IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2020, accepted October 22, 2020, date of publication October 27, 2020,
date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033987

Behavior Sequence Mining Model Based on Local Differential Privacy

JIANEN YAN, YAN WANG, AND WENLING LI
School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Corresponding author: Jianen Yan (yanjianen@hit.edu.cn)

This work was supported in part by the National Science and Technology Major Project under Grant 2017YFB0803001, and in part by the China Education and Research Network (CERNET) Innovation Project under Grant NGLL20170412.

ABSTRACT Most of local differential privacy frameworks target statistics on certain privacy behaviors of users, but not behavior sequence. In this paper, we explore and propose a behavior sequence mining model that satisfies the local differential privacy requirement to settle the matter. We decompose their potential behavior sequence into multiple temporal pairs that are computed by the server to infer indirectly behavior sequence of users, shrinking the statistical sample space with adjacent temporal pairs to reduce statistical errors. The experiment takes an example, trajectories of users can be inferred by their location information, to demonstrate the effect our model achieved. It shows that the model can approximate users' trajectories under the requirement of local differential privacy.

INDEX TERMS Behavioral sequence, local differential privacy, privacy protection, user trajectory.

I. INTRODUCTION

The data age has spawned a variety of data collecting technologies that require different data sources, including personal privacy data. Thus, how to ensure that personal privacy data is not infringed when collect it has gradually become the focus of information security. The problem that traditional differential privacy models require a trusted third party to centrally deal with individual privacy data is solved by Local Differential Privacy (LDP), and its privacy protection is theoretically more rigorous than the one offered by non-local different privacy. Meanwhile, LDP also facilitates differential privacy to be applied gradually in practice.

However, existing LDP models can only acquire event element frequency of collected users, but not behavior sequences. For instance, the flow of people can be judged according to temporal relationships of their position information in a certain area; individual browsing routes can be funded according to the temporal relationships of the time to visit these websites; buzzwords can be inferred according to temporal relationships among popular words. Therefore, an algorithm model that satisfies LDP privacy is designed in the paper, decomposing users' behavior sequences into multiple adjacent temporal pairs represented by a matrix. The Server-side could conclude behavior sequences of users in accordance with the privacy of LDP.

The associate editor coordinating the review of this manuscript and approving it for publication was Ilsun You.

II. RELATED WORK

Differential Privacy [1], [2] is the latest privacy protection framework proposed by Dwork for the privacy leakage of a statistical database. It is the strictest privacy protection framework without consideration of the attacker's background knowledge. At the time of Dwork proposed the differential privacy, he also proposed a basic method to implement differential privacy, namely the Laplace mechanism[2]. After that, McSherry and Talwar[3] proposed an Exponential mechanism differential privacy framework for non-numeric data. Releasing data in the form of histograms[4] is also a hot topic in the field of differential privacy due to its tiny global sensitivity. For the data mining field, there are also some mining algorithms based on differential privacy, Bhaskar *et al.* [5] presented two efficient algorithms for discovering the k most frequent patterns in a data set of sensitive records, Li *et al.* [6] leveraged a novel notion called basis sets. A θ-basis set has the property that any item set with a frequency higher than θ is a subset of some basis.

However, the premise of the differential privacy theory framework is a trusted third-party center, making the differential privacy theory framework unable to be applied on a large scale. For this reason, emerges the Local Differential

Privacy (LDP)[7]. LDP does not require the participation of the trusted third-party, even assumes that the third-party will perform a malicious attack. In LDP, the third-party data processing center can only collect the noisy data for mining valuable information. LDP applies the Laplace mechanism, Exponential mechanism, and Random response[8] to achieve the requirement as traditional differential privacy does. Since then, Dwork[9] studied how to make the collection of streaming data based on LDP; To facilitate data mining, in [10] applied likelihood estimation and EM algorithm to try to restore the probability distribution of original data on the noisy data processed by Laplace mechanism. Xiao and Xiong [11] took the user's temporal series trajectory into account when collecting their location information, and implement local differential privacy protection under this condition. First applied RAPPOR[12] technology based on random response to Google Chrome in 2014. To solve the problems in which existing trajectory privacy-preserving models have poor data availability or difficulty to resist complex privacy attacks, Xiao and Xiong [11] devise novel trajectory privacy-preserving method based on clustering using differential privacy. It is difficult to obtain available statistics, and the current solutions to privacy protection are inefficient, costly. To solve these problems, Xiao and Xiong [11] propose a local differential privacy sensitive data collection protocol in human-centered computing. A service provider might require aggregate data from end-users which often contain sensitive information to perform mining tasks. Xiao and Xiong [11] propose the first locally differentially private K-means mechanism under this distributed scenario. Google engineer Erlingsson *et al.* A series of algorithms such as CMS proposed by Apple in 2016, is the first time that differential privacy is applied to its devices on a large scale.

Differential privacy applies to a wide range of data types, such as numerical data [2], non-numerical data [3], histogram [4], etc. Local differential privacy can better protect users' privacy compared with centralized differential privacy, which has also attracted more scholars' attention. Local differential privacy is more suitable for actual production environment, and the relevant applications include stream data collection [9], the protection of users' location information [11], [13], the protection of user preference setting information [12], the protection of user device usage record information [14], [15], etc. However, Most of local differential privacy frameworks target statistics on certain privacy behaviors of users, but not behavior sequences, this is the issue to be solved in this article.

III. PRELIMINARY KNOWLEDGE

A. DIFFERENTIAL PRIVACY

Definition 1 [1]: A randomized function Q achieve ε-differential privacy if for all data sets D and D′ differing in at most one element, and S $\subseteq$ Range(Q) for all the outcome of Q:

$$\Pr\left[\mathrm{Q}\left(\mathrm{D}\right) \in \mathrm{S}\right] \leq \exp(\varepsilon) \times \Pr[\mathrm{Q}\left(\mathrm{D}'\right) \in \mathrm{S}] \tag{1}$$

And ε is known as the privacy budget. Assuming that one of the data sets is a regular release version while the other is the attacker's background knowledge, the hypothetical attack scene to differential privacy model is that the attacker has mastered all information except the target client. The protected of the target client— differential privacy requires that even in this worst case, the attacker cannot determine the specific information of the target client, that is, the result of the operation on the possible different data sets should be ''similar'' as far as possible, the degree of ''similarity'' is reflected in ε. It can be seen that the more similar operation results with different data sets (the smaller the ε), the better privacy protection effect achieve, also means the more loss of information. The ranges of ε are usually from 0.1 to 10.

1) SEQUENTIAL COMPOSITION

Suppose differential privacy algorithms $Q_1, Q_2, \ldots, Q_n$ acting on isolated data sets $D_1, D_2, \ldots, D_n$ with $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ respectively. The final privacy guarantee is said to be $\left(\sum_{i=1}^{n} \varepsilon_i\right)$-differential privacy.

2) PARALLEL COMPOSITION

Suppose differential privacy algorithms $Q_1, Q_2, \ldots, Q_n$ acting on non-isolated data sets $D_1, D_2, \ldots, D_n$ with $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ respectively. The final privacy guarantee is said to be $(\max \varepsilon_i)$-differential privacy.

Although differential privacy protection framework can be rigorously proven, the premise is a trusted third-party that collects and processes user's private information. However, users are unwilling to disclose the private information to the third-party in some cases, while the third-party is unwilling to obtain row data to increase the cost of data maintenance or the risk of privacy disclosure either. For this reason, emerges the concept of local differential privacy. Local differential privacy ensures that the row data has been added to the noise before uploading, without the assumption of a trusted third-party data processing center. Meanwhile, it also should ensure that the third-party can accurately infer user's statistic information according to the noisy data. For simplicity, the third-party data processing center is referred to as the server-side and the individual user as the client. As shown in Fig.1 the most obvious difference between the traditional differential privacy and the local differential privacy is the time of adding noise. In the traditional differential privacy model, the row data is uploaded to the server directly, then the server-side add noise to it. In the local differential privacy model, the user uploads the noisy data to the server after adding noise locally.

IV. COLLECTION OF BEHAVIOR SEQUENCE

The behavior sequence in this paper refers to a series of sequential event element collections that occur on the client-side. Generally, the data collection process which meets the requirement of local differential privacy framework can only ensure the server-side obtain statistical frequency information of an event element (such as the population density of a

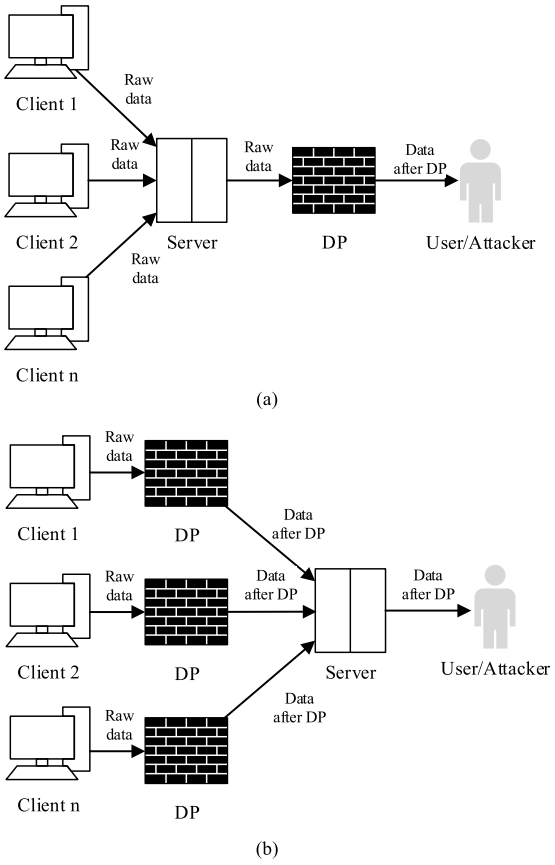


FIGURE 1. The comparison of the traditional differential privacy model and the local differential privacy model. In the figure, (a) shows the traditional differential privacy model while (b) is the local differential privacy model.

certain place in a certain period, the frequency of access to the destination website, hot words, etc.), does not reflect the temporal relationship between event elements, which usually implies the user's behavior sequence (such as trajectories, website browsing routes, hotspot statements, etc.). For this reason, the article decomposes the user's behavior sequence into a set of temporal pairs and infers the user's behavior sequence indirectly by a statistic of temporal pairs.

A. PROBLEM MODEL

Let say U be the user set, $\{u_1, u_2, \ldots, u_n\}$ be the users participating in the data collection; E is the target event consisting of elements $\{e_1, e_2, \ldots, e_m\}$; M is an arbitrary collection of permutations and combinations of event elements; $d^i \in M$ is the private data generated by the user u_i on his/her client for the event E with sequential temporal relationship; $\hat{d}^i$ is the noise processing of the d^i at the client to meet the local differential privacy required desensitized data; W is a set of potential behavior sequences for all users (such as human trajectories, browsing routes, popular sentences), $W = \{m_1, m_2, \ldots, m_k\}, m_x \in M$; $\hat{W}$ is behavior sequence excavated by the server according to the collected noisy data set $\left\{\hat{d}^1, \hat{d}^2, \ldots, \hat{d}^n\right\}$, now, it is required to ensure that $\hat{W} \approx W$ as much as possible.

B. MODEL SIMPLIFICATION

Considering the strict requirement of differential privacy, the amount of noise added to raw data is linear with the amount of data collected from a single user, which results in limited information collection. Therefore, the potential behavior sequence set W in this paper is decomposed into the form of a temporal pair set, $m_i = \left(\left(e^{1th}, e^{2th}\right), \left(e^{2th}, e^{3th}\right) \ldots\right)$. Different behavior sequences may resolve the same temporal pair $\left(e^{1th}, e^{2th}\right)$ after decomposing. Now, the problem is simplified to the collection of temporal series pair $\left(e^{1th}, e^{2th}\right)$ regardless of the relationship between the decomposed m_i temporal pairs (which in some cases causes a large loss of information). The following shows how to collect user temporal pairs with the requirement of differential privacy.

V. TEMPORAL PAIR COLLECTION BASED ON LOCAL DIFFERENTIAL PRIVACY

This chapter first introduces the concept of temporal matric, and then shows how to implement the collection of the temporal pairs of event elements generated by the clients under the requirement of the local differential privacy framework. The main method is to apply a matrix to represent the temporal relationship between the event elements.

A. TEMPORAL MATRIX

For any event element temporal pair (e_i, e_j) in event E, applying an m × m matrix T(m is the number of elements in event E) which element at T_{ij} is 1 and the rest is 0 represents a temporal pair (e_i, e_j). The matrix of such interpretation is referred to below as the temporal matrix T. As shown below.

$$\left|\begin{array}{l} T_{11}=0\ T_{12}=0\ \ldots T_{1j}=0\ \ldots\ T_{1m}=0 \\ T_{21}=0\ T_{22}=0\ \ldots\ T_{2j}=0\ \ldots\ T_{2m}=0 \\ \ldots \\ T_{i1}=0\ T_{i2}=0\ \ldots\ T_{ij}=1\ \ldots\ T_{im}=0 \\ \ldots \\ T_{m1}=0\ T_{m2}=0\ \ldots\ T_{mj}=0\ \ldots\ T_{mm}=0 \end{array}\right| \leftrightarrow (e_i, e_j)$$

1) LOCAL DIFFERENTIAL PRIVACY MODEL

The user behavior sequence collection model consists of two algorithms—the client-side algorithm and the server-side algorithm. The client-side algorithm runs locally on each collected user, responsible for performing noise processing on the original private data generated by the user to meet the differential privacy requirement with a certain budget, and then submitting the noisy data to the server. The server applies the server-side algorithm to transform and parse the collected data for valuable information after collecting noisy data from clients. Since the server collects only the noisy data, it cannot cause any unexpected privacy leak no matter what kind of mining it does.

2) CLIENT-SIDE ALGORITHM A_{client}

Assuming that the event elements generated by the user are recorded locally on the client, the client algorithm

$\boldsymbol{A}_{client}$ first randomly selects a sequential pair within the valid range from the local record set and converts them into a form of a temporal matrix. Then, inspired by the idea of random response, each bit in the matrix is flipped independently with probability P (0 becomes 1, 1 becomes 0), and the temporal matrix after flipping satisfies $2ln\left(\frac{1-P}{P}\right) - LDP$.

Proof: Let T, T' be the temporal pair selected from the client and the corresponding temporal matrix, $\hat{T}$ is the flip matrix. Since the $\boldsymbol{A}_{client}$ the algorithm runs the client, to prove that the $\boldsymbol{A}_{client}$ satisfies $2ln\left(\frac{1-P}{P}\right) - LDP$, it only needs to prove that for any two raw temporal matrices T, T' and any flipped matrix $\hat{T}$, satisfied (2).

$$\left| \ln \left(\frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T, P\right) = \hat{T}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T', P\right) = \hat{T}\right)} \right) \right| \leq 2 \ln \left(\frac{1-P}{P} \right) \tag{2}$$

The client algorithm $\boldsymbol{A}_{client}$ is independent of each element of the original temporal matrix, so it is possible to apply the algorithm $\boldsymbol{A}_{client}$ to each matrix element

$$\frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T, P\right) = \hat{T}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T', P\right) = \hat{T}\right)} = \prod_{i,j=0}^{m} \frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{ij}, P\right) = \hat{T}_{ij}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T'_{ij}, P\right) = \hat{T}_{ij}\right)} \tag{3}$$

Since a temporal matrix is only applied to represent an event element temporal pair (only one bit is set to 1), Let two original temporal matrix be T, T', and the elements they set are $T_{rc}, T'_{r'c'}$. If the positions of the two matrix set elements are the same ($r = r', c = c'$), then $T = T'$, so

$$\prod_{i,j=0}^{m} \frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{ij}, P\right) = \hat{T}_{ij}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T'_{ij}, P\right) = \hat{T}_{ij}\right)} = 1 \tag{4}$$

If the positions of the two matrix set elements are different, the elements at the position other than $T_{rc}, T'_{r'c'}$ are the same (0). So for other arbitrary positions T_{ij}, there is $\frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{ij},P\right)=\hat{T}_{ij}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T'_{ij},P\right)=\hat{T}_{ij}\right)} = 1$, and

$$\begin{aligned} &\prod_{i,j=0}^{m} \frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{ij}, P\right) = \hat{T}_{ij}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T'_{ij}, P\right) = \hat{T}_{ij}\right)} \\ &= \frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{rc}, P\right) = \hat{T}_{rc}\right) \boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{r'c'}, P\right) = \hat{T}_{r'c'}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T'_{rc}, P\right) = \hat{T}_{rc}\right) \boldsymbol{Pr}\left(\boldsymbol{A}_{client}\left(T_{r'c'}, P\right) = \hat{T}_{r'c'}\right)} \end{aligned} \tag{5}$$

Algorithm 1 $\boldsymbol{A}_{client}$

// $\boldsymbol{A}_{client}$ $(E\,(m)\,, P, Th_t)$
// Input: $E(m)$, P, Th_t
// Output: $\hat{T}$
1: randomly select $d^{1th} \in E(m)$
2: record $t_{d^{1th}}$ of d^{1th}
3: randomly select $d^{2th} \in E\,(m)$ and $t_{d^{2th}} \in [t_{d^{1th}}, t_{d^{1th}+}Th_t]$
4: T : T $\leftarrow \left(d^{1th}, d^{2th}\right)$
5: ***for*** $i \in [m]$ ***do***
6: ***for*** $j \in [m]$ ***do***
7: $\widehat{T_{ij}} \leftarrow 2 - 2^{T_{ij}}$ *with probability P*
8: ***end for***
9: ***end for***
10: return $\hat{T}$;

Now we discuss the possible distribution of $\hat{T}_{rc}, \hat{T}_{r'c'}$, first, assume that $\hat{T}_{rc} = 0, \hat{T}_{r'c'} = 0$, then (3) is equal to (6), as shown at the bottom of the page.

Similarly, the above result is also 1 when $\hat{T}_{rc} = 1$, $\hat{T}_{r'c'} = 1$. Now consider the case of $\hat{T}_{rc} \neq \hat{T}_{r'c'}$, if $\hat{T}_{rc} = 0$, $\hat{T}_{r'c'} = 1$, then (3), (7), as shown at the bottom of the page.

Similarly, (3) $= \frac{(1-P)^2}{P^2}$ when $\hat{\mathrm{T}}_{rc} = 1, \hat{\mathrm{T}}_{r'c'} = 0$. In summary, we can draw the following conclusions since P ≤ 0.5.

$$\left| \ln \left(\frac{\boldsymbol{Pr}\left(\boldsymbol{A}_{\boldsymbol{A}_{client}}\left(T, P\right) = \hat{T}\right)}{\boldsymbol{Pr}\left(\boldsymbol{A}_{\boldsymbol{A}_{client}}\left(T', P\right) = \hat{T}\right)} \right) \right| \leq 2 \ln \left(\frac{1-P}{P} \right) \tag{8}$$

The steps of $\boldsymbol{A}_{\boldsymbol{client}}$ algorithm show in Algorithm 1. And $E\,(m)$ represents a fixed sequence of event elements E (including m elements), $d^{ith} \in E\,(m)$; Flip probability P(P<0.5); Time interval for defining the temporal pair Th_t.

3) SERVER-SIDE ALGORITHM A_{server}

The original data is encrypted by the client to achieve the purpose of protecting privacy. The purpose of the server-side algorithm is to restore the statistics of the client as a whole from the collected noise-added data. Therefore, the server-side algorithm is usually designed according to client algorithms. The above-mentioned client algorithm uses the actual noise-adding step of the data only to flip each bit of the temporal matrix with the probability of P. Therefore, the server-side only needs to perform certain offset transform $(\frac{\hat{T}_{ij}-P}{1-2P})$ on every bit in the data collected from clients so that the expectation of the obtained matrix element is equal to the original matrix. That is to say, let $\boldsymbol{T}$ be the raw matrix, $\hat{\mathbf{T}} = \boldsymbol{A}_{client}\,(T, P)$, $\boldsymbol{H} = \boldsymbol{A}_{server}\left(\hat{T}, P\right)$ it can be proven that for

$$\frac{Pr\left(\boldsymbol{A}_{\textbf{client}}\left(T_{rc}, P\right) = 0 | T_{rc} = 1\right) Pr\left(\boldsymbol{A}_{\text{client}}\left(T_{r'c'}, P\right) = 0 | T_{r'c'} = 0\right)}{Pr\left(\boldsymbol{A}_{\textbf{client}}\left(T_{rc}, P\right) = 0 | T_{rc} = 0\right) Pr\left(\boldsymbol{A}_{\textbf{client}}\left(T_{r'c'}, P\right) = 0 | T_{r'c'} = 1\right)} = \frac{P\,(1-P)}{(1-P)\,P} = 1 \tag{6}$$

$$\frac{Pr\left(\boldsymbol{A}_{\boldsymbol{A}_{client}}\left(T_{rc}, P\right) = 0 | T_{rc} = 1\right) Pr\left(\boldsymbol{A}_{\boldsymbol{A}_{client}}\left(T_{r'c'}, P\right) = 1 | T_{r'c'} = 0\right)}{Pr\left(\boldsymbol{A}_{clien}\left(T_{rc}, P\right) = 0 | T_{rc} = 0\right) Pr\left(\boldsymbol{A}_{\boldsymbol{A}_{client}}\left(T_{r'c'}, P\right) = 1 | T_{r'c'} = 1\right)} = \frac{P^2}{(1-P)^2} \tag{7}$$

Algorithm 2 A_{server}

// A_{server} $(P, \hat{T}^{(1)}, \hat{T}^{(2)}, \ldots, \hat{T}^{(n)})$
// Input: $P, \hat{T}^{(1)}, \hat{T}^{(2)}, \ldots, \hat{T}^{(n)}$
// Output: H
1: a $= 1/(1-2P)$
2: initialize H $[m \times m]$
3: ***for*** $c \in$ [n] ***do***
4: ***for*** $j \in [m]$ ***do***
5: ***for*** $i \in [m]$ ***do***
6: $H_{ij} \leftarrow H_{ij} + a \times T_{ij}^{(c)} - a \times P$
7: ***end for***
8: ***end for***
9: ***end for***
10: return H;

any positioned element in the matrix.

$$\mathrm{E}\left(\boldsymbol{H}_{ij}\right) = \boldsymbol{T}_{ij} \tag{9}$$

Proof: When the raw matrix element $T_{ij} = 0$, $\mathrm{E}\left(\hat{T}_{ij}\right) =$ P $\times$ 1 + (1-P) $\times$ 0 = P, then

$$\mathrm{E}\left(\frac{1}{1-2P} \times \hat{T}_{ij} - \frac{1}{1-2P} \times P\right) = \frac{1}{1-2P} \times \mathrm{E}\left(\hat{T}_{ij}\right) - \frac{1}{1-2P} \times P = 0 \tag{10}$$

When $T_{ij} = 1$, $\mathrm{E}\left(\hat{T}_{ij}\right) =$ P $\times$ 0 + (1-P) $\times$ 1 = 1 − P, then

$$\mathrm{E}\left(\frac{1}{1-2P} \times \hat{T}_{ij} - \frac{1}{1-2P} \times P\right) = \frac{1}{1-2P} \times \mathrm{E}\left(\hat{T}_{ij}\right) - \frac{1}{1-2P} \times P = 1 \tag{11}$$

Since A_{client} algorithm is independent between clients, it can be seen that for the data transmitted by n clients, $\hat{T}^{(1)}, \hat{T}^{(2)}, \ldots, \hat{T}^{(\mathrm{n})}$, we have $\mathrm{E}\left(H_{ij}\right) = T_{ij}^{(1)} + T_{ij}^{(2)} + \ldots, + T_{ij}^{(n)}$. Furthermore, A_{client} is independent of each element in the matrix, therefore, the matrix H generated by A_{server} satisfies $\mathrm{E}(H) = T^{(1)} + T^{(2)} + \ldots, + T^{(n)}$. That is, the elements of matrix H are the statistical expectation of the temporal pair of each event element.

The steps of A_{server} algorithm show in Algorithm 2.

B. REPLACE RANDOM SELECTION WITH ADJACENT TEMPORAL PAIR SELECTION

Although the above model can theoretically obtain statistical information about users' behavior temporal pairs, however, the sample space will increase dramatically(m^2) with the growth of the amount of event element, which will make it difficult to obtain a reliable statistical result. The reason is that the client algorithm takes no restrictions when randomly select basic event elements after the former has been randomly selected. Therefore, this paper replaces the random selection with the selection of adjacent temporal pair on the premise that the server-side have certain background knowledge about the "distance" between event elements. (The "distance" here can be the distance between the actual location of the user, or the relevance of the content between the websites), adjacent temporal pair refer to temporal pair within the "distance" range, the client algorithm selects only adjacent temporal pair. That is to say, the potential temporal pairs are further decomposed into adjacent temporal pairs to reduce the sample space, and the user behavior sequence is indirectly inferred by statistics on adjacent temporal pairs. Although it may not be straightforward to infer the user behavior sequence on the server-side, this method can make the model applicable to the actual problem compared to the problem that the statistical result is not reliable due to the large sample space. Due to the variety of events, this paper does not intend to give a specific distance calculation method. The following is an example to show the actual effect of the model by mining trajectories according to users' position information.

C. USER TRAJECTORY ANALYSIS

The algorithm described above is a general model algorithm, the following describes how to apply the model to infer the trajectories of users based on the temporal information of users' location information.

Since the coordinates of the location are continuous and cannot be directly represented by the temporal matrix, the map is first divided into m disjoint regions $\{Region_1, Region_2, \ldots, Region_m\}$, and the adjacent area is called a neighborhood.

Refer to the model in the first section of this chapter, the users' location at a certain moment is regarded as an event element, so behavior sequence is a trajectory. Generally, the trajectory is continuous. It is assumed that the user location is accurate recorded at a certain time (you can use the average to improve the accuracy), so every trajectory of user can be composed of several adjacent pairs, called adjacent trajectory, such as the user trajectory $\{Region_A, Region_B, Region_C\}$ can be represented by adjacent trajectory ($Region_A$,$Region_B$) and ($Region_B$,$Region_C$).

On the other hand, the paper applies the temporal matrix to represent the temporal pair of any two events. Therefore, the sample space obtained from users is theoretically m^2, which is limited by the privacy budget. However, the amount of collected data is limited by the number of users, and the amount of every basic event will decrease when the sample space is large, affecting the statistical results. In this paper, only adjacent trajectory consisting of adjacent pairs with a distance of 1 in the trajectory is selected (assuming that the trajectories of all users saved by the clients are continuous, and there are four adjacent trajectories for every region where each user is located) when analyzing the trajectories of users. That is, the selection of the adjacent trajectory replaces random selection in A_{client}, and the purpose of reducing the relative error is achieved by reducing the sample space.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental data in this paper is the location-related data collected by Microsoft Academy of Geology on 182 volunteers within five years (2007∼2012). Due to the limited participation of volunteers, this experiment will generate each adjacent pair of volunteers as different users. The data was screened in this experiment to reflect the flow of people. Only the volunteers' location data in activity-intensive areas (39.8∼40.1, 116.2∼116.4) from 6:00 am to 9:00 am were selected. With 0.003 and 0.002 respectively, we divide the area into 10,000 unit areas, and the experimental data totals 124,292. Fig.2 shows the raw trajectories of volunteers, For convenience of the display, the horizontal and vertical coordinates in the figure are obtained by converting the original longitude and latitude position into a unit area, for example, the abscissa 20 represents the actual latitude of $39.8 + 20 \times 0.003$, and the ordinate 15 represents the actual longitude of $116.2 + 15 \times 0.002$.

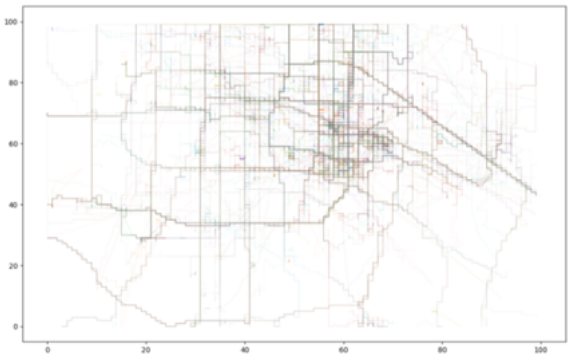
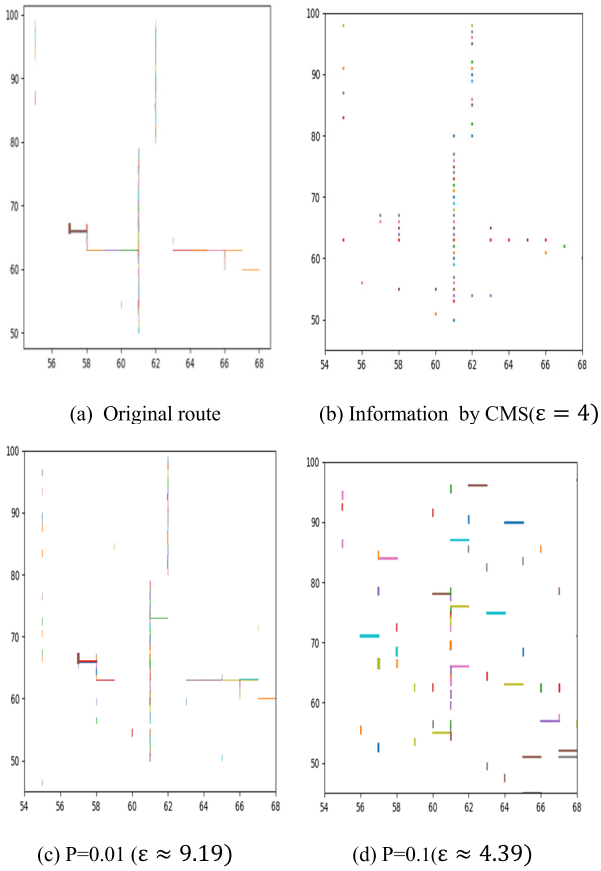


FIGURE 2. Original Trajectory.



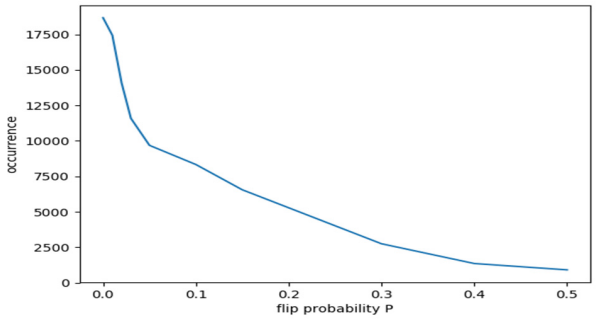
(a) Original route (b) Information by CMS($\varepsilon = 4$)

(c) P=0.01 ($\varepsilon \approx 9.19$) (d) P=0.1($\varepsilon \approx 4.39$)

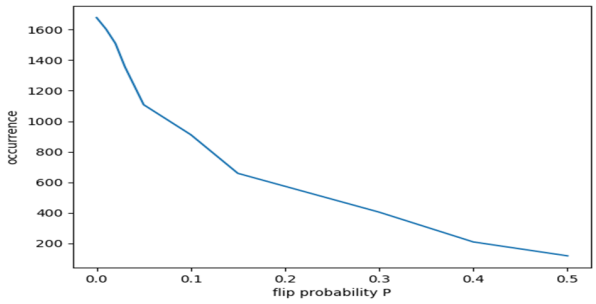
FIGURE 3. High-frequency route calculated by different algorithms and parameters.

To improve the accuracy of statistical data, the experiment sets a threshold (100) for the high occurrence frequency of displayed trajectories. Figure.3 (a) is the user trajectories composed of 100 highest frequency of occurrence adjacent trajectories in the original data (16115 in total). Fig.3 (b) is the static location information graph of the client that is only calculated for a single event element (the location information of the client) by CMS($\varepsilon = 5$) algorithm. Fig.3 (c) is the client trajectory map that satisfies $P = 0.05$ ($\varepsilon \approx 5.89$) that is calculated by our model, and Fig.3(d) when $P = 0.1$ ($\varepsilon \approx 4.39$). It can be seen that although the overall location information of the client collected by the CMS algorithm is able to reflect the overall trajectories of clients to a certain extent, it depends on the way of reading data, such as in Fig.3(b), you can draw a variety of roadmaps according to these points while the results obtained by our model can restore the routes more directly: from the perspective of adjacent trajectories, there are 18,672 times in the highest 100 adjacent trajectories (44,826 times in total), which means that 0.6% of the high-frequency adjacent trajectories occupy the total of 41.6%, under the protection of our model with $P = 0.01$ ($\varepsilon \approx 9.19$), the high-frequency trajectories excavated appeared 17439 (39.3%) times in the original adjacent trajectories and gradually decreased as p increased, as shown in Fig.4 (a); from the perspective of trajectory, there are 3228 original trajectories that meet the conditions mentioned above, of which 1678 trajectories participate in at least one adjacent trajectory in the original high-frequency trajectories, under the protection of our model with $P = 0.01$ ($\varepsilon \approx 9.19$), there are 1597 original trajectories participating in at least one adjacent trajectory in the excavated high-frequency trajectories and gradually decreased as p increased, as shown in Fig.4(b). It can be seen that the model can reflect the users' activity trajectories to some extent, and the reduction degree of the original trajectories will be reduced when the privacy budget is reduced.

The purpose which is unavailable in other local differential privacy models of the following experiment is to reveal the user flow of each route from 6:00 is to 9:00 am. In addition to the privacy budget, this experiment has another threshold t to determine whether there is an obvious flow between the two locations. For example, $t = 0.75$ for two points $\boldsymbol{a}$ and $\boldsymbol{b}$, when the occurrences of $\boldsymbol{a}$ to $\boldsymbol{b}$ is three times the occurrences of $\boldsymbol{b}$ to $\boldsymbol{a}$, it is considered that there exit user flowing from $\boldsymbol{a}$ to $\boldsymbol{b}$. This experiment only shows the trajectory extended by the route with the "most" user's flow (first select the most adjacent route from all the adjacent routes satisfying the

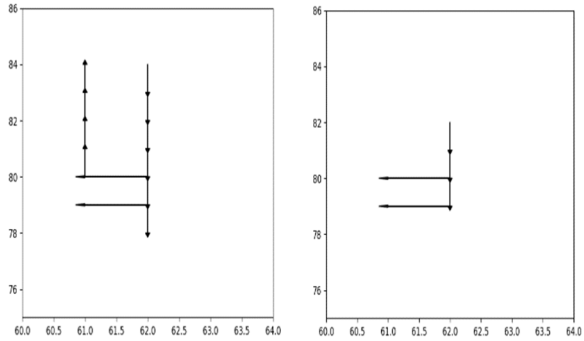


(a) The relationship between the actual number of participants in the adjacent trajectory and p

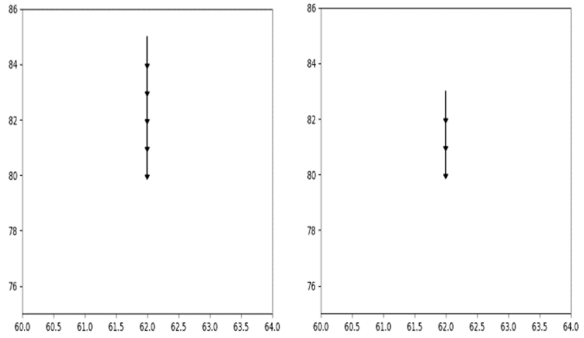


(b) The relationship between the actual number of participants and p

FIGURE 4. Relationship between the actual number of participants and the flip probability of frequent routes excavated under different parameters.



(a) Original flow (t=0.7) (b) Flow with p=0.05(t=0.7)



(c) Original flow (t=0.8) (d) Flow with p=0.05(t=0.8)

FIGURE 5. The "most flow" mined under different parameters.

threshold t condition and then start from the start point and the end point respectively traverse the adjacent route satisfying the threshold t, thereby obtaining a continuous route of "most flow" of users). Fig.5 (a) and (c) are the "most flow" routes in the original data when t = 0.7 and t = 0.8 respectively. Fig.5 (b) and (d) are the corresponding local difference results that satisfy the privacy budget of 5.8. Taking Fig.5(d) as an example, from the actual data, the number of occurrences of the route is 496, and the reverse is 31. It can be seen that the statistical route processed by the local differential privacy framework approximates the original flow, and the larger the threshold t, the higher the degree of approximation.

In order to simulate real users' trajectories, this experiment applies Geolife as the experimental data, however, the data involved a small number of volunteers while the sample space is large leads to poor statistical performance. Therefore, the data was cropped before the experiment and only the routes with a large amount of data were selected when displaying the model effect.

As can be seen from Fig. 3., the original trajectory data is divided into discrete points after our model processing. The suspected trajectory obtained from these discrete points has many possibilities. From an accurate trajectory into a series of confusing discrete point, thus, played a protective effect of the user trajectory. The LDP model has strict mathematical theoretical requirements. It can be seen from Fig. 5. that the statistical information of the user's track record is highly similar to the original track, so as to avoid the attackers from using statistical information to infer the user's privacy. The model in this paper realizes the privacy protection of users' behavior sequence information.

VII. CONCLUSION

This paper designs a user behavior sequence mine model that satisfies the local differential privacy framework and proves it. The model enables the server-side to mine users' behavior sequences while protecting their privacy, and reduce sample space by using the adjacency pair to improve the accuracy of the statistics. In order to simulate the real trajectories, the effect of the model algorithm is finally demonstrated by using Geolife as the experimental data. results show that the model can approximate the users' trajectories under the requirement of local differential privacy. It is worth mentioning that this experiment assumes that the overall activity trajectories of the users can reflect individual activity trajectory, which may require further research in some special situations.

REFERENCES

[1] C. Dwork, "Differential Privacy," in *Proc. 33rd Int. Conf. Automata, Lang., Program.* Berlin, Germany: Springer, 2006.
[2] C. Dwork, *Differential Privacy: A Survey of Results*. Berlin, Germany: Springer, 2008.
[3] F. Mcsherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2007, pp. 94–103.
[4] C. Dwork, F. McSherry, and K. Nissim, "Calibrating noise to sensitivity in private data analysis," in *Proc. Conf. Theory Cryptogr.*, 2006, pp. 265–284.

[5] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, "Discovering frequent patterns in sensitive data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2010, pp. 503–512.
[6] N. Li, W. Qardaji, D. Su, and J. Cao, "PrivBasis: Frequent itemset mining with differential privacy," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1340–1351, Jul. 2012.
[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 429–438.
[8] Ú. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized aggregatable privacy-preserving ordinal response," in *Proc. CCS*, 2014, pp. 1054–1067.
[9] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, "Differential privacy under continual observation," in *Proc. 42nd ACM Symp. Theory Comput.*, 2010.
[10] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong, "Quantifying differential privacy under temporal correlations," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, Apr. 2017, pp. 821–832.
[11] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1298–1309.
[12] O. C. Novac, M. Novac, C. Gordan, T. Berczes, and G. Bujdoso, "Comparative study of Google android, apple iOS and microsoft windows phone mobile operating systems," in *Proc. 14th Int. Conf. Eng. Modern Electr. Syst.*, Jun. 2017, pp. 154–159.
[13] X. Zhao, D. Pi, and J. Chen, "Novel trajectory privacy-preserving method based on clustering using differential privacy," *Expert Syst. Appl.*, vol. 149, Jul. 2020, Art. no. 113241.
[14] X. Fang, Q. Zeng, and G. Yang, "Local differential privacy for human-centered computing," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1063–1072, Dec. 2020.
[15] C. Xia, J. Hua, W. Tong, and S. Zhong, "Distributed K-means clustering guaranteeing local differential privacy," *Comput. Secur.*, vol. 90, Mar. 2020, Art. no. 101699.

JIANEN YAN received the Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2018. He currently works as a Lecturer with the School of Computer Science and Technology, Harbin Institute of Technology. His research interests include network simulation, domain name system security, and network data security.

YAN WANG received the M.S. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2019.

WENLING LI is currently pursuing the master's degree with the School of Computer Science and Technology, Harbin Institute of Technology. Her research interest includes data security.