have decreasing levels. This representation will be useful for the query algorithms below.

In addition, we also define two basic search operations.

ϕ-consistent upward search. In upward search, we perform a depth-first search from a checkpoint and always search for neighbors of the same level or higher. We start from a checkpoint $v \in V_0$. We perform depth-first search until we find some node u whose level is 1. Beyond u we continue the search on level 1 and repeat the process above until we hit a node of level 2 or we have visited all edges. We continue in the same manner.

During the depth-first search we may also record and calculate the path signature from v to the current node, and trim the search if the Jaccard similarity drops below ϕ. This is called the ϕ-consistent upward search.

ϕ-consistent downward search. In the downward search, we perform a similar depth-first search with the upward search but enforce that the next checkpoint to visit to be of the same level or lower.

LEMMA 7.1. *The number of checkpoints in ϕ-consistent upward search is $O(\frac{1}{\beta\phi}\log(m))$.*

PROOF. The upward search from v will generate a tree rooted at v such that a path from the root to a leaf of this tree is a ϕ-consistent path from v. Each of this path is an *upward path*, containing nodes of at most $\log(m)$ levels. The number of consecutive nodes of the same level on this path is expected to be $1/\beta$ (before we hit a node of a higher level). Since there are only $1/\phi$ ϕ-consistent paths from v, the total number of checkpoints in this tree is $O(\frac{1}{\beta\phi}\log(m))$. □

LEMMA 7.2. *The number of checkpoints in ϕ-consistent downward search is $O(\frac{1}{\beta\phi}\log(m))$.*

PROOF. The downward search also generates a tree in which a path from the root to a leaf node is a ϕ-consistent downward path from v. Similarly, the number of nodes on one such path is at most $O(\frac{1}{\beta}\log(m))$ and there are only $1/\phi$ ϕ-consistent paths from v. Thus the total number of checkpoints in this tree is $O(\frac{1}{\beta\phi}\log(m))$. □

7.1 ϕ-Consistent Path From v To w

Now we answer the query of all the ϕ-consistent paths that start from checkpoint v and end at w.

First we conduct a ϕ-consistent upward search from v. All the nodes visited during this modified depth-first search is denoted as R_v. Meanwhile, we also perform a reversed ϕ-consistent upward search from w, by checking all ϕ-consistent paths that arrive at w. This is almost the same as the above except that we travel in the opposite direction of the edges. The set of nodes discovered is denoted as R_w. An example is illustrated in Figure 4, with green and orange lines demonstrating two searches and the grey denoting the common nodes during the search. If $R_v \cap R_w$ is not empty, for any node u in $R_v \cap R_w$, we concatenate the ϕ-consistent paths from v to u and the ϕ-consistent paths from u to w, in the previous two upward searches and test if the whole path is ϕ-consistent. In particular, we apply the function Path and Sig to each pair of neighbors in the sequence, merge the subpaths and get the intersection of the MinHash signatures.

Theorem 7.3. *The bidirectional depth-first search algorithm can find all the ϕ-consistent paths between two nodes. The number of checkpoints examined is $O(\frac{1}{\beta\phi}\log(m))$.*

Proof. Consider a ϕ-consistent path P from v to w. Denote by u the checkpoint with highest level among all checkpoints on P (u does not have to be unique). Clearly the subpath from v to u and the subpath from u to w are both ϕ-consistent. Therefore u will surely show up in both depth-first search results. Thus all ϕ-consistent paths from v to w will be discovered by the algorithm. The total number of checkpoints visited immediately follows after Lemma 7.1. □

7.2 All ϕ-Consistent Paths From v

To find all ϕ-consistent paths starting from v, we first perform a ϕ-consistent upward search, which gives a tree T_v. For each node u on T_v, we perform a ϕ-consistent downward search. The final outcome gives all the ϕ-consistent paths from v.

Theorem 7.4. *The above algorithm can find all the ϕ-con-sistent paths from v. The number of checkpoints examined is $O(\frac{1}{\beta^2\phi^2}\log^2(m))$.*

Proof. Every ϕ-consistent path from v has a canonical representation. The upward path is discovered by the upward search from v, while the downward path can be discovered from the downward search. Thus all such paths are found.

The total number of checkpoints visited immediately follows after Lemma 7.1 and Lemma 7.2. □

7.3 Jaccard Similarity of a Path

To find the Jaccard similarity of a path P, we can get the canonical representation of the path. We communicate with the first checkpoint on P, do an upward search. With the path given, the checkpoint can find its neighbor on its highest level, communicate with that neighbor, and repeat this process recursively from the neighbor until one checkpoint has no neighbor on its highest level belonging to the path. Thereafter, from that checkpoint we do a downward search for the neighbor, 1-level lower, on the given path and repeat this process until we reach the last checkpoint on the path. We calculate the intersection of the MinHash signatures of the subpaths while finding the canonical representation. The communication complexity of finding the canonical represent and calculating the Jaccard similarity is $O(\frac{1}{\beta}\log(m))$. Remark if the Jaccard similarity ϕ is 0, we can terminate the search and return 0.

8 EXPERIMENTS

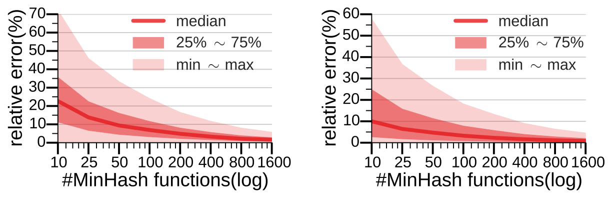
In this section, we evaluate our MinHash hierarchy and algorithms on the taxi dataset of Shenzhen. First, we analyse the accuracy of MinHash estimation of Jaccard similarity and set cardinality. We compare the complexity of constructing MinHash hierarchy with *SPADE* [51] and *PrefixSpan* [24]. Moreover, we demonstrate the efficiency to answer ϕ-consistent path queries with MinHash hierarchy compared with the brute-force method In the end, we provide evaluations for privacy preserving properties.

8.1 Data Description

The data used contains taxis' GPS locations collected every 1.01 minutes on average in Shenzhen. We capture a two-hour duration with 29,639 trajectories, choose the checkpoints along the roadmap, then interpolate the sampled GPS locations on the roadmap, and represent the trajectories as the sequences of checkpoints passing by. The average number of checkpoints each taxi visits is 123.91, with the minimum and maximum number of 31 and 1,580, respectively.

8.2 MinHash Performance

We first evaluate the accuracy of the estimation of set cardinality and Jaccard similarity. For each checkpoint, we calculate the MinHash estimation of both the ID set cardinality of mobile entities passing by and the Jaccard similarity between its ID set and their neighbors', using different numbers of hash functions k. The relative errors of the estimations are illustrated in Figure 5(a) and Figure 5(b), where the relative error is defined as the difference between the exact value and the estimation divided by the exact value. We observe that the larger k is, the smaller the error is. In this work, we choose $k = 200$ as a reasonable value that balances the accuracy, the storage complexity, and privacy. When $k = 200$, the median relative errors of set cardinality and Jaccard similarity are 4.57% and 2.55%, respectively.



(a) Cardinality. (b) Jaccard similarity coefficient.

Figure 5: Relative error of MinHash estimation.

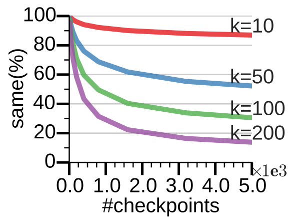
8.3 Privacy Protection

Here we demonstrate the privacy protection ability of our MinHash signature with the numbers of hash functions, mobile entities and checkpoints. With the different combinations of the three factors, we choose a trajectory set and get the MinHash signature of the set, then we add more trajectories into the trajectory set to find out the probability that the MinHash signature of the new trajectory set remains the same, in the setting of ε-differential privacy.

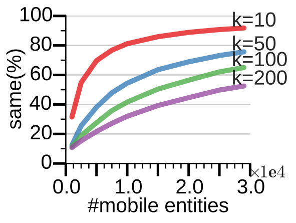
Figure 6 demonstrates the results. The Y-axis is the probability that the MinHash signature of the set remains the same after adding one extra mobile entity with the change of #hash functions k, #checkpoints m, and #mobile entities n. In Figure 6(a), we observe that the probability goes down with the increase of k and m. Figure 6(b) demonstrates the probability goes up with the increase of n. The results follow Theorem 5.1.

8.4 MinHash Hierarchy

We assign each taxi a unique ID and use $k = 200$ hash functions to generate the signature for each taxi. Each checkpoint is assigned with the levels it belongs to and the neighbors on the lowest level.



(a) With different k, m. $n = 29,639$.



(b) With different k, n. $m = 500$.

Figure 6: Probability of signatures being the same.

Table 1: Hierarchy structures details

Level	Checkpoints	Edges	Avg. subpath length	Max subpath length
0	7,714	11,713	1	1
1	3,874	7,651	2.40	13
2	1,949	4,406	4.40	24
3	961	1,999	6.45	32
4	434	731	8.25	40
5	171	188	9.82	42

All checkpoints update their MinHash signatures when the taxis pass by. With the MinHash signatures, all the checkpoints on different levels communicate with their neighbors to build the MinHash hierarchy.

Structure. We build the 6-level hierarchy for the ϕ-consistent paths recursively where $\phi = 5\%$, the chance of each checkpoint to be chosen to a higher level $\beta = 1/2$, as shown in Figure 7. The details of the hierarchy structure are listed in Table 1. On each level, we just count the checkpoints that have at least one edge.

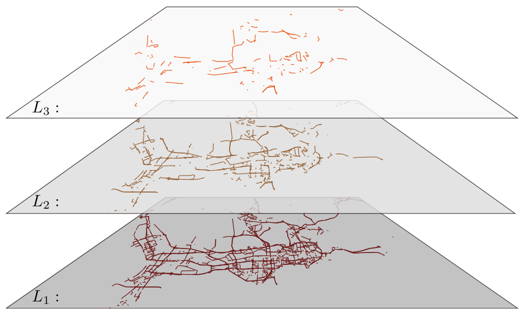


Figure 7: L_1, L_2, and L_3 of the MinHash hierarchy structure for ϕ-consistent path.

We also build a 6-level MinHash hierarchy for ψ-popular path with $\psi = 50$, $\beta = 1/2$, with similar statistics as above. We use the same values for ψ and ϕ in the rest of experiments. Here, we examine the accuracy of the MinHash hierarchy. For each path kept in the hierarchy, we evaluate whether the path is a ϕ-consistent path/ψ-popular path with the real mobile entity ID sets of the checkpoints on the path. If the answer is no, we count it as a false positive. Meanwhile, we find all the real ϕ-consistent paths/ψ-popular paths between checkpoints on each level, if a path is not included in the hierarchy, we count it as a false negative. The statistics of the numbers of total paths, false positive paths and false negative paths are shown in Table 2. The percentiles in the parentheses are the values divided by the number of total paths. With $k = 200$, we can achieve enough accuracy for both hierarchies. The error rate for ψ-popular paths is approximately twice the one for ϕ-consistent paths, because the estimation of ψ-popular paths depends on both the estimation of set cardinality and Jaccard similarity.

Table 2: Accuracy of MinHash hierarchy

Algorithm	#Total paths	#False positive	#False negative
ϕ-consistent path	28,316	970 (3.45%)	600 (2.12%)
ψ-popular path	24,839	1,554 (6.25%)	1,222 (4.92%)

Table 3: Hierarchy structures vs SPADE

Algorithm	Computation complexity	Storage for paths	Storage for signatures
SPADE	79,838,400	4,382,387	73,432,800
Hierarchy	8,688,200 (10.8%)	63,980 (1.46%)	5,663,200 (7.71%)

Complexity. We first compare our MinHash Hierarchy with SPADE [51], which takes the same data format to find sequential patterns as in our algorithm, where trajectory data are aggregated by checkpoints. The algorithm is Apriori-based with the "bottom up" feature in finding the frequent patterns. It can be used to find both ϕ-consistent paths and ψ-popular paths. Here we only demonstrate the results of ϕ-consistent paths. The SPADE algorithm with MinHash is implemented as following. Denote ℓ-path as the path with length ℓ. First, we calculate all the ϕ-consistent 1-paths between neighboring checkpoints on the lowest level. Remark that in the original SPADE, the ϕ-consistent path is obtained by the set operations of the ID set of mobile entities passing by, while in this experiment, we use MinHash to replace the set operations in order to focus on the differences of the algorithms. After obtaining the ϕ-consistent $l-1$-paths, we recursively generate the set of ϕ-consistent ℓ-paths. For a path $P = [v_1, v_2, \cdots v_\ell]$, if the subpath from v_1 to $v_{\ell-1}$ and the subpath from v_2 to v_ℓ are both ϕ-consistent, we calculate the intersection of the MinHash signatures of the two subpaths. If the result is ϕ-consistent, we add P to the set of ϕ-consistent ℓ-paths. The process is repeated until no new consistent paths can be found.

We compare the complexity of building the MinHash hierarchy and the SPADE algorithm. The computation cost in our experiment is the cost of comparisons. Each comparison of two MinHash values has a unit cost. The storage cost is the space used to store all the values. The space to store a MinHash value or a checkpoint ID is one unit. In our analysis, each checkpoint MinHash signature has 200 MinHash values. The computation cost of two checkpoint MinHash signatures is 200, and the storage cost of one checkpoint MinHash signature is 200. The complexity analysis is demonstrated in Table 3. The percentiles in the parentheses are the values of MinHash hierarchy divided by the ones of SPADE. Since SPADE just grows one node for each step, there are a lot of overlapping paths to compute and store. Compared with SPADE, our hierarchy is more efficient in both computation and storage.

Table 4: Hierarchy structures vs PrefixSpan

Algorithm	Computation complexity	Storage complexity
PrefixSpan	47,063,962	187,112,931
Hierarchy	7,369,600 (15.7%)	5,022,280 (2.68%)

We also compare our hierarchy with PrefixSpan [24], one of the most efficient algorithms for sequential pattern mining. Different from SPADE, PrefixSpan take the whole trajectories of all the taxis as input and it can only be used to find the ψ-popular path. Hence, PrefixSpan is compared with our hierarchy of ψ-popular paths over the total computation and storage cost. Remark that to adapt to our problem, we restrict PrefixSpan to detect only contiguous frequent sequential patterns. The comparison results are in Table 4. We observe that it takes PrefixSpan a lot of spaces and computations finding frequent patterns in long sequences, such as trajectories. Our hierarchy outperforms PrefixSpan.

8.5 Query

We compare the communication cost of the queries for ϕ-consistent paths using MinHash hierarchy with the *brute force algorithm*. The brute force algorithm answers queries with a breadth first search: once visiting a checkpoint, the checkpoint communicates with all the neighbors to see if appending the neighbor to the current path still results in a ϕ-consistent path; if the answer is yes, then add the neighbor to the current path and update the MinHash signature accordingly; the neighbor starts the same process. The above process is repeated until the ϕ-consistent path cannot be extended any more.

ϕ-consistent paths from v to w. For the query of ϕ-consistent paths between two checkpoints, we run the bidirectional depth first search(BDFS) on the MinHash hierarchy and compare with brute force algorithm. We submit $17,938$ queries. In Figure 8(a), the red points illustrate the communication costs of our algorithm corresponding to the ones of brute force for each query. When the communication costs of brute force increase from 100 to 250, the costs of BDDFS remains between 20 to 30. In Figure 8(b), we show the average communication costs of brute force and BDFS corresponding to different lengths of the ϕ-consistent paths between two checkpoints. The communication costs of brute force increase with the increase of the length of ϕ-consistent path, while BDFS almost remains the same. We can see in both figures, our algorithm scales well and outperforms the brute force.

All ϕ-consistent paths from v. We submit ϕ-consistent path queries from all the $7,701$ checkpoints. In Figure 8(a), the blue line shows the average communication costs corresponding to the ones of brute force search. Finding all ϕ-consistent paths from v with our algorithm needs at most 2 times the communication cost of finding ϕ-consistent path between v and w, meanwhile it scales well with large communication costs of brute force.

Jaccard similarity of a path. To calculate the Jaccard similarity of a given path with length ℓ, naively, the checkpoints can communicate in the sequence of the path with communication cost of $O(\ell)$. While with MinHash hierarchy, we just need communicate to obtain the canonical representation. Remark that for our analysis, we don't terminate the search, when the Jaccard similarity diminishes to 0, to give an upper bound for the communication cost of our algorithm. As illustrated in Figure 8(c), the average communication complexity of a path with length ℓ is only $O(\log(\ell))$.

9 CONCLUSION

In this paper, we proposed a distributed sensing framework by using the MinHash hierarchy for efficient real-time query of popular paths among distributed checkpoints. The proposed method navigates through a variety of design objectives including low data collection cost and highly efficient queries as well as preserving cumulative group behavior while protecting individual user privacy. We believe that the proposed scheme stands at a unique position at the interface of collecting, managing, and analyzing real-time, large scale human motion traces.

ACKNOWLEDGMENTS

J. Ding, C.-C. Ni and J. Gao would like to acknowledge support through NSF DMS-1418255, CCF-1535900, CNS-1618391 and AFOSR FA9550-14-1-0193.

REFERENCES

[1] O. Abul, F. Bonchi, and M. Nanni. 2008. Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on.* 376–385.

[2] R. Agrawal and R. Srikant. 1995. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on.* 3–14.

[3] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential PAttern Mining Using a Bitmap Representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02).* 429–435.

[4] A.R. Beresford and F. Stajano. 2003. Location privacy in pervasive computing. *Pervasive Computing, IEEE* 2, 1 (Jan 2003), 46–55.

[5] Kevin Beyer, Peter J Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On synopses for distinct-value estimation under multiset operations. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data.* ACM, 199–210.

[6] Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings.* IEEE, 21–29.

[7] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. 1998. Min-wise independent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing.* ACM, 327–336.

[8] Hong Cheng, Xifeng Yan, and Jiawei Han. 2004. IncSpan: Incremental Mining of Sequential Patterns in Large Database. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04).* 527–532.

[9] Chi-Yin Chow and Mohamed F. Mokbel. 2011. Trajectory Privacy in Location-based Services and Data Publication. *SIGKDD Explor. Newsl.* 13, 1 (Aug. 2011), 19–29.

[10] Edith Cohen. 1997. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.* 55, 3 (1997), 441–453.

[11] Edith Cohen. 2008. *Min-Hash Sketches.* Springer US, 1–7.

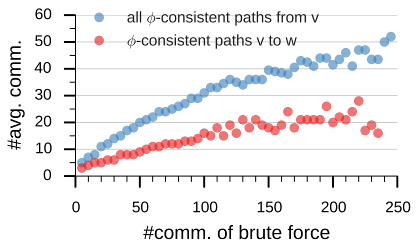
[12] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. 2013. Unique in the Crowd: The privacy bounds of human mobility. *Scientific Reports* 3 (25 March 2013).

[13] Cynthia Dwork. 2006. Differential privacy. In *Automata, languages and programming.* Springer, 1–12.

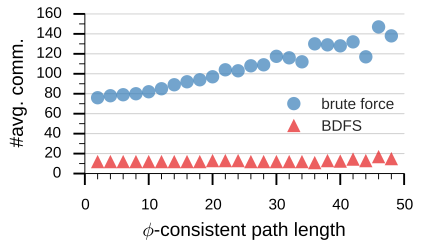
[14] Philippe Flajolet and G Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31, 2 (1985), 182–209.

[15] Julien Freudiger, Maxim Raya, Márk Félegyházi, Panos Papadimitratos, and Jean-Pierre Hubaux. 2007. Mix-zones for location privacy in vehicular networks. In *ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS).*

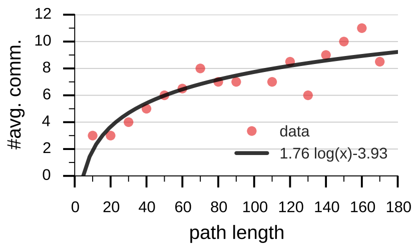
[16] Sheng Gao, Jianfeng Ma, Cong Sun, and Xinghua Li. 2014. Balancing Trajectory Privacy and Data Utility Using a Personalized Anonymization Model. *J. Netw. Comput. Appl.* 38 (Feb. 2014), 125–134.



(a) The average communication costs of bidirectional depth first search and the corresponding cost of the brute force over two queries.



(b) The average communication costs of the brute force and the bidirectional depth first search(BDFS) with respect to the same popular path length.



(c) The average communication costs of the queries of popularity of given paths with MinHash hierarchy illustrate logarithm of the length of the popular paths. $R^2 = 0.79$.

Figure 8: Communication costs.

[17] Fosca Giannotti, Mirco Nanni, and Dino Pedreschi. 2006. Efficient mining of temporally annotated sequences. In *In Proc. SDM'06.* 346–357.

[18] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. Trajectory pattern mining. *KDD* (2007), 330–339.

[19] Marta C. González, César A. Hidalgo, and Albert-Laśzló Barabási. 2008. Understanding Individual Human Mobility Patterns. *Nature* 453 (June 2008).

[20] Olof Görnerup, Nima Dokoohaki, and Andrea Hess. 2015. Privacy-preserving mining of frequent routes in cellular network data. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, Vol. 1. IEEE, 581–587.

[21] Marco Gruteser and Dirk Grunwald. 2003. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03).* ACM, 31–42.

[22] Joachim Gudmundsson and Marc van Kreveld. 2006. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems.* ACM, 35–42.

[23] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2000. FreeSpan: Frequent Pattern-projected Sequential Pattern Mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00).* ACM, 355–359.

[24] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and MC Hsu. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering.* 215–224.

[25] 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.

[26] 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 (2008), 1068–1080.

[27] Ryota Jinno, Kazuhiro Seki, and Kuniaki Uehara. 2012. Parallel distributed trajectory pattern mining using MapReduce. (2012), 269–273.

[28] 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.

[29] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. 2005. On discovering moving clusters in spatio-temporal data. In *International Symposium on Spatial and Temporal Databases.* Springer, 364–381.

[30] H. Kido, Y. Yanagisawa, and T. Satoh. 2005. An anonymous communication technique using dummies for location-based services. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on.* 88–97.

[31] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hong Cheng. 2011. Mining Discriminative Patterns for Classifying Trajectories on Road Networks. *Knowledge and Data Engineering, IEEE Transactions on* 23, 5 (May 2011), 713–726.

[32] Yunhao Liu, Yiyang Zhao, Lei Chen, Jian Pei, and Jinsong Han. 2012. Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. *IEEE Transactions on Parallel and Distributed Systems* 23, 11 (Oct. 2012), 2138–2149.

[33] E.H.-C. Lu, V.S. Tseng, and P.S. Yu. 2011. Mining Cluster-Based Temporal Mobile Sequential Patterns in Location-Based Service Environments. *Knowledge and Data Engineering, IEEE Transactions on* 23, 6 (June 2011), 914–927.

[34] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. 2006. The New Casper: Query Processing for Location Services Without Compromising Privacy. In *Proceedings of the 32Nd International Conference on Very Large Data Bases (VLDB '06).* VLDB Endowment, 763–774.

[35] Mehmet Ercan Nergiz, Maurizio Atzori, Yücel Saygin, and Baris Güç. 2009. Towards Trajectory Anonymization: A Generalization-Based Approach. *Trans. Data Privacy* 2, 1 (April 2009), 47–75.

[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] C. Paar and J. Pelzl. 2010. *Understanding Cryptography: A Textbook for Students and Practitioners.* Springer-Verlag New York Inc.

[38] K.G. Shin, Xiaoen Ju, Zhigang Chen, and Xin Hu. 2012. Privacy protection for users of location-based services. *Wireless Communications, IEEE* 19, 1 (February 2012), 30–39.

[39] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of Predictability in Human Mobility. *Science* 327, 5968 (2010), 1018–1021.

[40] Yi Song, Daniel Dahlmeier, and Stephane Bressan. 2014. Not So Unique in the Crowd: a Simple and Effective Algorithm for Anonymizing Location Data.. In *PIR@ SIGIR.* Citeseer, 19–24.

[41] 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.

[42] Latanya Sweeney. 2002. K-anonymity: A Model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 5 (Oct. 2002), 557–570.

[43] Florian Verhein. 2009. Mining Complex Spatio-Temporal Sequence Patterns. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[44] 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.

[45] Mengwen Xu, Dong Wang, and Jian Li. 2016. DESTPRE: a data-driven approach to destination prediction for taxi rides. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ACM, 729–739.

[46] Toby Xu and Ying Cai. 2007. Location anonymity in continuous location-based services. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems.* ACM, 39.

[47] Toby Xu and Ying Cai. 2008. Exploring historical location data for anonymity preservation in location-based services. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 547–555.

[48] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. 2008. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on.* 366–375.

[49] 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.

[50] Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. 2007. Protecting Moving Trajectories with Dummies. In *Proceedings of the 2007 International Conference on Mobile Data Management (MDM '07).* IEEE Computer Society, Washington, DC, USA, 278–282.

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

[52] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proceedings of the 18th International Conference on World Wide Web (WWW '09).* 791–800.