




Synthesizing differentially private location traces including co-locations

Jun Narita¹ · Takao Murakami² · Hideitsu Hino^{2,4} · Masakatsu Nishigaki¹ · Tetsushi Ohki^{1,3,4} 

Published online: 29 August 2023
© The Author(s) 2023

Abstract

Privacy-preserving location synthesizers have been widely studied to perform private geo-data analysis. They have also been used for generating datasets for research or competitions. However, existing location synthesizers do not take into account the friendship information of users. Because friends tend to visit the same place at the same time in practice, a location synthesizer should consider such co-locations of friends to generate a more realistic dataset. In this paper, we propose a novel location synthesizer that generates location traces including co-locations of friends. Our location synthesizer models the information about the co-locations with two parameters: *friendship probability* and *co-location count matrix*. Our synthesizer generates a synthetic graph based on the friendship probability and then generates synthetic co-locations using the synthetic graph and the co-location count matrix. The two parameters in our synthesizer provide strong privacy guarantees—the friendship probability provides node differential privacy (DP) and the co-location count matrix provides user-level DP. We evaluate our synthesizer using two real datasets. Our experimental results show that our synthesizer preserves co-locations and other statistical features while providing DP with reasonable privacy budgets, e.g., 0.2-node DP and 2-user-level DP.

Keywords Location synthesizer · Differential privacy · User-level DP · Node DP · Co-location

1 Introduction

Location-based services (LBS), such as point-of-interest (POI) search and route suggestion, have been increasingly used in recent years. Consequently, a large amount of location traces (time-series location trails) are accumulated in an LBS provider. The LBS provider can provide the location traces

to a third party to perform various geo-data analyses such as finding popular POIs [54], modeling human mobility patterns [24], and semantic annotation of POIs [51]. However, the disclosure of the traces can lead to a serious privacy issue because they may include sensitive locations, e.g., homes and hospitals. In addition, several methods have been developed to identify the users' behaviors [27, 53] or to re-identify traces from pseudonymized traces [12, 30, 31].

Many privacy-preserving location synthesizers have been proposed to address this privacy issue [3, 4, 13, 16, 28, 29]. These approaches first train a generative model from real location traces. Then they generate synthetic traces based on the trained generative model. Ideally, synthetic traces preserve various statistical features, such as a population distribution [54] and transition matrix [47], while strongly protecting user privacy. The preserved statistical features play a significant role in the geo-data analysis. Moreover, applications of synthetic traces are not limited to geo-data analysis. For example, they are useful for research purposes [20, 33] and competitions [28, 38].

Existing location synthesizers, however, do not consider friendship information between users. In particular, friends tend to visit the same place at the same time [50]. This event

✉ Tetsushi Ohki
ohki@inf.shizuoka.ac.jp

Jun Narita
narita@sec.inf.shizuoka.ac.jp

Takao Murakami
tmura@ism.ac.jp

Hideitsu Hino
hino@ism.ac.jp

Masakatsu Nishigaki
nisigaki@inf.shizuoka.ac.jp

¹ Shizuoka University, Shizuoka, Japan

² The Institute of Statistical Mathematics, Tachikawa, Japan

³ National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

⁴ RIKEN AIP, Tokyo, Japan

is called a *co-location* [35, 36]. Co-locations of friends are important to make synthetic traces more realistic. For example, a recent study [50] shows that there is a correlation between co-locations and friendships on Twitter. Therefore, synthetic traces including co-locations of friends can be used as a dataset to study the effectiveness of friend suggestion algorithms based on locations.

In this paper, we propose a novel location synthesizer that synthesizes location traces including co-locations of friends. To preserve co-location information, our proposed method trains two parameters: *friendship probability* and *co-location count matrix*. The friendship probability represents a probability that two users are friends. The co-location count matrix is composed of a co-location count for each time instant and each location. Thus, it models a location that is likely to be visited by friends at a certain time period, e.g., amusement parks in the daytime and restaurants at night.

Our location synthesizer works as follows. First, we train the two parameters using a friendship (or social) graph and location traces, which we call a *training graph* and *training traces*, respectively. We train the friendship probability from a training graph and the co-location count matrix from training traces. Then, we generate a *synthetic graph* using the friendship probability. We generate co-locations in synthetic traces using the synthetic graph and the co-location count matrix. Finally, we generate other locations using an existing location synthesizer providing DP [4, 13, 16] based on the Markov chain model.

One promising feature of our location synthesizer is that both the two parameters in our synthesizer provide strong privacy guarantees: differential privacy (DP) [10], which is known as a gold standard for data privacy. The friendship probability provides ϵ_1 -(*bounded*) *node DP* [21], a strong type of DP on graphs, for the training graph. The co-location count matrix provides ϵ_2 -*user-level DP* [9], which is a strong type of DP on time-series data, for the training traces. The parameters ϵ_1 and ϵ_2 are non-negative real values and called the privacy budgets [10]. It is well known that DP strongly protects user privacy when they are small, e.g., smaller than 1 or 2 [18, 34].

We use the existing synthesizer providing ϵ_3 -user-level DP for the training traces [4, 13, 16] to generate locations other than co-locations. Then, by the composition theorem in DP [10], the entire synthetic traces provide ϵ_1 -node DP for the training graph and $(\epsilon_2 + \epsilon_3)$ -user-level DP for the training traces. As with [4, 13, 16], we use the privacy budget ϵ_3 to preserve statistical features, such as a population distribution [54] and transition matrix [47]. We additionally use ϵ_1 and ϵ_2 to preserve the information about co-locations, which has not been considered in the existing synthesizers. ϵ_1 and ϵ_2 are additional costs required to incorporate co-locations into synthetic traces. Table 1 summarizes the existing synthesizers [4, 13, 16] and our synthesizer.

Through comprehensive experiments, we show that our synthesizer preserves the information about co-locations and other statistical features (e.g., population distribution, transition matrix) with reasonable privacy budgets, e.g., $\epsilon_1 = 0.2$ and $\epsilon_2 = \epsilon_3 = 1$.

1.1 Our contributions

In summary, we provide the following contributions:

- We propose a novel location synthesizer that generates location traces including co-locations of friends. To our knowledge, we are the first to synthesize traces including co-locations. Our synthesizer models the information about co-locations of friends with two parameters: friendship probability and co-location count matrix. The friendship probability provides node DP, and the co-location count matrix provides user-level DP.
- We evaluate our synthetic traces using two real datasets: the Foursquare [50] and Gowalla [25] datasets. Our experimental results show that our synthetic traces preserve the information about co-locations and other statistical features (e.g., population distribution, transition matrix) while satisfying DP with reasonable privacy budgets, e.g., 0.2-node DP ($\epsilon_1 = 0.2$) for the training graph and 2-user-level DP ($\epsilon_2 = \epsilon_3 = 1$) for the training traces.

This paper is a significant extension of the previously published conference paper [32]. The main enhancements are as follows:

- In [32], we did not discuss the overall privacy guarantee for the entire dataset including the training graph and training traces. In this paper, we define *total DP* (Definition 3) to provide the overall privacy guarantee for the entire dataset and prove that our synthesizer provides total DP (Sect. 4.5).
- In [32], we generated a synthetic graph based on the Erdős-Rényi (ER) model [5]. However, it is well known that the ER model does not reflect an actual graph property. Specifically, most actual graphs have a power-law degree distribution [1], whereas the ER model does not. In this paper, we also generate a synthetic graph based on the Barabási-Albert (BA) model [1], which has a power-law degree distribution and therefore is much more realistic. Our BA graph also provides node DP. We show through experiments that our BA graph has a degree distribution close to a training graph (Sects. 4 and 5).
- In [32], we evaluated our synthesizer using only one dataset: the Foursquare dataset. In this paper, we add the Gowalla dataset to make the evaluation more comprehensive (Sect. 5).

Table 1 The existing and our location synthesizers

	Privacy for a training graph	Privacy for training traces	Co-locations	Other statistical features
Existing synthesizer [4, 13, 16]	N/A	ϵ_3 -user-level DP	×	✓
Our synthesizer	ϵ_1 -node DP	$(\epsilon_2 + \epsilon_3)$ -user-level DP	✓	✓

The existing synthesizers do not use a training graph and therefore do not preserve the information about co-locations of friends. We use additional privacy budgets ϵ_1 and ϵ_2 to preserve this new information while keeping other statistical features, such as a population distribution and transition matrix

- In [32], we did not show examples of co-locations preserved in our synthesizer. In this paper, we show ten pairs of locations and time instants where co-location events are most likely to occur for each dataset. Using these examples, we show how well our synthesizer preserves the information about co-locations (Sect. 5).
- In [32], we evaluated the utility of the probability distributions (e.g., population distribution, transition matrix) by only the mean absolute error (MAE) and the mean squared error (MSE). In this paper, we also evaluate the utility of them using the Kullback–Leibler (KL) divergence and the Jensen–Shannon (JS) divergence. These utility metrics also make our evaluation more comprehensive (Sect. 5).
- In [32], we did not provide the details of how to apply Privelet [49] in our location synthesizer. In this paper, we provide the details of applying Privelet and provide a proof that Privelet provides user-level DP (Appendix 1).

1.2 Paper organization

The rest of this paper is organized as follows. In Sect. 2, we review the previous work closely related to ours. In Sect. 3, we introduce some preliminaries for our work. In Sect. 4, we propose our location synthesizer. In Sect. 5, we show our experimental results. Finally, in Sect. 6, we conclude this paper with future directions of this work.

2 Related work

2.1 Co-locations

A co-location refers to an event that two users are in the same place at the same time. In particular, we focus on a co-location of friends and generate synthetic traces on the basis of the co-location.

Co-locations have been widely studied, especially through the impact on location privacy and the relationship with friendships. Olteanu et al. [35, 36] showed that co-location information improves the accuracy of location inference attacks. The users' benefits of sharing co-locations and the impact of co-locations on location privacy were also studied

in [37]. Yang et al. [50] showed that there is a correlation between co-location information and friendships on Twitter. However, to the best of the authors' knowledge, there are no existing studies that use co-locations to synthesize location traces.

2.2 Location synthesizers

The generation of synthetic location traces has long been recognized as an important research subject; see [3, 29] for detailed surveys. Bindschaeder and Shokri [3] developed a synthetic location generation algorithm considering semantic features of locations. For example, most people tend to stay a night at their homes, which are geographically different but semantically the same. Their synthesizer preserves this kind of information while satisfying their own privacy notion called plausible deniability. Bindschaeder et al. [4] also proposed a synthetic data generator for synthesizing various types of data with DP. This synthesizer can be applied to various data, including location traces [29]. In the case of location traces, the synthetic data generator in [4] trains a transition matrix common to all users as a generative model (see [29] for details). Some studies [13, 16] proposed more complicated algorithms for generating synthetic traces with DP using a transition matrix common to all users. Murakami et al. [29] proposed a method to generate synthetic traces with high utility based on an observation that there should be a small number of typical groups of users, e.g., those who often go to malls and those who frequently go to offices. Specifically, they clustered a transition matrix for each user using tensor factorization. They also applied a modified version of their algorithm to a location anonymization contest [28]. The synthesizers in [3, 28, 29] do not provide DP, whereas the synthesizers in [4, 13, 16] provide DP.

Thus, many studies have been made on the artificial generation of location traces. However, to the best of the authors' knowledge, no generation method using co-locations has been proposed so far. As explained above, it is shown in [50] that there is a correlation between co-locations and friendships, i.e., friends tend to be in the same place at the same time. Hence, to synthesize more realistic traces, it is important to take the co-location information into account for generating location traces.

Table 2 Basic notations

Symbol	Description
\mathcal{U}	Finite set of users in training data
n	Number of users ($n = \mathcal{U} $)
\mathcal{X}	Finite set of locations
\mathcal{T}	Finite set of time instants
\mathcal{E}	Finite set of events ($\mathcal{E} = \mathcal{X} \times \mathcal{T}$)
\mathcal{R}	Finite set of traces ($\mathcal{R} = \mathcal{U} \times \mathcal{E}^*$)
\mathcal{S}	Finite set of training traces ($\mathcal{S} \subseteq \mathcal{R}$)
u_i	i -th user ($u_i \in \mathcal{U}$)
x_i	i -th location ($x_i \in \mathcal{X}$)
t_i	i -th time instant ($t_i \in \mathcal{T}$)
\mathbf{A}	Training adjacency matrix ($\mathbf{A} \in \{0, 1\}^{n \times n}$)
s_i	i -th user's training trace ($s_i \subseteq \mathcal{S}$)

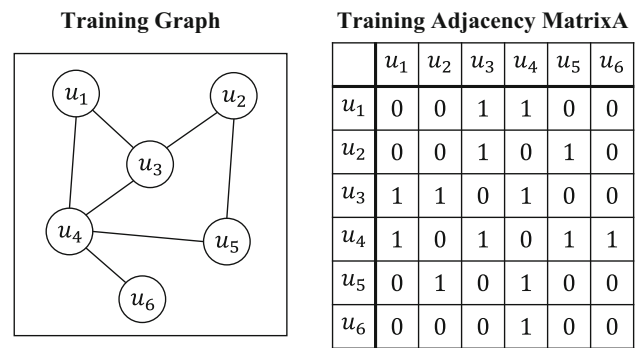
Finally, a recent study [48] empirically showed that synthetic data did not provide a better trade-off between privacy and utility for data analysis than a traditional anonymization (generalization and deletion) technique satisfying k -anonymization. In the case of geo-data analysis, the result in [48] indicates that location synthesizers might not provide a better empirical trade-off between privacy and utility than location obfuscation (e.g., generalization, deletion) methods providing k -anonymity [2, 6]. However, synthesizers are useful for not only data analysis but also generating a *dataset* for research [20, 33] or competitions [28, 38]. These important applications cannot be realized by generalization and deletion. We aim at generating a synthetic yet realistic dataset that is useful for research or competitions by preserving various statistical features including co-locations of friends.

3 Preliminaries

In this section, we introduce some preliminaries for our work. In Sect. 3.1, we define basic notations used in this paper. In Sect. 3.2, we explain friendship graphs and location traces. In Sect. 3.3, we describe our threat model and review differential privacy.

3.1 Basic notations

Below, we define the basic notations used in this paper. Let \mathbb{R} , $\mathbb{R}_{\geq 0}$, \mathbb{N} , and $\mathbb{Z}_{\geq 0}$, be the set of real numbers, non-negative real numbers, natural numbers, and non-negative integers, respectively. For a finite set \mathcal{Z} , let \mathcal{Z}^* be the set of all finite sequences of elements of \mathcal{Z} . Let $\mathcal{P}(\mathcal{Z})$ be the power set of \mathcal{Z} . For $a \in \mathbb{N}$, let $[a] = \{1, 2, \dots, a\}$. We represent a matrix as a bold capital letter, such as \mathbf{M} . We denote the i -th row of the matrix \mathbf{M} by \mathbf{M}_i and the (i, j) -th element of \mathbf{M} by \mathbf{M}_{ij} .

**Fig. 1** Examples of the training graph and the corresponding adjacency matrix ($n = 6$)

We follow the notations in [29] to define users, locations, and time. Specifically, let \mathcal{U} be a finite set of users in training data. Let $n \in \mathbb{N}$ be the number of users, i.e., $n = |\mathcal{U}|$. Let $u_i \in \mathcal{U}$ be the i -th user, i.e., $\mathcal{U} = \{u_1, \dots, u_n\}$. We consider discrete locations. For example, we can divide an area of interest into some regions or extract some POIs. Let \mathcal{X} be a finite set of locations. Let x_i be the i -th location. We also consider a discrete version of time, called time instant. For example, we can round down minutes to a multiple of 20. Let \mathcal{T} be a finite set of time instants. Let $t_i \in \mathcal{T}$ be the i -th time instant.

The basic notations used in this paper are shown in Table 2. Symbols that are not explained in Sect. 3.1 will be explained in Sect. 3.2.

3.2 Friendship graphs and location traces

Friendship graphs A friendship (or social) graph includes friendship information between any pair of two users. It is represented as an undirected graph, where a node represents a user and an edge represents that two users are friends. The friendship graph is also represented as an adjacency matrix of size $n \times n$. In the adjacency matrix, an element between two friends is set to 1, and an element between two users who are not friends is set to 0. Diagonal elements are set to 0 because there are no friend relationships between users and themselves.

Figure 1 shows examples of the friendship graph and the corresponding adjacency matrix in training data. In this work, we call them the *training graph* and the *training adjacency matrix*. In this example, user u_1 is a friend with u_3 and u_4 . User u_6 is a friend with only u_4 .

Formally, let $\mathbf{A} \in \{0, 1\}^{n \times n}$ be a training adjacency matrix. \mathbf{A}_i is the i -th row of \mathbf{A} . In Fig. 1, $\mathbf{A}_1 = (0, 0, 1, 1, 0, 0)$, \dots , $\mathbf{A}_6 = (0, 0, 0, 1, 0, 0)$.

Location traces A location trace includes a location in each time instant. A pair of the location and the time instant is called an *event* [29, 45].

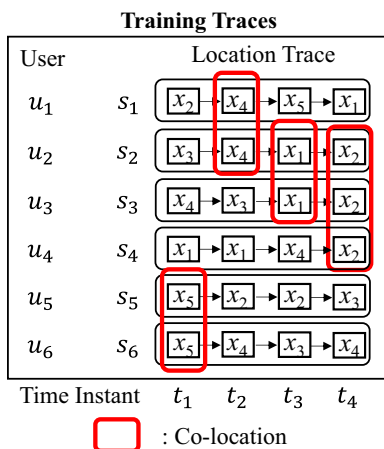


Fig. 2 Examples of the training traces ($n = 6, |\mathcal{X}| = 5, |\mathcal{T}| = 4$)

Figure 2 shows examples of the location traces in training data, which we call the *training traces*. We mark co-location events with red. In this example, users u_1 and u_2 have a co-location event at location x_4 and time instant t_2 . Users u_2 and u_3 have a co-location event at x_1 and t_3 .

Let $\mathcal{E} = \mathcal{X} \times \mathcal{T}$ be a finite set of events. Let $\mathcal{R} = \mathcal{U} \times \mathcal{E}^*$ be a finite set of traces. Let $\mathcal{S} \subseteq \mathcal{R}$ be a finite set of training traces. Let $s_i \in \mathcal{S}$ be the i -th training trace. In Fig. 2, $s_1 = (u_1, (x_2, t_1), (x_4, t_2), (x_5, t_3), (x_1, t_4)), \dots, s_6 = (u_6, (x_5, t_1), (x_4, t_2), (x_3, t_3), (x_4, t_4))$, and $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$. Note that although each trace includes four events in Fig. 2, we do not assume the length of the training trace is the same among all users. In fact, the length of the training trace is different in our experiments.

3.3 Threat model and differential privacy

Threat model We use training data that includes a friendship graph and location traces to generate synthetic traces. We assume that the number n of users in training data is public. We also assume that an adversary has any background knowledge other than the training data. The adversary obtains the synthetic traces and attempts to violate user privacy in the training data on the basis of the synthetic traces and the background knowledge. For example, the adversary performs a membership inference attack [39, 44], which infers whether a location trace of a specific user is included in the training data.

To strongly protect user privacy in the training data from the adversary with any background knowledge, we use differential privacy (DP) [8, 10] as a privacy metric. DP provides user privacy against adversaries with any background knowledge. Below, we explain DP for training graphs and training traces.

DP for training graphs There are two types of DP on graphs: *edge DP* and *node (or vertex) DP* [21, 46]. Edge DP hides

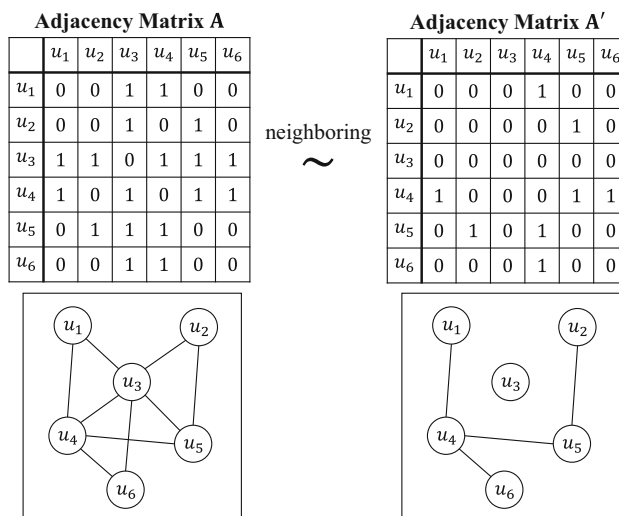


Fig. 3 Example of neighboring adjacency matrices \mathbf{A} and \mathbf{A}' in (bounded) node DP [21] ($n = 6$)

the existence of one edge, i.e., friendship. In contrast, node DP hides the existence of all edges connected to one node. Therefore, node DP guarantees much stronger privacy than edge DP and much more difficult to attain [15, 19, 41]. To strongly protect user privacy in the training graphs, we tackle this challenge and use node DP as a privacy metric.

The original definition of node DP [15] follows the direction of *unbounded* DP [22], where a neighboring graph is obtained by removing one node. Since we assume that n is public (as described in Sect. 3.3 “Threat Model”), we use node DP in [21] that follows the direction of *bounded* DP [22], where a neighboring graph is obtained by changing at most $n - 1$ edges of one node. Bounded node DP is also much stronger than edge DP because it hides all sensitive edges of a user.

Formally, (bounded) node DP in [21] considers two neighboring adjacency matrices \mathbf{A} and \mathbf{A}' such that \mathbf{A}' is obtained by an arbitrary rewiring of edges connected to one node. In other words, \mathbf{A} and \mathbf{A}' differ in at most $n - 1$ edges of one user. Fig. 3 shows an example of two neighboring adjacency matrices \mathbf{A} and \mathbf{A}' ($n = 6$). In this example, \mathbf{A} and \mathbf{A}' differ in $n - 1 = 5$ edges connected to u_3 .

Using the notion of neighboring adjacency matrices, (bounded) node DP is defined as follows.

Definition 1 [ϵ_1 -(bounded) node DP [21]] Let $\epsilon_1 \in \mathbb{R}_{\geq 0}$. A randomized mechanism \mathcal{M}_1 with domain $\{0, 1\}^{n \times n}$ provides ϵ_1 -node DP if for any two neighboring adjacency matrices $\mathbf{A}, \mathbf{A}' \in \{0, 1\}^{n \times n}$ that differ in at most $n - 1$ edges of one user and any $z \in \text{Range}(\mathcal{M}_1)$,

$$\Pr[\mathcal{M}_1(\mathbf{A}) = z] \leq e^{\epsilon_1} \Pr[\mathcal{M}_1(\mathbf{A}') = z]. \tag{1}$$

By (1), if ϵ_1 is close to 0, then \mathbf{A} and \mathbf{A}' are almost equally likely. Thus, an adversary who obtains the output of \mathcal{M}_1 cannot determine whether it is come from \mathbf{A} or \mathbf{A}' . If the privacy budget ϵ_1 is small (e.g., smaller than 1 or 2 [18, 34]), each user's privacy is strongly protected.

DP for training traces For time-series data such as training traces, there are two types of DP: *event-level DP* and *user-level DP* [9]. Event-level DP protects one event in time-series data. In contrast, user-level DP protects the entire history (i.e., entire time-series data) of one user. Thus, user-level DP guarantees much stronger privacy than event-level DP and is much more difficult to attain. To strongly protect user privacy in training traces, we use user-level DP.

Formally, user-level DP for training traces considers two neighboring sets \mathcal{S} and \mathcal{S}' of traces such that \mathcal{S}' is obtained by changing the entire trace of one user in \mathcal{S} . For example, consider a set \mathcal{S}' of traces obtained by changing s_2 in Fig. 2 to s'_2 as follows: $s'_2 = (u_2, (x_1, t_1), (x_1, t_2), (x_1, t_3))$ (note that the trace length can also be changed). In this example, \mathcal{S} and \mathcal{S}' are neighboring sets.

Using neighboring sets of traces, user-level DP is defined as follows.

Definition 2 [ϵ_2 -user-level DP] Let $\epsilon_2 \in \mathbb{R}_{\geq 0}$. A randomized mechanism \mathcal{M}_2 with domain $\mathcal{P}(\mathcal{R})$ provides ϵ_2 -user-level DP if for any two neighboring sets $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{R}$ of traces that differ in the entire trace of one user and any $z \in \text{Range}(\mathcal{M}_2)$,

$$\Pr[\mathcal{M}_2(\mathcal{S}) = z] \leq e^{\epsilon_2} \Pr[\mathcal{M}_2(\mathcal{S}') = z]. \quad (2)$$

By (2), if ϵ_2 is close to 0, an adversary who obtains the output of \mathcal{M}_2 cannot determine whether it is come from \mathcal{S} or \mathcal{S}' . Thus, the privacy of each user is strongly protected when the privacy budget ϵ_2 is small.

Note that the neighboring sets \mathcal{S} and \mathcal{S}' have the same number of users. Thus, user-level DP follows the direction of bounded DP [22] in the same way as node DP in [21].

Total DP In this work, we use a dataset that includes both the training graph (adjacency matrix) \mathbf{A} and the training traces \mathcal{S} . Assume that we use an algorithm providing ϵ_1 -node DP for \mathbf{A} and an algorithm providing ϵ_2 -user-level DP for \mathcal{S} . Then, a natural question would be: *what is the total privacy guarantee of these algorithms for a single user?* To answer this question, we define total DP.

The above dataset can be expressed as a tuple $(\mathbf{A}, \mathcal{S})$. We consider two neighboring tuples $(\mathbf{A}, \mathcal{S})$ and $(\mathbf{A}', \mathcal{S}')$ such that $(\mathbf{A}', \mathcal{S}')$ is obtained by changing the entire trace and at most $n - 1$ edges of one user, i.e., an arbitrary rewiring of *all personal data* of one user.

Using the neighboring tuples, we define total DP:

Definition 3 [ϵ -total DP] Let $\epsilon \in \mathbb{R}_{\geq 0}$. A randomized mechanism \mathcal{M} with domain $\{0, 1\}^{n \times n} \times \mathcal{P}(\mathcal{R})$ provides ϵ -node DP if for any two neighboring tuples $(\mathbf{A}, \mathcal{S})$ and $(\mathbf{A}', \mathcal{S}')$ that

differ in the entire trace and at most $n - 1$ edges of one user and any $z \in \text{Range}(\mathcal{M})$,

$$\Pr[\mathcal{M}(\mathbf{A}, \mathcal{S}) = z] \leq e^\epsilon \Pr[\mathcal{M}(\mathbf{A}', \mathcal{S}') = z]. \quad (3)$$

ϵ is a *total* privacy budget over the entire dataset $(\mathbf{A}, \mathcal{S})$. We can answer our question above by using total DP:

Proposition 1 Let $\epsilon_1, \epsilon_2 \in \mathbb{R}_{\geq 0}$. Assume that randomized mechanisms \mathcal{M}_1 with domain $\{0, 1\}^{n \times n}$ and \mathcal{M}_2 with domain $\mathcal{P}(\mathcal{R})$ provide ϵ_1 -node DP and ϵ_2 -user-level DP, respectively. In addition, assume that \mathcal{M}_1 and \mathcal{M}_2 are independently executed. Then, the independent execution of \mathcal{M}_1 and \mathcal{M}_2 provides $(\epsilon_1 + \epsilon_2)$ -total DP.

Proof The randomness in \mathcal{M}_1 is independent of the randomness in \mathcal{M}_2 . Thus, given inputs $\mathbf{A} \in \{0, 1\}^{n \times n}$ and $\mathcal{S} \subseteq \mathcal{R}$, the outputs of \mathcal{M}_1 and \mathcal{M}_2 are independent of each other. Therefore, for any two neighboring tuples $(\mathbf{A}, \mathcal{S})$ and $(\mathbf{A}', \mathcal{S}')$ and for any $z_1 \in \text{Range}(\mathcal{M}_1)$ and $z_2 \in \text{Range}(\mathcal{M}_2)$, we have

$$\begin{aligned} & \Pr[(\mathcal{M}_1(\mathbf{A}), \mathcal{M}_2(\mathcal{S})) = (z_1, z_2)] \\ & \leq e^{\epsilon_1} \Pr[(\mathcal{M}_1(\mathbf{A}'), \mathcal{M}_2(\mathcal{S})) = (z_1, z_2)] \quad (\text{by (1)}) \\ & \leq e^{\epsilon_1 + \epsilon_2} \Pr[(\mathcal{M}_1(\mathbf{A}'), \mathcal{M}_2(\mathcal{S}')) = (z_1, z_2)] \quad (\text{by (2)}), \end{aligned}$$

which proves Proposition 1. \square

Proposition 1 means that if we provide ϵ_1 -node DP for the training graph and ϵ_2 -user-level DP for the training traces, then the total privacy budget ϵ over the entire dataset $(\mathbf{A}, \mathcal{S})$ is the sum of ϵ_1 and ϵ_2 , i.e., $\epsilon = \epsilon_1 + \epsilon_2$.

Both node DP in [21] and user-level DP follow the direction of bounded DP, as explained above. Thus, total DP also follows the direction of bounded DP.

4 Proposed method

We propose a novel location synthesizer that generates synthetic traces including co-locations of friends. In Sect. 4.1, we describe the overview of our synthesizer. We explain the details of our synthesizer in the remaining subsections. In Sects. 4.2 and 4.3, we explain how to train the friendship probability and the co-location count matrix, respectively. In Sect. 4.4, we explain how to generate synthetic traces based on the friendship probability and the co-location count matrix. In Sect. 4.5, we provide end-to-end privacy analysis of our synthesizer.

4.1 Overview

Figure 4 shows the overview of our location synthesizer. The main feature of our synthesizer is that it generates

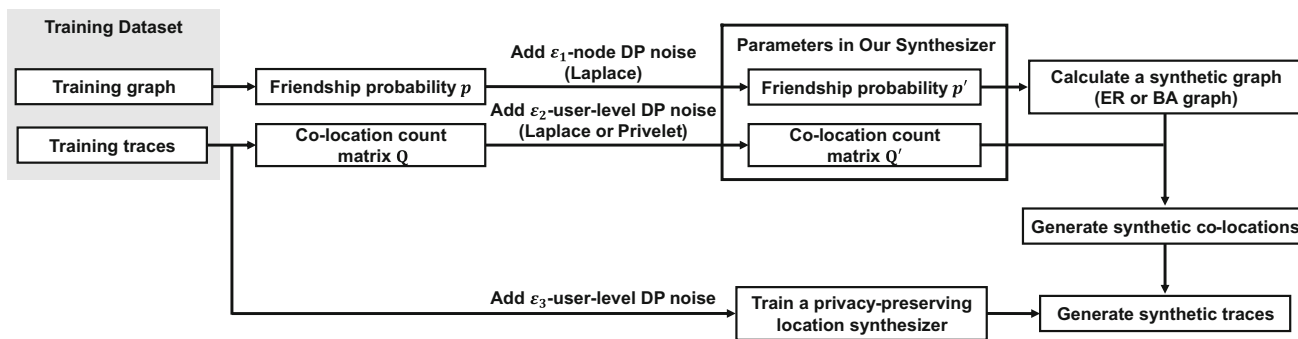


Fig. 4 Overview of our location synthesizer. We first train the friendship probability p' , the co-location count matrix Q' , and an existing location synthesizer [4, 13, 16] based on the Markov chain model. Then, we cal-

culate a synthetic graph based on p' . Finally, we generate co-locations using the synthetic graph and Q' and other locations using the existing location synthesizer

synthetic traces including co-locations. The synthetic traces preserve a friendship probability (i.e., how likely two users will be friends) and a co-location count matrix (i.e., how likely a co-location event will happen at a certain location for each time instant). The two parameters in our synthesizer strongly protect user privacy; the friendship probability provides node DP and the co-location count matrix provides user-level DP.

Our location synthesizer uses a location dataset that includes both location traces and a friendship graph (e.g., Foursquare dataset [50] and Gowalla dataset [25]) as training data. Below, we briefly explain how to train parameters from the training data and how to generate synthetic traces from the parameters.

Training parameters From a training graph, we first calculate a friendship probability $p \in [0, 1]$, which represents a probability that two users are friends. Then we add the Laplace noise [10] to p to obtain a noisy friendship probability p' providing node DP.

From training traces, we first calculate a co-location count matrix $Q \in \mathbb{Z}_{\geq 0}^{|\mathcal{I}| \times |\mathcal{L}|}$, which comprises a co-location count for each time instant and each location. Specifically, we calculate Q by simply counting co-locations between friends. Then we add noise to Q to obtain a noisy co-location count matrix Q' providing user-level DP.

The simplest approach to providing DP for Q' is to add the Laplace noise to each element in Q . We refer to this approach as the *Laplace mechanism*.

Another approach is to apply Privelet (for one-dimensional nominal data) [49], a DP mechanism based on a wavelet transform, to each row of Q . Privelet uses a nominal wavelet transform to a one-dimensional count vector and adds the Laplace noise to each wavelet coefficient, i.e., each node in a tree structure. When a category (or tree structure) of locations is known, Privelet significantly reduces the amount of noise for each category. For example, categories (e.g., “travel & transport” and “shopping”) and subcategories (e.g.,

“train station”, “airport”, “bookstore”, and “discount store”) of POIs are available in the Foursquare dataset [7] and the Gowalla dataset [25]. Thus, we can use Privelet to provide DP for Q' with a much smaller amount of noise for each POI category.

Other methods than the Laplace mechanism and Privelet include the hierarchical method in [40] and the matrix mechanism in [23, 52]. Specifically, the hierarchical method in [40] finds the optimal branching factor in a tree that minimizes the mean square error of a range query. However, this optimization method cannot be applied to our setting where a tree structure of locations (i.e., POI categories and subcategories) is given in advance. In addition, the matrix mechanism in [23, 52] is inefficient and provides worse utility than the hierarchical method, as described in [34]. Therefore, we focus on the Laplace mechanism and Privelet and evaluate these two mechanisms in our experiments.

Generating synthetic traces Based on two parameters p' and Q' , we generate synthetic traces including co-locations. First, we calculate a synthetic graph based on the friendship probability p' . In this paper, we propose to calculate two types of graphs: a graph based on the Erdős-Rényi model (the ER graph) [5] and a graph based on the Barabási-Albert model [1] (the BA graph). Note that both the ER and BA graphs are generated using only a single friendship probability p' providing node DP. Thus, by the immunity to the post-processing [10], both the ER and BA graphs also provide node DP.

The BA graph is more realistic than the ER graph in that it has a power-law degree distribution. In our experiments, we show that the BA graph preserves statistical properties of the training graph, including the average degree (number of friends) and the degree distribution.

After synthesizing the friendship graph, we generate co-locations of friends at a specific location and a time instant based on the synthetic graph and Q' . The generated co-locations preserve the information about co-locations in the training data, e.g., friends tend to meet at a restaurant from

7PM to 8PM. After generating co-locations, we generate other locations using an existing differentially private location synthesizer [4, 13, 16] based on the Markov chain model, which models human movement patterns as a transition matrix.

The existing synthesizers in [4, 13, 16] provide user-level DP for the training traces. Then, by the composition theorem [10], the entire synthetic traces provide node DP for the training graph and user-level DP for the training traces.

Remark As shown in Fig. 4, we add DP noise to the friendship probability p and the co-location count matrix \mathbf{Q} independently from one another. However, there might be a correlation between p and \mathbf{Q} , and it might be possible to add smaller noise by considering the correlation.

For example, assume that a training dataset is collected from students in a class. In this case, many users (students) tend to be friends, and co-locations tend to happen in the school. Suppose we publish \mathbf{Q}' that includes a large count in the school. Then, it may suffice to add small noise to p because, given \mathbf{Q}' , it is highly unlikely that the friendship probability is small. Thus, the amount of noise might be reduced by considering the correlation between p and \mathbf{Q} .

We argue that this kind of improvement is extremely challenging in practice because the correlation information itself needs to satisfy DP, e.g., we may need other datasets to obtain differentially private correlation information. Therefore, we treat p and \mathbf{Q} independently and leave the improvement of our algorithm using the correlation for future work.

4.2 Training the friendship probability p'

Training p' Below, we explain how to train the noisy friendship probability $p' \in [0, 1]$ in detail.

We first calculate the friendship probability p by simply calculating the proportion of edges in the training graph, i.e., the proportion of 1s in non-diagonal elements of the training adjacency matrix \mathbf{A} . For example, we can calculate p as $p = \frac{14}{6 \times 5} = 0.467$ in Fig. 1. If $n = 1$, then we calculate p as $p = 0$.¹ After calculating p , we calculate p' by adding the Laplace noise with mean 0 and scale $\frac{2}{n\epsilon_1}$. For $b \in \mathbb{R}_{\geq 0}$, let $\text{Lap}(b)$ be the Laplace noise with mean 0 and scale b . Then we calculate p' as follows: $p' = p + \text{Lap}\left(\frac{2}{n\epsilon_1}\right)$.

DP of p' Let $\mathcal{M}_1^{\text{Lap}} : \{0, 1\}^{n \times n} \rightarrow [0, 1]$ be a randomized mechanism that takes a training adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ as input and outputs $p' \in [0, 1]$. $\mathcal{M}_1^{\text{Lap}}$ has the following privacy guarantee.

Theorem 1 $\mathcal{M}_1^{\text{Lap}}$ provides ϵ_1 -node DP.

¹ Note that the number n of users in training data is much larger than 1 in practice. Here we clarify how to calculate p when $n = 1$ because it is used in the proof of node DP (Theorem 1).

Proof Let $f : \{0, 1\}^{n \times n} \rightarrow [0, 1]$ be a function that takes a training adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ as input and outputs the friendship probability $p \in [0, 1]$. Let Δf be the global sensitivity [10] of f given by

$$\Delta f = \max_{\mathbf{A} \sim \mathbf{A}'} |f(\mathbf{A}) - f(\mathbf{A}')|, \tag{4}$$

where $\mathbf{A} \sim \mathbf{A}'$ represents that \mathbf{A} and \mathbf{A}' are neighboring matrices that differ in at most all edges of one user.

Below, we upper bound the global sensitivity Δf . Let $d \in \mathbb{Z}_{\geq 0}$ be the number of 1s in \mathbf{A} . Δf takes the maximum value when $n - 1$ edges of a user are removed (or added). If $n \geq 2$, we have

$$\begin{aligned} |f(\mathbf{A}) - f(\mathbf{A}')| &\leq \frac{d}{n(n-1)} - \frac{d-2(n-1)}{n(n-1)} \leq \frac{2(n-1)}{n(n-1)} = \frac{2}{n}. \end{aligned} \tag{5}$$

Note that the denominator of $f(\mathbf{A}')$ is $n(n - 1)$ (rather than $(n - 1)(n - 2)$) because we consider a bounded version of node DP [21] that does not remove a node to obtain a neighboring graph, as described in Sect. 3.3.²

If $n = 1$, then $|f(\mathbf{A}) - f(\mathbf{A}')| = |0 - 0| = 0$. Thus, for any $n \in \mathbb{N}$, Δf in (4) can be upper bounded as $\Delta f \leq \frac{2}{n}$.

Adding the Laplace noise $\text{Lap}\left(\frac{\Delta f}{\epsilon_1}\right)$ to p provides ϵ_1 -DP [10]. Therefore, the randomized mechanism $\mathcal{M}_1^{\text{Lap}}$, which adds $\text{Lap}\left(\frac{2}{n\epsilon_1}\right)$ to p , provides ϵ_1 -node DP. \square

4.3 Training the co-location count matrix \mathbf{Q}'

Training \mathbf{Q}' Next, we explain how to train the co-location count matrix $\mathbf{Q}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}|}$ in detail.

We first calculate the co-location count matrix $\mathbf{Q} \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}|}$, which includes the number of co-locations for each time instant and each location, from the training traces. Here, to upper bound the global sensitivity in DP, we introduce an upper limit $c \in \mathbb{Z}_{\geq 0}$ on the number of co-locations per user. In other words, if the number of co-locations reaches c , then the user's co-locations are not read anymore. This technique is called trimming in DP [26]. Figure 5 shows an example

² We also note that even if we use unbounded node DP in [15], Theorem 1 holds. This can be explained as follows. Let \mathcal{A} be a set of all possible adjacency matrices of any size larger than or equal to 1. Let $f : \mathcal{A} \rightarrow [0, 1]$ be a function that takes a training adjacency matrix $\mathbf{A} \in \mathcal{A}$ as input and outputs the friendship probability $p \in [0, 1]$. Then, if $n \geq 3$, we have:

$$\begin{aligned} |f(\mathbf{A}) - f(\mathbf{A}')| &\leq \frac{d}{n(n-1)} - \frac{d-2(n-1)}{(n-1)(n-2)} \\ &\leq \frac{d}{n(n-1)} - \frac{d-2(n-1)}{n(n-1)} \leq \frac{2(n-1)}{n(n-1)} = \frac{2}{n}. \end{aligned} \tag{6}$$

If $n = 2$, then $|f(\mathbf{A}) - f(\mathbf{A}')| \leq |\frac{2}{2} - 0| = 1$. If $n = 1$, then $|f(\mathbf{A}) - f(\mathbf{A}')| = 0$. Thus, adding $\text{Lap}\left(\frac{2}{n\epsilon_1}\right)$ to p provides ϵ_1 -node DP.

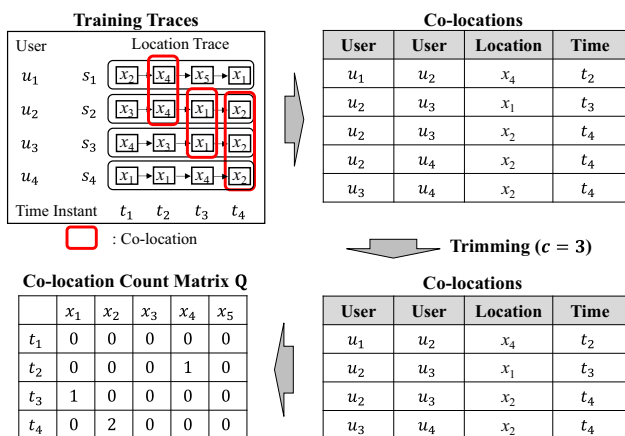


Fig. 5 Overview of calculating the co-location count matrix Q

of training Q in the case where $c = 3$. In this example, the co-location of users u_2 and u_4 are not read, because three co-locations of u_2 have already been read.

After calculating Q , we add noise to Q to obtain Q' . To add noise to Q , we use the Laplace mechanism or apply Privelet (for one-dimensional nominal data) [49] to each row of Q . The Laplace mechanism simply adds $Lap(\frac{c}{\epsilon_2})$ to each element of Q . Privelet performs the wavelet transform to a tree structure of locations. Then it adds the Laplace noise to a wavelet coefficient for each node in the tree.

For more details of the algorithm of Privelet, see Appendix 1. DP of Q' Let $\mathcal{M}_2^{Lap} : \mathcal{P}(\mathcal{R}) \rightarrow \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}'|}$ be the Laplace mechanism, which takes training traces $\mathcal{S} \subseteq \mathcal{R}$ as input and outputs $Q' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}'|}$ by adding $Lap(\frac{c}{\epsilon_2})$ to each element of Q . \mathcal{M}_2^{Lap} has the following privacy guarantee.

Theorem 2 \mathcal{M}_2^{Lap} provides ϵ_2 -user-level DP.

Proof By the trimming, we read at most c co-locations per user from \mathcal{S} . Thus, changing the entire trace of one user in \mathcal{S} will change each element of Q by at most c . Therefore, the global sensitivity of the co-location count in each element of Q is at most c . Since \mathcal{M}_2^{Lap} adds $Lap(\frac{c}{\epsilon_2})$ to each element of Q , it provides ϵ_2 -user-level DP. \square

Let $\mathcal{M}_2^{Privelet} : \mathcal{P}(\mathcal{R}) \rightarrow \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}'|}$ be Privelet. As with the Laplace mechanism, $\mathcal{M}_2^{Privelet}$ adds the Laplace noise based on the global sensitivity. Thus, $\mathcal{M}_2^{Privelet}$ has the following privacy guarantee:

Theorem 3 $\mathcal{M}_2^{Privelet}$ provides ϵ_2 -user-level DP.

See Appendix 1 for the proof.

4.4 Generating synthetic traces

Figure 6 shows the generation of synthetic traces using our location synthesizer. After training the friendship probabil-

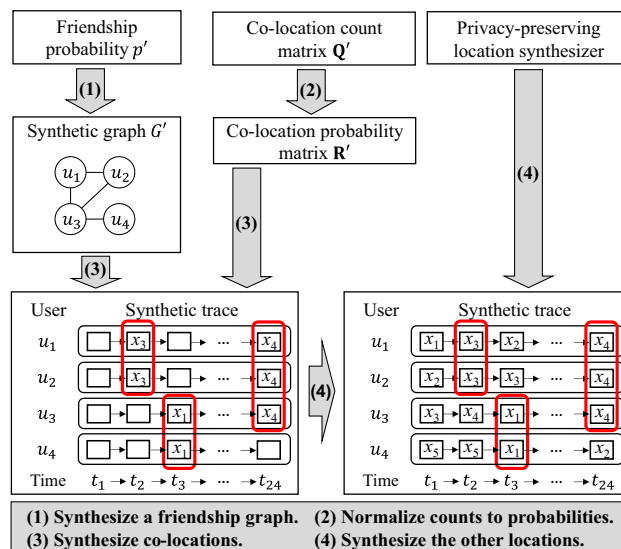


Fig. 6 Overview of generating synthetic traces in our proposed method

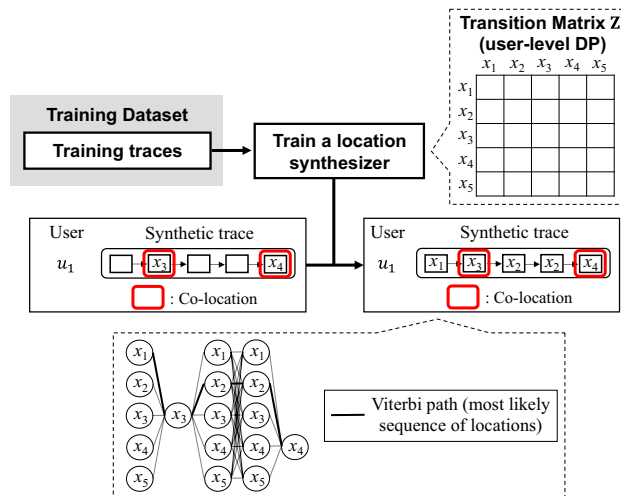


Fig. 7 Example of complementing locations using the synthesizer in [4]. We train a transition matrix providing user-level DP from training traces. Then we complement locations using the Viterbi algorithm, which finds the most likely sequence of locations, i.e., Viterbi path

ity p' and the co-location count matrix Q' , our synthesizer generates a synthetic trace for each of n users as follows.

1. Generate a *synthetic graph* G' (the ER or BA graph) with n nodes from p' . We explain how to generate the ER graph and the BA graph in detail at the end of Sect. 4.4.
2. Calculate a matrix called a *co-location probability matrix* $R' \in [0, 1]^{|\mathcal{T}| \times |\mathcal{X}'|}$ from Q' . Specifically, we calculate R' by normalizing each row of Q' so that the sum of the rows is 1. Here, we add the absolute value of the minimum value in Q' to all elements so that each element in Q' (hence R') does not have a negative value.

Input: Friendship probability $p' \in [0, 1]$, co-location count matrix $\mathbf{Q}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}| \times |\mathcal{X}|}$, transition probability matrix $\mathbf{Z} \in [0, 1]^{|\mathcal{X}| \times |\mathcal{X}|}$ providing ϵ_3 -user-level DP, #users $n \in \mathbb{N}$, #co-locations $\theta \in \mathbb{N}$

Output: Synthetic traces $\mathcal{S}_{syn} \subseteq \mathcal{U} \times \mathcal{X} \times \mathcal{T}$.

```

/* (1) Synthesize a friendship graph */
1  $G' \leftarrow \text{GenerateGraph}(n, p')$ ;
/* (2) Normalize counts to probabilities */
2  $\mathbf{Q}'_{min} \leftarrow \min\{\mathbf{Q}'_{ij} | i \in [|\mathcal{T}|], j \in [|\mathcal{X}|]\}$ ;
3 foreach  $i \in [|\mathcal{T}|]$  do
4   foreach  $j \in [|\mathcal{X}|]$  do
5      $\mathbf{R}'_{ij} \leftarrow (\mathbf{Q}'_{ij} + \mathbf{Q}'_{min}) / \sum_{k=1}^{|\mathcal{X}|} (\mathbf{Q}'_{ik} + \mathbf{Q}'_{min})$ ;
6   end
7 end
/* (3) Synthesize co-locations */
8  $\mathcal{S}_{syn} \leftarrow \emptyset$ ;
9 foreach  $\tau \in [\theta]$  do
10  Randomly select a user-pair  $(u_i, u_j)$  from  $G'$ ;
11  Randomly select a time instant  $t_l$  from  $\mathcal{T}$ ;
12  if  $\exists x_k \in \mathcal{X}, (u_i, x_k, t_l) \in \mathcal{S}_{syn}$  then
13     $\mathcal{S}_{syn} \leftarrow \mathcal{S}_{syn} \cup \{(u_j, x_k, t_l)\}$ ;
14  end
15  else if  $\exists x_k \in \mathcal{X}, (u_j, x_k, t_l) \in \mathcal{S}_{syn}$  then
16     $\mathcal{S}_{syn} \leftarrow \mathcal{S}_{syn} \cup \{(u_i, x_k, t_l)\}$ ;
17  end
18  else
19    Randomly generate a co-location  $x_k$  from  $\mathbf{R}'_j$ ;
20     $\mathcal{S}_{syn} \leftarrow \mathcal{S}_{syn} \cup \{(u_i, x_k, t_l), (u_j, x_k, t_l)\}$ ;
21  end
22 end
/* (4) Synthesize the other locations */
23  $\mathcal{S}_{syn} \leftarrow \text{ViterbiAlgorithm}(\mathcal{S}_{syn}, \mathbf{Z})$ ;
24 return  $\mathcal{S}_{syn}$ 

```

Algorithm 1: Generating synthetic traces in our proposed method. Here, we represent synthetic traces \mathcal{S}_{syn} as a set of triplets (u_i, x_k, t_l) of user u_i , location x_k , and time instant t_l for ease of presentation.

- Synthesize $\theta \in \mathbb{N}$ co-locations of friends from G' and \mathbf{R}' . Specifically, we iterate the following three steps until we obtain θ co-locations: (i) randomly select a pair of friends from G' ; (ii) randomly select a time instant from \mathcal{T} ; (iii) randomly generate a co-location at the selected time instant using the corresponding row of \mathbf{R}' . In step (iii), if one of the two users has already had a co-location at the selected time instant, we use it as a co-location for consistency with the previously generated co-location.
- Synthesize the other locations in n synthetic traces using a transition matrix of the existing DP location synthesizer [4, 13, 16] based on the Markov chain model. Specifically, we complement the remaining locations using the Viterbi algorithm [42].

Fig. 7 shows an example of complementing the remaining locations using the synthesizer in [4]. In the case of location traces, the synthesizer in [4] trains a transition probability matrix $\mathbf{Z} \in [0, 1]^{|\mathcal{X}| \times |\mathcal{X}|}$ common to all users from training traces and adds the Laplace noise to each

element to provide ϵ_3 -user-level DP (see [29] for details). Based on the trained matrix, we complement the remaining locations using the Viterbi algorithm [42], which finds the most likely sequence of locations, i.e., Viterbi path. In our experiments, we use the synthesizer in [4] to complement the remaining locations.

The number θ of co-locations is a parameter in our location synthesizer. In our experiments, we set θ to various values. It is also possible to calculate the frequency of co-locations with DP noise from training traces and set θ on based on the noisy co-location frequency.

Algorithm 1 shows the proposed algorithm when we use the synthesizer in [4] to complement locations other than co-locations. Line 1, lines 2 to 7, lines 8 to 22, and line 23 correspond to steps (1), (2), (3), and (4), respectively, in Fig. 6. In Appendix 1, we explain the `ViterbiAlgorithm` function (line 23) in detail. The `GenerateGraph` function (line 1) takes the number n of users and the noisy friendship probability p' as input and outputs a synthetic graph G' . The ER model (with parameters n and p') and the BA model (with parameters n and $\frac{p'(n-1)}{2}$) satisfy this requirement³, as explained below.

Generating the ER graph The Erdős-Rényi (ER) model [5] has two parameters $n \in \mathbb{N}$ and $q \in [0, 1]$ and is denoted by $\mathcal{G}(n, q)$. The ER model $\mathcal{G}(n, q)$ is a simple graph generation model that randomly and independently generates each edge between n nodes with probability q . Since the friendship probability p' represents the probability that two users are friends, we set $q = p'$. In other words, we generate a synthetic graph G' based on $\mathcal{G}(n, p')$. Note that G' is generated using only a single friendship probability p' .

Recall that we calculate p' by $p' = p + \text{Lap}(\frac{2}{n\epsilon_1})$. Since the expectation of $\text{Lap}(\frac{2}{n\epsilon_1})$ is 0, p' is an unbiased estimate of p . Thus, the expected number of edges in G' is equal to the number of edges in the training graph. In other words, our ER graph preserves the average degree (number of friends) of the training graph.

However, the ER model does not have a power-law degree distribution [1]. Therefore, our ER graph does not reflect an actual graph property well.

Generating the BA graph The Barabási-Albert (BA) model [1] is a graph generation model that has a power-law degree distribution. It has two parameter $n \in \mathbb{N}$ and $\lambda \in \mathbb{Z}_{\geq 0}$ is denoted by $\mathcal{B}(n, \lambda)$. The BA model $\mathcal{B}(n, \lambda)$ generates a graph with n nodes by sequentially attaching new nodes so that each new node is connected to λ existing nodes. An edge is connected to an existing node with the probability proportional to its degree.

³ We can easily generate a synthetic graph G' based on the ER or BA model by using NetworkX [14], a Python library for graph analysis.

The BA graph with parameter λ has about $n\lambda$ edges. In contrast, the training graph has $\frac{pn(n-1)}{2}$ edges. These numbers coincide when $\lambda = \frac{p(n-1)}{2}$. Therefore, we set $\lambda = \frac{p'(n-1)}{2}$ (we round decimals) and synthesize G' based on the BA model $\mathcal{B}(n, \frac{p'(n-1)}{2})$. Again, note that G' is generated using only a single friendship probability p' .

Since p' is an unbiased estimate of p , our BA graph preserves the average degree of the training graph. In addition, our BA graph has a power-law degree distribution. In our experiments, we show how well our BA graph preserves these statistical properties of the training graph.

DP of the ER and BA graphs Let $\mathcal{M}_1^{\text{ER}}$ (resp. $\mathcal{M}_1^{\text{BA}}$): $\{0, 1\}^{n \times n} \rightarrow \{0, 1\}^{n \times n}$ be a randomized mechanism that takes a training adjacency matrix \mathbf{A} as input and outputs an adjacency matrix of the ER (resp. BA) graph G' . Then, we have the following privacy guarantees.

Theorem 4 Both $\mathcal{M}_1^{\text{ER}}$ and $\mathcal{M}_1^{\text{BA}}$ provide ϵ_1 -node DP.

Proof By Theorem 1, the randomized mechanism $\mathcal{M}_1^{\text{Lap}}$ that takes \mathbf{A} as input and outputs p' provides ϵ_1 -node DP. In addition, both the ER and BA graphs are generated using only a single friendship probability p' , as explained above. Thus, by the immunity to the post-processing [10], both $\mathcal{M}_1^{\text{ER}}$ and $\mathcal{M}_1^{\text{BA}}$ provide ϵ_1 -node DP. \square

Scalability As shown in Fig. 6, our proposed method consists of the following steps: (1) synthesize a friendship graph, (2) normalize counts to probabilities, (3) synthesize co-locations, and (4) synthesize the other locations. The time complexity of steps (1), (2), (3), (4) is $O(n^2)$, $O(|\mathcal{X}|^2)$, $O(\theta(|G'| + |\mathcal{T}| + |\mathcal{X}|))$, and $O(n|\mathcal{T}||\mathcal{X}|^2)$, respectively, where G' is a synthetic graph. In addition, the time complexity of calculating p' and \mathbf{Q}' from the training dataset is $O(|G|)$ and $O(n|\mathcal{T}| + |\mathcal{X}|^2)$, respectively, where G is a training graph. Note that most training graphs are sparse, and $|G|, |G'| \ll n^2$ in that case. Thus, the time complexity of our proposed method can be expressed as $O(n^2 + \theta(|G'| + |\mathcal{T}| + |\mathcal{X}|) + n|\mathcal{T}||\mathcal{X}|^2)$ in total. The factor of $n|\mathcal{T}||\mathcal{X}|^2$ is caused by the Viterbi algorithm in step (4). Although the Viterbi algorithm is known as an efficient algorithm, the run time might still be large when n and $|\mathcal{X}|$ are extremely large. We can improve the run time by, e.g., parallel implementation [11].

4.5 End-to-end privacy analysis

Below, we provide end-to-end privacy analysis of our location synthesizer. Let $\mathcal{S}_* \subseteq \mathcal{R}$ be our synthetic traces. Let $\mathcal{M}_* : \{0, 1\}^{n \times n} \times \mathcal{P}(\mathcal{R}) \rightarrow \mathcal{P}(\mathcal{R})$ be our location synthesizer that takes the dataset $(\mathbf{A}, \mathcal{S})$ as input and outputs \mathcal{S}_* . In addition, let \mathcal{M}_3 be a training algorithm that takes \mathcal{S} as input and outputs a generative model (transition matrix) of

the existing synthesizer providing ϵ_3 -user-level DP [4, 13, 16].

Our synthesizer \mathcal{M}_* uses $\mathcal{M}_1^{\text{ER}}$ or $\mathcal{M}_1^{\text{BA}}$ to generate a synthetic graph G' from \mathbf{A} . Then, \mathcal{M}_* uses $\mathcal{M}_2^{\text{Lap}}$ or $\mathcal{M}_2^{\text{Privelet}}$ to output a co-location count matrix \mathbf{Q}' from \mathcal{S} . Finally, \mathcal{M}_* generates synthetic traces using G', \mathbf{Q}' , and a transition matrix output by \mathcal{M}_3 .

Then, we have the following privacy guarantees:

Theorem 5 The composition $(\mathcal{M}_2^{\text{Lap}}, \mathcal{M}_3)$ or $(\mathcal{M}_2^{\text{Privelet}}, \mathcal{M}_3)$ provides $(\epsilon_2 + \epsilon_3)$ -user-level DP.

Proof By Theorems 2 and 3, both $\mathcal{M}_2^{\text{Lap}}$ and $\mathcal{M}_2^{\text{Privelet}}$ provide ϵ_2 -user-level DP. In addition, \mathcal{M}_3 provides ϵ_3 -user-level DP. Thus, by the composition theorem [10], $(\mathcal{M}_2^{\text{Lap}}, \mathcal{M}_3)$ or $(\mathcal{M}_2^{\text{Privelet}}, \mathcal{M}_3)$ provides $(\epsilon_2 + \epsilon_3)$ -user-level DP. \square

Theorem 6 Our location synthesizer \mathcal{M}_* provides $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ -total DP.

Proof As explained above, our synthesizer \mathcal{M}_* generates synthetic traces using the outputs of the three mechanisms: (i) $\mathcal{M}_1^{\text{ER}}$ or $\mathcal{M}_1^{\text{BA}}$, (ii) $\mathcal{M}_2^{\text{Lap}}$ or $\mathcal{M}_2^{\text{Privelet}}$, and (iii) \mathcal{M}_3 . The first mechanism (i) provides ϵ_1 -node DP (Theorem 4). The composition of the second and third mechanisms (ii) and (iii) provides $(\epsilon_2 + \epsilon_3)$ -user-level DP (Theorem 5). Then, by Proposition 1, the composition of the three mechanisms (i), (ii), and (iii) provides $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ -total DP.

Our synthesizer \mathcal{M}_* generates synthetic traces based on the post-processing on the outputs of (i), (ii), and (iii). Thus, by the immunity to the post-processing [10], \mathcal{M}_* also provides $(\epsilon_1 + \epsilon_2 + \epsilon_3)$ -total DP. \square

Theorem 6 guarantees the overall privacy of our location synthesizer. Note that the existing synthesizer [4, 13, 16] provides ϵ_3 -user-level DP hence ϵ_3 -total DP. As shown in Table 1, our synthesizer uses additional privacy budgets ϵ_1 and ϵ_2 to incorporate new information (i.e., co-locations) into synthetic traces. In Sect. 5, we show that ϵ_1 and ϵ_2 are small, e.g., $\epsilon_1 = 0.2$ and $\epsilon_2 = 1$.

5 Experimental evaluation

We evaluated our location synthesizer to show its effectiveness. In Sect. 5.1, we explain datasets used in our experiments. In Sect. 5.2, we describe utility metrics. In Sect. 5.3, we explain location synthesizers evaluated in our experiments. In Sect. 5.4, we report experimental results for parameters in our location synthesizer. In Sect. 5.5, we report results of comparison experiments. In Sect. 5.6, we summarize the experimental results.

Table 3 POI categories and sub-categories (Foursquare)

POI category	POI sub-category
Travel & Transport	train station, airport, platform subway, airport terminal
Shop & service	Electronics store, hobby shop record shop, mall
Arts & entertainment	Arcade
Professional & Other places	tech startup, convention center

Table 4 POI categories and sub-categories (Gowalla)

POI category	POI sub-category
Community	Apartment, condo, modern, corporate office Skyscraper
Outdoors	Historic landmark, tower
Shopping	Bookstore, discount store, drugstore & pharmacy other-shopping, technology
Travel	Airport, subway, train station

5.1 Datasets

In our experiments, we used the Foursquare dataset [50] and the Gowalla dataset [25] (denoted by Foursquare and Gowalla, respectively). Both datasets include the users' friendship data (i.e., training graph) on SNS and categories/sub-categories of POIs [7]. For our experiments, we used the Tokyo check-in data in each dataset. The Foursquare dataset contained 916,136 check-ins, 8357 users, and 83,647 POIs in Tokyo. The Gowalla dataset contained 184,354 check-ins, 2434 users, and 17,866 POIs in Tokyo. We set the length of a time instant to one hour and extracted two temporally-continuous location events from the dataset ($|\mathcal{T}| = 24$).

In both datasets, check-ins are concentrated in some POIs. Thus, the matrix \mathbf{Q} becomes extremely sparse when using all POIs. Hence, we used check-in data for 100 POIs whose check-in counts are the largest. In this case, $|\mathcal{X}| = 100$ and the number n of users was $n = 8357$ in the Foursquare dataset and $n = 1463$ in the Gowalla dataset.

The categories and sub-categories of POIs in each dataset are shown in Table 3. The number m of categories was 4 in both datasets. The total number of co-location events in the traces is 2012 (resp. 51) in Foursquare (resp. Gowalla). Gowalla has much fewer co-location events than Foursquare.

5.2 Utility metrics

Co-locations First, we evaluated the utility of our two parameters – the friendship probability p' and the co-location count matrix \mathbf{Q}' – to quantitatively show how our location synthesizer preserves the information about co-locations. For p' ,

we evaluated the absolute error $|p - p'|$ between p and p' as a utility metric. We denote the absolute error of p' by $\text{AE}_{p'}$.

For \mathbf{Q}' , co-location counts for each POI category and each time instant (e.g., “travel & transport” from 7AM to 9AM) are particularly important. Thus, we evaluated the utility for each POI category and each time instant. Specifically, let $\mathbf{Q}^* \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}|}$ be a co-location count matrix before adding noise when we do not perform trimming. \mathbf{Q}^* is identical to \mathbf{Q} when $c = \infty$. We calculated a *per-category* co-location count matrix $\overline{\mathbf{Q}}^* \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times m}$ ($|\mathcal{T}| = 24$, $m = 4$), which is composed of counts for each time instant and each POI category, by summing up counts in \mathbf{Q}^* for each POI category. $\overline{\mathbf{Q}}^*$ is obtained by counting co-locations for each POI category and each time instant in the training traces. Similarly, we calculated a *per-category* co-location count matrix $\overline{\mathbf{Q}}' \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times m}$ by summing up counts in \mathbf{Q}' for each POI category.

Then we evaluated the mean absolute error (MAE) and the mean square error (MSE) between $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$. The MAE is given by $\frac{1}{|\mathcal{T}|m} \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^m |\overline{\mathbf{Q}}_{ij}^* - \overline{\mathbf{Q}}'_{ij}|$. The MSE is given by $\frac{1}{|\mathcal{T}|m} \sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^m (\overline{\mathbf{Q}}_{ij}^* - \overline{\mathbf{Q}}'_{ij})^2$. Note that the difference between $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$ can be caused by two factors: trimming and DP noise. We denote the MAE and MSE of $\overline{\mathbf{Q}}'$ by $\text{MAE}_{\mathbf{Q}}$ and $\text{MSE}_{\mathbf{Q}}$, respectively.

Note that our location synthesizer normalizes counts in \mathbf{Q}' to probabilities and synthesizes co-locations based on the co-location probability matrix \mathbf{R}' . Therefore, we also normalized counts in $\overline{\mathbf{Q}}^*$ and $\overline{\mathbf{Q}}'$ to probabilities and evaluated the difference between them. Specifically, let $\overline{\mathbf{R}}^*$ (resp. $\overline{\mathbf{R}}'$) $\in [0, 1]^{|\mathcal{T}| \times m}$ be a *per-category* co-location probability matrix such that $\overline{\mathbf{R}}_{ij}^* = \frac{\overline{\mathbf{Q}}_{ij}^*}{\sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^m \overline{\mathbf{Q}}_{ij}^*}$ (resp. $\overline{\mathbf{R}}'_{ij} = \frac{\overline{\mathbf{Q}}'_{ij}}{\sum_{i=1}^{|\mathcal{T}|} \sum_{j=1}^m \overline{\mathbf{Q}}'_{ij}}$). In both $\overline{\mathbf{R}}^*$ and $\overline{\mathbf{R}}'$, the sum of all elements is 1. We evaluated the MAE and MSE between $\overline{\mathbf{R}}^*$ and $\overline{\mathbf{R}}'$.

Because both $\overline{\mathbf{R}}^*$ and $\overline{\mathbf{R}}'$ are probability distributions, we also evaluated the JS divergence between $\overline{\mathbf{R}}^*$ and $\overline{\mathbf{R}}'$. The KL divergence and the JS divergence are popular measures of the distance between two probability distributions. We did not evaluate the KL divergence between $\overline{\mathbf{R}}^*$ and $\overline{\mathbf{R}}'$, because the number of co-locations was too small and the KL divergence could be infinity in this case.

Formally, for $d \in \mathbb{N}$, let Δ_d be the d -probability simplex. Let $\mathbf{z}, \mathbf{z}' \in \Delta_d$ be the two probability distributions. Then, the KL divergence is given by $D_{KL}(\mathbf{z}||\mathbf{z}') = \sum_{i=1}^d \mathbf{z}_i \log \frac{\mathbf{z}_i}{\mathbf{z}'_i}$, where \mathbf{z}_i and \mathbf{z}'_i are the i -th elements of \mathbf{z} and \mathbf{z}' , respectively. The JS divergence is given by $D_{JS}(\mathbf{z}||\mathbf{z}') = \frac{1}{2} D_{KL}(\mathbf{z}||\mathbf{m}) + \frac{1}{2} D_{KL}(\mathbf{z}'||\mathbf{m})$, where $\mathbf{m} = \frac{1}{2}(\mathbf{z} + \mathbf{z}')$. We denote the MAE, MSE, and JS of $\overline{\mathbf{R}}'$ by $\text{MAE}_{\mathbf{R}}$, $\text{MSE}_{\mathbf{R}}$, and $\text{JS}_{\mathbf{R}}$, respectively.

Furthermore, we selected top-10 co-location events (i.e., ten pairs of time instants and POI categories) whose counts in the training traces are the largest, i.e., top-10 elements in $\overline{\mathbf{Q}}^*$. Then, we visualized the values (counts or probabilities)

of the top-10 events in $\bar{\mathbf{Q}}^*$, $\bar{\mathbf{Q}}'$, $\bar{\mathbf{R}}^*$, and $\bar{\mathbf{R}}'$. Formally, let $\Omega \subset [|\mathcal{T}|] \times [m]$ ($|\Omega| = 10, |\mathcal{T}| = 24, m = 4$) be the set of the top-10 events. We visualized the values of $\bar{\mathbf{Q}}_{ij}^*$, $\bar{\mathbf{Q}}'_{ij}$, $\bar{\mathbf{R}}_{ij}^*$, and $\bar{\mathbf{R}}'_{ij}$ for $(i, j) \in \Omega$.

Our location synthesizer normalizes \mathbf{Q}' to \mathbf{R}' and randomly generates co-locations in synthetic traces based on p' and \mathbf{R}' . Therefore, if the absolute error of p' is small, our synthetic traces preserve the information about how likely two users will be friends. If the MAE, MSE, and JS of $\bar{\mathbf{R}}'$ are small, our synthetic traces preserve the information about how likely a co-location of friends will happen at a certain POI category for each time instant, e.g., “travel & transport” from 7 to 9AM.

It is shown in [50] that there is a correlation between co-locations and friendships on Twitter. Thus, if the MAE, MSE, and JS of $\bar{\mathbf{R}}'$ are small, then a location-based friend suggestion algorithm developed based on the synthetic location data would also be useful for real location data.

Other statistical features Next, we evaluated how our synthetic traces preserve other statistical features about training traces. Specifically, we calculated two basic statistical features: *population distribution* and *transition probability matrix*. The population distribution ($|\mathcal{X}|$ -dimensional probability vector) is a key feature to find popular POIs [54]. The transition probability matrix ($|\mathcal{X}| \times |\mathcal{X}|$ matrix) is a key feature to model user movement patterns [47]. We calculated these statistical features from the training traces and the synthetic traces.

Formally, let \mathbf{r} (resp. $\mathbf{r}' \in \Delta_{|\mathcal{X}|}$) be a population distribution calculated from the training trace (resp. synthetic traces). For example, $\mathbf{r} = (\frac{5}{24}, \frac{1}{4}, \frac{1}{6}, \frac{1}{4}, \frac{1}{8})$ in Fig. 2. In the transition probability matrix, each row of the matrix represents a probability distribution. Let \mathbf{M} (resp. $\mathbf{M}' \in [0, 1]^{|\mathcal{X}| \times |\mathcal{X}|}$) be the transition probability matrix calculated from the training traces (resp. synthetic traces). For $i \in |\mathcal{X}|$, let \mathbf{M}_i (resp. $\mathbf{M}'_i \in \Delta_{|\mathcal{X}|}$) be the i -th row of \mathbf{M} (resp. \mathbf{M}'). For example, $\mathbf{M}_1 = (\frac{1}{4}, \frac{1}{2}, 0, \frac{1}{4}, 0)$ in Fig. 2.

For each statistical feature, we evaluated the distance between the synthetic traces and the training traces. We adopted the MAE, MSE, KL divergence, and JS divergence as distance measures. Here, we evaluated the KL divergence because the number of locations is large (unlike co-locations).

For the transition probability matrix, each row represents a probability distribution. Thus, we evaluated the weighted average of the KL/JS divergence, where we used a stationary distribution calculated from the matrix as a weight vector. We denote the MAE/MSE/KL/JS of \mathbf{r}' (resp. \mathbf{M}') by $\text{MAE}_r/\text{MSE}_r/\text{KL}_r/\text{JS}_r$ (resp. $\text{MAE}_M/\text{MSE}_M/\text{KL}_M/\text{JS}_M$).

Downstream tasks, such as finding popular POIs [54] and predicting the next POI [47], are based on the population distribution and the transition matrix. For example, popular

Table 5 Utility metrics in our experiments

Utility metrics	Notations
Absolute error of the friendship probability p'	$\text{AE}_{p'}$
MAE/MSE of the per-category co-location count matrix $\bar{\mathbf{Q}}'$	$\text{MAE}_{\mathbf{Q}}/\text{MSE}_{\mathbf{Q}}$
MAE/MSE/JS of the per-category co-location probability matrix $\bar{\mathbf{R}}'$	$\text{MAE}_{\mathbf{R}}/\text{MSE}_{\mathbf{R}}/\text{JS}_{\mathbf{R}}$
MAE/MSE/KL/JS of the population distribution \mathbf{r}'	$\text{MAE}_r/\text{MSE}_r/\text{KL}_r/\text{JS}_r$
MAE/MSE/KL/JS of the transition probability matrix \mathbf{M}'	$\text{MAE}_M/\text{MSE}_M/\text{KL}_M/\text{JS}_M$

POIs can be obtained by selecting locations whose values in the population distribution are the largest. Given a specific POI, the next POI can be predicted by selecting a POI whose probability in the transition matrix is the largest. Thus, if the distance measures (MAE, MSE, KL, and JS) of the population distribution (resp. transition matrix) are small, then the synthetic data would be useful for finding popular POIs (resp. predicting the next POI).

Table 5 summarizes the utility metrics and their notations in our experiments.

5.3 Location synthesizers

In our experiments, we evaluated three location synthesizers for comparison. The first synthesizer is to independently and randomly generates a location at each time instant from a uniform distribution. We call this simple method Uniform.

The second synthesizer is the synthetic data generator proposed in [4]. This synthesizer can be applied to any data, including location traces [29]. Following [29], we applied this synthesizer to location traces as follows. First, we trained a transition probability matrix ($|\mathcal{X}| \times |\mathcal{X}|$ matrix) common to all users from training traces and added the Laplace noise $\text{Lap}(\frac{c}{\epsilon_3})$ to each element. Adding the Laplace noise provides ϵ_3 -user-level DP. Then, we randomly generated the first location based on the stationary distribution and then generated the remaining locations using the transition matrix. Because this method is designed on the basis of the transition probability matrix, we call it TPM.

The third synthesizer is our proposed synthesizer. We call it Proposal. In Proposal, we trained p' from a training graph by adding the Laplace noise, and \mathbf{Q}' from training traces using the Laplace mechanism or Privelet. Then we generated θ co-locations using p' and \mathbf{Q}' . Finally, we generated the remaining locations using TPM, which provides ϵ_3 -user-level DP as explained above.

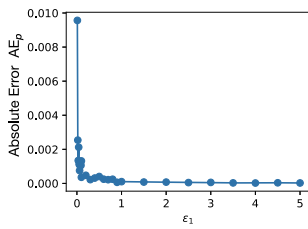


Fig. 8 Absolute error of p' versus ϵ_1 in Foursquare

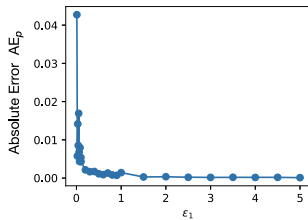


Fig. 9 Absolute error of p' versus ϵ_1 in Gowalla

Note that besides TPM [4], there are other existing location synthesizers such as [13, 16]. We evaluated TPM for two reasons. First, TPM is easy to implement. Second, all of the existing synthesizers [4, 13, 16] do not introduce the concept of “friends” and therefore lack the utility of co-locations. In other words, they have the same values for the utility of co-locations. In our experiments, we quantitatively show that TPM lacks the utility of co-locations, which means that the synthesizers in [13, 16] also lack the utility of co-locations. We leave synthesizing locations other than co-locations in Proposal using the synthesizers in [13, 16] for future work.

In each of Uniform, TPM, and Proposal, the length of a time instant was set to be one hour, and a trace with the length of one day was generated for each of n users. For each synthesizer, synthetic traces were generated five times, and the utility metrics were averaged over the five runs to stabilize the performance.

5.4 Experimental results for parameters in our location synthesizer

First, we evaluated how well parameters of our synthesizer (Proposal) preserve the information about co-locations.

Friendship probability p' In Figs. 8 and 9, we show the absolute error of p' when we changed the privacy budget ϵ_1 from 0.01 to 5 in Foursquare and Gowalla.

It is seen from Fig. 8 that the absolute error rapidly decreases as ϵ_1 increases from 0.01 to 0.5. Note that the absolute error depends only on the Laplace noise, as $p' = p + \text{Lap}(\frac{2}{n\epsilon_1})$. This explains the reason that the absolute error decreases as the value of ϵ increases. It is also seen from Fig. 8 that the absolute error is extremely small and almost equals to 0 after $\epsilon_1 = 0.2$. This result demonstrates that we can accurately estimate the friendship probability p' with a

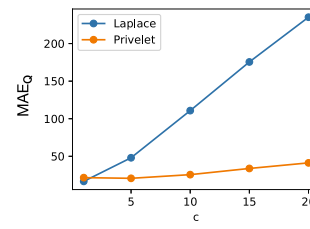


Fig. 10 MAE of \bar{Q}' versus c ($\epsilon_2 = 1$) in Foursquare

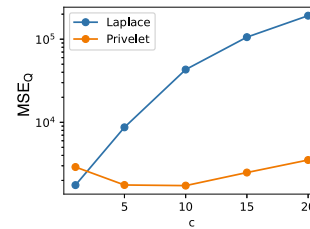


Fig. 11 MSE of \bar{Q}' versus c ($\epsilon_2 = 1$) in Foursquare

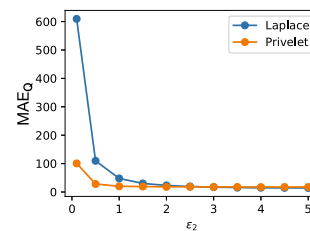


Fig. 12 MAE of \bar{Q}' versus ϵ_2 ($c = 5$) in Foursquare

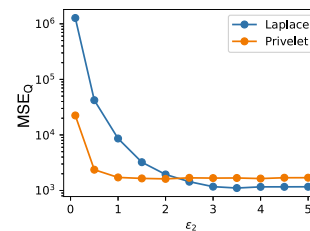


Fig. 13 MSE of \bar{Q}' versus ϵ_2 ($c = 5$) in Foursquare

small privacy budget $\epsilon_1 = 0.2$ in node DP for friendship data. We observe from Fig. 9 that the result of Gowalla is similar to that of FourSquare.

Per-category co-location count matrix \bar{Q}' The MAE and the MSE of \bar{Q}' in Foursquare are shown in Figs. 10 and 11. Here, we set the privacy budget ϵ_2 in user-level DP for training traces to $\epsilon_2 = 1$, and we set the upper limit c on the number of co-locations per user in trimming to $c = 1, 5, 10, 15, \text{ or } 20$.

It is seen from Figs. 10 and 11 that Privelet has much smaller MAE and MSE than the Laplace mechanism. This means that Privelet significantly reduces the amount of noise for each POI category and each time instant, e.g., subway in the morning. It is also seen that when $c = 5$ and 10, Privelet has the smallest MAE and MSE, respectively. These results

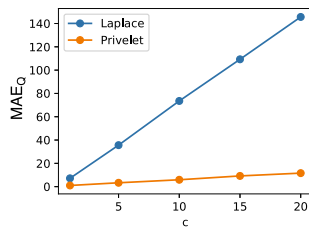


Fig. 14 MAE of \bar{Q}' versus c ($\epsilon_2 = 1$) in Gowalla

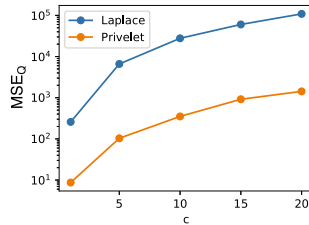


Fig. 15 MSE of \bar{Q}' versus c ($\epsilon_2 = 1$) in Gowalla

show that there is a trade-off between the effects of trimming and the Laplace noise; i.e., the effect of trimming is large when c is small, whereas the amount of the Laplace noise is large when c is large.

In Figs. 12 and 13, we also show the relationship between ϵ_2 and the MAE/MSE in Foursquare when we set c to $c = 5$. It is seen from these figures that the MAE and the MSE rapidly decrease as ϵ_2 increases from 0.1 to 1 and they remain almost unchanged after $\epsilon_2 = 1$.

It is seen from Fig. 13 that when ϵ_2 is 2.5 or more, the MSE of Privelet is larger than that of Laplace. One reason for this is that Privelet adds noise to each node of a tree structure and the number of nodes in Privelet is larger than that of elements in \mathbf{Q} .

The MAE and the MSE of \bar{Q}' in Gowalla are shown in Figs. 14, 15, 16, and 17. We set the privacy budget ϵ_2 and the upper limit c to the same value as Foursquare.

It is seen from Figs. 14 and 15 that Privelet has the smallest MAE and MSE when $c = 1$, unlike the results in Foursquare. This is because the number of co-locations in Gowalla was much smaller than that of Foursquare and was not affected by trimming even when $c = 1$.

It is also seen from Figs. 16 and 17 that the MAE and the MSE decrease significantly when we increase ϵ_2 from 0.1 to 1. In addition, when ϵ_2 is larger than 1, the MAE of Privelet is almost unchanged. Therefore, we can obtain an accurate co-location count matrix with a reasonable factor of $\epsilon_2 = 1$. *Top-10 co-location events* Table 6 shows the top-10 co-location events whose counts in the training traces are the largest, i.e., top-10 elements in \bar{Q}^* . For example, co-location events are likely to occur in “travel & transport” in the morning in Foursquare and “travel” in the morning or at night in Gowalla. Figs. 18 and 19 show the counts and probabilities (i.e., values of \bar{Q}_{ij}^* , \bar{Q}'_{ij} , \bar{R}_{ij}^* , and \bar{R}'_{ij}) of the top-10

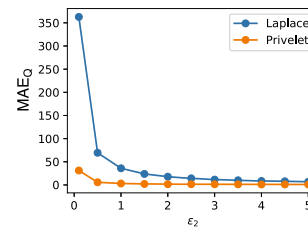


Fig. 16 MAE of \bar{Q}' versus ϵ_2 ($c = 5$) in Gowalla

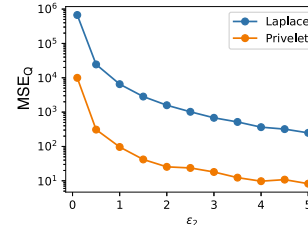


Fig. 17 MSE of \bar{Q}' versus ϵ_2 ($c = 5$) in Gowalla

co-location events. It is seen from these figures that count values in the Laplace mechanism are much larger than those in the training datasets. This is because we normalize each row of \mathbf{Q}' so that all elements are non-negative and the sum of the rows is 1. In other words, the normalization introduces a large positive bias. Privelet provides much smaller errors in counts, especially in Gowalla. This explains the reason that the MAE/MSE of Privelet is much smaller than that of the Laplace mechanism in Figs. 14, 15, 16, and 17.

Fig. 18b shows that Privelet preserves the probability information well in Foursquare. However, Fig. 19b shows that this is not the case with Gowalla. This is because Gowalla has a very small number of co-location events (only 51 events, as described in Sect. 5.1). In this case, almost all elements (83 out of 96 elements) of \bar{Q}^* are 0, and Privelet also assigns 0 to almost all elements in \bar{Q}' . In other words, Gowalla is much more challenging than Foursquare. Based on Figs. 18 and 19, we use Privelet in Foursquare and the Laplace mechanism in Gowalla in the following comparison experiments.

We emphasize that the existing synthesizers do not introduce a concept of “friends” in their model and therefore do not preserve any information about co-locations (as shown in Table 1). Our synthesizer adds the new information by training a friendship probability and co-location count matrix. Additional privacy budgets ϵ_1 and ϵ_2 are reasonably small, e.g., $\epsilon_1 = 0.2$ and $\epsilon_2 = 1$.

5.5 Results of comparison experiments

Three location synthesizers Next, we compare Proposal with two baselines: TPM and Uniform. We evaluated the utility of co-locations, population distributions, and transition matrices for each synthesizer. For the utility of co-locations, we evaluated the MAE, MSE, and JS divergence of \bar{R}' ,

Table 6 Top-10 co-location events (time instants and POI categories) in the training datasets

Rank	Foursquare	Gowalla
1	(9:00–9:59, travel & transport)	(23:00–23:59, travel)
2	(10:00–10:59, travel & transport)	(0:00–0:59, community)
3	(8:00–8:59, travel & transport)	(9:00–9:59, travel)
4	(7:00–7:59, travel & transport)	(0:00–0:59, travel)
5	(11:00–11:59, travel & transport)	(10:00–10:59, travel)
6	(12:00–12:59, travel & transport)	(11:00–11:59, travel)
7	(6:00–6:59, travel & transport)	(12:00–12:59, travel)
8	(5:00–5:59, travel & transport)	(2:00–2:59, travel)
9	(13:00–13:59, travel & transport)	(3:00–3:59, travel)
10	(4:00–4:59, travel & transport)	(4:00–4:59, shopping)

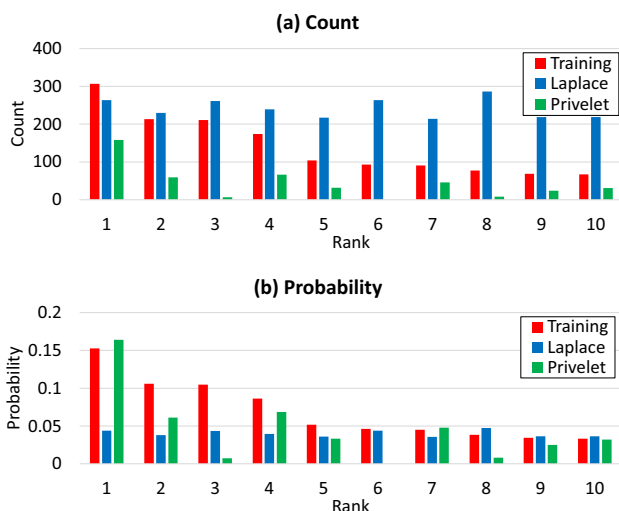


Fig. 18 Counts and probabilities of top-10 co-location events in Foursquare ($\epsilon_2 = 1, c = 5$). For counts, we show the values in \bar{Q}^* (Training) and \bar{Q}^{\prime} (Laplace/Privelet). For probabilities, we show the values in \bar{R}^* (Training) and \bar{R}^{\prime} (Laplace/Privelet)

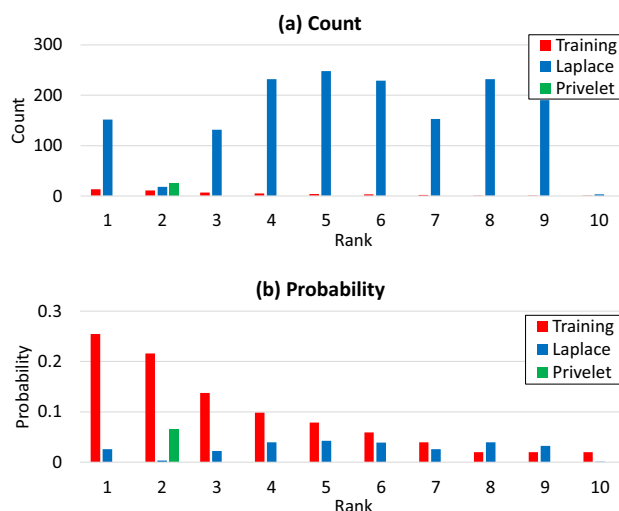


Fig. 19 Counts and probabilities of top-10 co-location events in Gowalla ($\epsilon_2 = 1, c = 5$). For counts, we show the values in \bar{Q}^* (Training) and \bar{Q}^{\prime} (Laplace/Privelet). For probabilities, we show the values in \bar{R}^* (Training) and \bar{R}^{\prime} (Laplace/Privelet)

as described in Sect. 5.2. In TPM and Uniform, we used a uniform distribution as \bar{R}^{\prime} because they had no concept of “friends.” For the utility of population distributions and transition matrices, we evaluated the MAE, MSE, KL divergence, and JS divergence. We used the ER or BA model for Proposal. We set $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1$ and $c = 5$. Although we set all the three privacy budgets to 1 for simplicity, we can also assign a smaller value to ϵ_1 without affecting the utility, e.g., $\epsilon_1 = 0.2$, as shown in Figs. 8 and 9. For the number θ of generated co-location events in Proposal, we set $\theta = 100$. We also report the relationship between θ and the utility in Appendix 1.

Figs. 20 and 21 show the results. Figures 20a and Fig. 21a show that Proposal significantly outperforms TPM and Uniform in terms of the utility of co-locations, which demonstrates the effectiveness of Proposal. In addition, Fig. 20b, c and Fig. 21b, c show that Proposal significantly outperforms

Uniform and has almost the same utility as TPM in terms of the utility of the population distribution and the transition matrix.

One exception is that both Proposal and TPM have larger MSE than Uniform in Gowalla (see Fig. 21b, c). One reason for this is that the check-in data in Gowalla includes many outliers who have unique transition patterns. Because the MSE squares the error, it is significantly affected by such outliers.

However, we emphasize that Proposal has smaller MAE, KL divergence, and JS divergence than Uniform, as shown in Fig. 21b, c. This result indicates that Proposal preserves the transition matrix well.

ER and BA graph models Finally, we compared the ER model with the BA model in Proposal. We first evaluated an average degree (i.e., the average number of friends) in the training graph, the ER graph, and the BA graph. Table 7 shows the

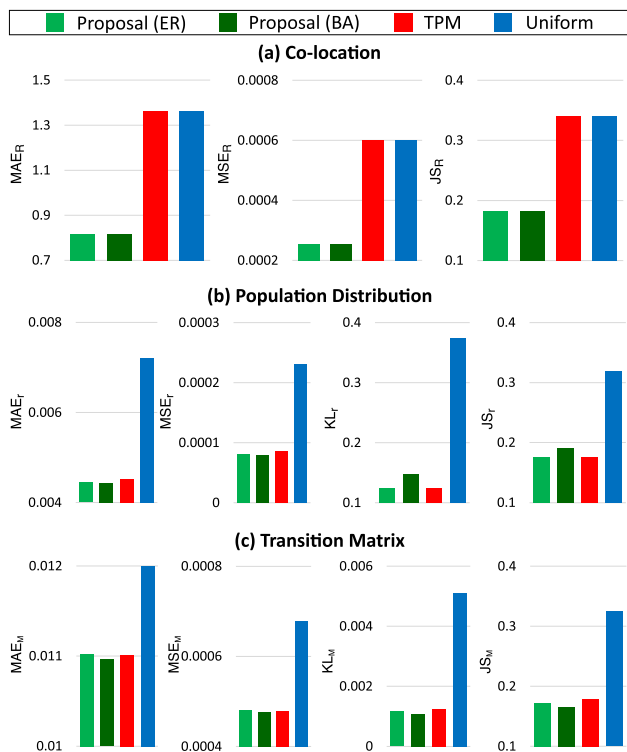


Fig. 20 Comparison results in Foursquare ($\epsilon_1 = \epsilon_2 = \epsilon_3 = 1$, $c = 5$, $\theta = 100$). **a** MAE/MSE/JS of $\bar{\mathbf{R}}'$, **b** MAE/MSE/KL/JS of \mathbf{r}' , **c** MAE/MSE/KL/JS of \mathbf{M}' . Smaller is better in all utility metrics. Proposal uses Privelet

Table 7 Average degree

	Training graph	ER graph	BA graph
Average degree	4.69	3.83	3.99

results. It is seen from this table that both the ER and BA graphs preserve the average degree well.

Next, we evaluated a degree distribution for each graph. Figures 22, 23, and 24 show the results. It is seen from these figures that the training graph has a power-law degree distribution. The ER graph does not preserve this property. In contrast, the BA graph has a power-law degree distribution and is very similar to the training graph. Therefore, the BA graph reproduces the friendship property of the original data more accurately than the ER graph.

5.6 Summary

In summary, through comprehensive evaluation using two real datasets and various utility metrics, we showed the following results.

- Proposal preserves the information about co-locations of friends, whereas existing location synthesizers such

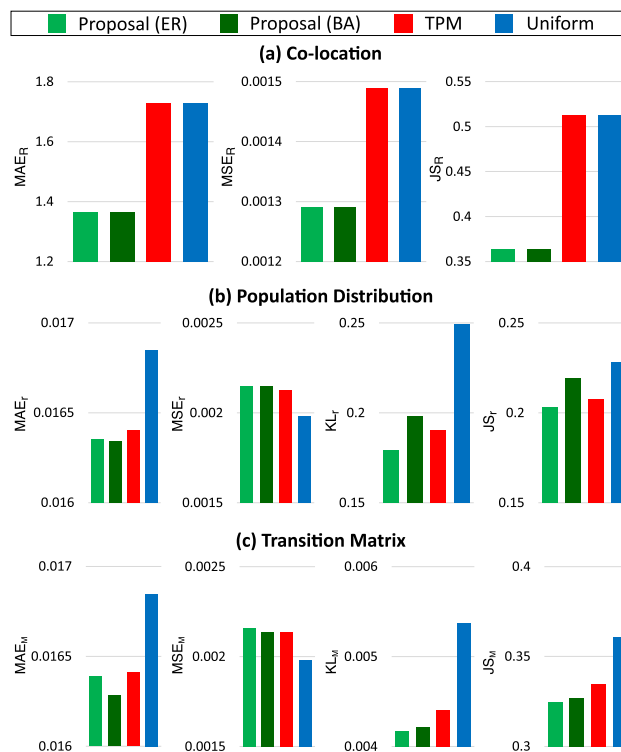


Fig. 21 Comparison results in Gowalla ($\epsilon_1 = \epsilon_2 = \epsilon_3 = 1$, $c = 5$, $\theta = 100$). **a** MAE/MSE/JS of $\bar{\mathbf{R}}'$, **b** MAE/MSE/KL/JS of \mathbf{r}' , **c** MAE/MSE/KL/JS of \mathbf{M}' . Smaller is better in all utility metrics. Proposal uses the Laplace mechanism

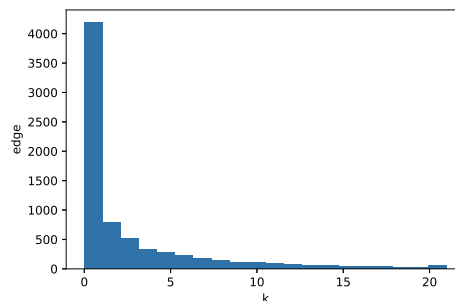


Fig. 22 Degree distribution of the training graph

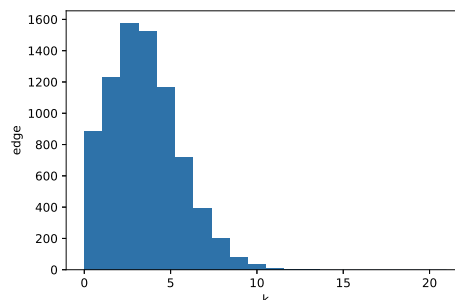


Fig. 23 Degree distribution of the ER graph ($\epsilon_1 = 1$)

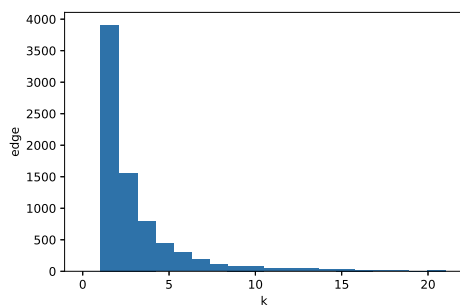


Fig. 24 Degree distribution of the BA graph ($\epsilon_1 = 1$)

as TPM do not. In particular, Proposal with the BA graph has a power-law degree distribution and preserves the friendship property of the training graph.

- Proposal also preserves other statistical features such as the population distribution and the transition matrix.
- Proposal synthesizes realistic traces while providing strong privacy, e.g., 0.2-node DP ($\epsilon_1 = 0.2$) for the training graph and 2-user-level DP ($\epsilon_2 = \epsilon_3 = 1$) for the training traces.

We need additional privacy budgets ϵ_1 and ϵ_2 to preserve the information about co-locations, which has not been considered in the existing synthesizers (as shown in Table 1). The additional privacy budgets are reasonably small, e.g., $\epsilon_1 = 0.2$ and $\epsilon_2 = 1$. Thus, we conclude that a synthetic trace generation that preserves co-locations of friends is now possible under strong privacy notations such as node DP and user-level DP.

6 Conclusion

We proposed a location synthesizer for generating synthetic traces that include co-locations of friends. Our location synthesizer generates such traces, while providing node DP for a training graph and user-level DP for training traces. Through comprehensive experiments using two real datasets, we showed that our synthesizer generates synthetic traces that preserve information about co-locations, such as the friendship probability, the co-location count matrix, and the degree distribution. Our synthetic traces also preserve other statistical features, such as the population distribution and transition matrix. The proposed synthesizer generates such traces while providing node DP and user-level DP with reasonable privacy budgets, e.g., 0.2-node DP ($\epsilon_1 = 0.2$) for the training graph and 2-user-level DP ($\epsilon_2 = \epsilon_3 = 1$) for the training traces. For example, our synthetic traces are useful for studying the effectiveness of friend suggestion on SNS based on co-locations [50].

In this work, we regarded the number θ of generated co-locations as a tuning parameter. For future work, we would

like to automatically determine an appropriate value of θ while providing DP for the training traces. Another interesting future work would be to incorporate the real-valued friendship level (rather than 0/1 considered in this work) between users into our algorithm. We would also like to use graph generation models with parameters other than a single friendship probability (e.g., exponential random graph model [43], stochastic block model [17]) under DP.

Acknowledgements This study was supported in part by JSPS KAKENHI JP22H00521, JP21H03442, and JST Moonshot R&D Grant Number JPMJMS2215.

Availability of data and material We can provide all data, figures, and Python scripts to reproduce the graphs and tables in this paper upon reasonable request.

Declarations

Conflict of interest Masakatsu Nishigaki is a member of the journal EB. Apart from this, we are not aware of any conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

A Details of Privelet

Applying Privelet to each row of \mathbf{Q} . Below, we explain how we apply Privelet (for one-dimensional nominal data) [49] to each row of the co-location count matrix \mathbf{Q} in detail. Let $\mathbf{Q}_i \in \mathbb{Z}_{\geq 0}^{|\mathcal{X}|}$ be the i -th row of \mathbf{Q} ($1 \leq i \leq |T|$). We call \mathbf{Q}_i the i -th co-location count vector.

First, we construct a tree structure of locations, e.g., a tree composed of categories/subcategories of POIs [7, 25] (see the left-hand side of Fig. 3 in [49] for an example of the tree). Each leaf node has a value in \mathbf{Q}_i . Then, we perform a nominal wavelet transform to the tree structure (see Section 5.1 in [49] for the details of the nominal wavelet transform and the right-hand side of Fig. 3 in [49] for an example of the wavelet coefficient). We add $\text{Lap}(\frac{c}{\mathcal{W}_\gamma \epsilon_2})$ to each wavelet coefficient γ , where \mathcal{W}_γ is a weight value assigned to the

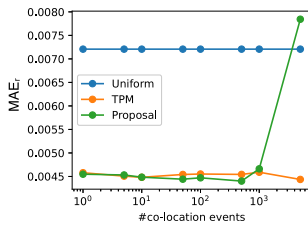


Fig. 25 MAE of the population distribution \mathbf{r}' (Proposal: BA graph, $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1, c = 5$) in Foursquare

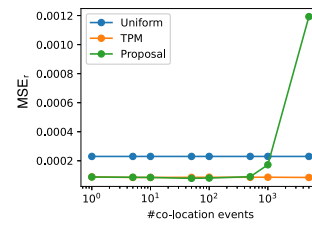


Fig. 26 MSE of the population distribution \mathbf{r}' (Proposal: BA graph, $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1, c = 5$) in Foursquare

coefficient γ . Specifically, if the coefficient γ is for the root node, then $\mathcal{W}_\gamma = 1$; otherwise, $\mathcal{W}_\gamma = \frac{f}{2f-2}$, where f is the fan-out (number of child nodes) of γ 's parent. We calculate values in leaf nodes from the noisy tree to obtain a noisy co-location count vector $\mathbf{Q}'_i \in \mathbb{R}^{|\mathcal{X}|}$. Finally, we concatenate $\mathbf{Q}'_1, \dots, \mathbf{Q}'_{|\mathcal{T}|}$ to obtain a noisy co-location count matrix $\mathbf{Q}' \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{X}|}$.

Proof of Theorem 3. Below, we provide a proof of Theorem 3.

Proof Let $\mathcal{S}^{(1)}, \mathcal{S}^{(2)} \subseteq \mathcal{R}$ be traces that differ in the entire trace of one user. Let $\mathbf{Q}^{(1)}, \mathbf{Q}^{(2)} \in \mathbb{Z}_{\geq 0}^{|\mathcal{T}| \times |\mathcal{X}|}$ be the co-location count matrices calculated from $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$, respectively. Let $d_i = \|\mathbf{Q}_i^{(1)} - \mathbf{Q}_i^{(2)}\|_1$, i.e., the l_1 distance between the i -th rows of $\mathbf{Q}^{(1)}$ and $\mathbf{Q}^{(2)}$.

Let $\mathcal{M}_i^{\text{Privelet}}$ be Privelet applied to the i -th row of the co-location count matrix. $\mathcal{M}_i^{\text{Privelet}}$ adds $\text{Lap}(\frac{c}{\mathcal{W}_\gamma \epsilon_2})$ to each wavelet coefficient γ . If $d_i = 1$, then $\mathcal{M}_i^{\text{Privelet}}$ provides the following inequality for any $z \in \text{Range}(\mathcal{M}_i^{\text{Privelet}})$:

$$\Pr[\mathcal{M}_i^{\text{Privelet}}(\mathbf{Q}_i^{(1)}) = z] \leq e^{\epsilon_2/c} \Pr[\mathcal{M}_i^{\text{Privelet}}(\mathbf{Q}_i^{(2)}) = z]$$

(see [49] for the proof). By group privacy [10], for a general case where $d_i \geq 1$, $\mathcal{M}_i^{\text{Privelet}}$ provides the following inequality for any $z \in \text{Range}(\mathcal{M}_i^{\text{Privelet}})$:

$$\Pr[\mathcal{M}_i^{\text{Privelet}}(\mathbf{Q}_i^{(1)}) = z] \leq e^{d_i \epsilon_2/c} \Pr[\mathcal{M}_i^{\text{Privelet}}(\mathbf{Q}_i^{(2)}) = z].$$

Concatenating $\mathbf{Q}_1, \dots, \mathbf{Q}_{|\mathcal{T}|}$, we have

$$\Pr[\mathcal{M}^{\text{Privelet}}(\mathbf{S}^{(1)}) = z] \leq e^{(\sum_{i=1}^{|\mathcal{T}|} d_i) \epsilon_2/c} \Pr[\mathcal{M}^{\text{Privelet}}(\mathbf{S}^{(2)}) = z] \leq e^{\epsilon_2} \Pr[\mathcal{M}^{\text{Privelet}}(\mathbf{S}^{(2)}) = z]$$

The last inequality holds because we have $\sum_{i=1}^{|\mathcal{T}|} d_i \leq c$ by trimming. Since the last inequality holds for any neighboring sets $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$, $\mathcal{M}^{\text{Privelet}}$ provides ϵ_2 -user-level DP. \square

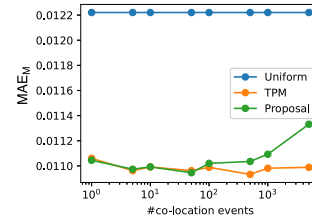


Fig. 27 MAE of the transition matrix \mathbf{M}' (Proposal: BA graph, $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1, c = 5$) in Foursquare

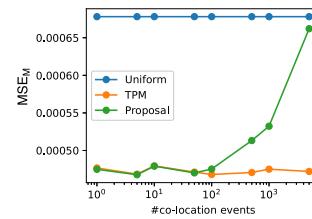


Fig. 28 MSE of the transition matrix \mathbf{M}' (Proposal: BA graph, $\epsilon_1 = \epsilon_2 = \epsilon_3 = 1, c = 5$) in Foursquare

B Effect of the number θ of generated co-location events

We evaluated the relationship between the number θ of generated co-location events and the utility of the population distribution and the transition matrix. Here, we used the MAE and MSE as utility metrics, used the BA model in Proposal, and used Foursquare as a dataset. We also confirmed that similar results were obtained when the KL divergence, JS divergence, the ER model, or Gowalla was used.

Figs. 25, 26, 27, and 28 show the results. These figures show that when the number θ of co-location events increases from 1000, the MAE and the MSE in Proposal increase. This comes from the fact that there are too many co-locations in the synthetic traces. In this case, the population distribution and the transition matrix are not preserved well even when we complement the remaining locations using the Viterbi algorithm. Thus, we should determine an appropriate value of θ in advance, either manually or automatically. One way to automatically set θ is to calculate the frequency of co-locations from training traces with DP and then set θ based on the private co-location frequency. We leave exploring such automatic setting of θ as future work.

```

Input: Synthetic traces  $\mathcal{S}_{\text{syn}} \subseteq \mathcal{U} \times \mathcal{X} \times \mathcal{T}$ , transition probability
matrix  $\mathbf{Z} \in [0, 1]^{|\mathcal{X}| \times |\mathcal{X}|}$  providing  $\epsilon_3$ -user-level DP.
Output: Synthetic traces  $\mathcal{S}_{\text{syn}} \subseteq \mathcal{U} \times \mathcal{X} \times \mathcal{T}$ .
/* Stationary distribution  $\mathbf{v} \in [0, 1]^{|\mathcal{X}|}$  */
1  $\mathbf{v} \leftarrow \text{ComputeEigenvector}(\mathbf{I} - \mathbf{Z}^\top)$ ;
/* Output probabilities  $\omega \in [0, 1]^{|\mathcal{U}| \times |\mathcal{T}| \times |\mathcal{X}|}$  */
2  $\omega = \mathbf{1}$ ;
3 foreach  $(u_i, x_k, t_l) \in \mathcal{S}_{\text{syn}}$  do
4   foreach  $k' \in [|\mathcal{X}|] \setminus \{k\}$  do
5      $\omega_{ilk'} = 0$ ;
6   end
7 end
/* Generate locations other co-locations */
8 foreach  $i \in [|\mathcal{U}|]$  do
9   /* Time  $t_1$  */
10  foreach  $k \in [|\mathcal{X}|]$  do
11     $\delta_{ilk} \leftarrow \mathbf{v}_k \omega_{ilk}$ ;
12  end
13  /* Time  $t_2, \dots, t_{|\mathcal{T}|}$  */
14  foreach  $l \in \{2, 3, \dots, |\mathcal{T}|\}$  do
15    foreach  $k \in [|\mathcal{X}|]$  do
16       $\delta_{ilk} \leftarrow \max_{k' \in [|\mathcal{X}|]} [\delta_{i(l-1)k'} \mathbf{Z}_{k'k}] \omega_{ilk}$ ;
17       $\psi_{ilk} \leftarrow \arg \max_{k' \in [|\mathcal{X}|]} [\delta_{i(l-1)k'} \mathbf{Z}_{k'k}]$ ;
18    end
19  end
20  /* Backtrack from time  $t_{|\mathcal{T}|}$  */
21   $q_{|\mathcal{T}|} \leftarrow \arg \max_{k \in [|\mathcal{X}|]} \delta_{i|\mathcal{T}|k}$ ;
22  foreach  $l \in \{|\mathcal{T}| - 1, |\mathcal{T}| - 2, \dots, 1\}$  do
23     $q_l \leftarrow \psi_{i(l+1)q_{l+1}}$ ;
24  end
25  /* Add the most likely sequence of
locations  $x_{q_1}, \dots, x_{q_{|\mathcal{T}|}}$  to  $\mathcal{S}_{\text{syn}}$  */
26  foreach  $l \in [|\mathcal{T}|]$  do
27     $\mathcal{S}_{\text{syn}} \leftarrow \mathcal{S}_{\text{syn}} \cup \{(u_i, x_{q_l}, t_l)\}$ ;
28  end
29 return  $\mathcal{S}_{\text{syn}}$ 

```

Algorithm 2: Generating locations other than co-locations using the Viterbi algorithm.

C Details of the Viterbi algorithm

Algorithm 2 shows the `ViterbiAlgorithm` function in Algorithm 1. Note that in both Algorithms 1 and 2, we represent synthetic traces \mathcal{S}_{syn} as a set of triplets (u_i, x_k, t_l) of user u_i , location x_k , and time instant t_l for ease of presentation. Algorithm 2 takes \mathcal{S}_{syn} (which includes only co-locations) and a transition probability matrix $\mathbf{Z} \in [0, 1]^{|\mathcal{X}| \times |\mathcal{X}|}$ as input and outputs \mathcal{S}_{syn} (which also includes locations other than co-locations). It complements locations other than co-locations by using the Viterbi algorithm [42], which finds the most likely sequence of locations (i.e., Viterbi path), as shown in Fig. 7.

Specifically, we first calculate a stationary distribution $\mathbf{v} \in [0, 1]^{|\mathcal{X}|}$ from the transition matrix \mathbf{Z} by calculating the eigenvector of $\mathbf{I} - \mathbf{Z}^\top$, where \mathbf{I} is the identity matrix (line 1). Then, we use the co-locations as observations and calculate traces that maximize the likelihood. Let

$\omega \in [0, 1]^{|\mathcal{U}| \times |\mathcal{T}| \times |\mathcal{X}|}$ be a tensor whose (i, l, k) -th element ω_{ilk} represents the probability of outputting the observation (u_i, x_k, t_l) ; i.e., if \mathcal{S}_{syn} include (u_i, x_k, t_l) , then $\omega_{ilk} = 1$ and $\omega_{ilk'} = 0$ for $k' \neq k$ (lines 2–7). For each user $u_i \in \mathcal{U}$, let $\delta_{ilk} \in [0, 1]$ be the joint probability of the observations until time t_l and the most likely trace until time t_l that includes location x_k at time t_l . We calculate δ_{i1k} at time t_1 as $\delta_{i1k} = \mathbf{v}_k \omega_{i1k}$, where \mathbf{v}_k is the k -th element of \mathbf{v} , and recursively calculate δ_{ilk} from time t_2 to $t_{|\mathcal{T}|}$ as follows:

$$\delta_{ilk} = \max_{k' \in [|\mathcal{X}|]} [\delta_{i(l-1)k'} \mathbf{Z}_{k'k}] \omega_{ilk}. \quad (7)$$

Let $\psi_{ilk} \in [|\mathcal{X}|]$ be the argument that maximizes (7). We calculate δ_{ilk} and ψ_{ilk} for each time t_l and location x_k (lines 9–17). Then, the joint probability of the observations and the most likely trace is

$$\max_{k \in [|\mathcal{X}|]} \delta_{i|\mathcal{T}|k}$$

and the most likely trace is obtained by backtracking from time $t_{|\mathcal{T}|}$ to 1 (lines 18–21). Finally, we add the most likely sequence of locations to \mathcal{S}_{syn} and output \mathcal{S}_{syn} (lines 22–25).

References

- Barabási, A.: Network Science. Cambridge University Press, Cambridge (2016)
- Bettini, C., Jajodia, S., Samarati, P., Wang, S.X.: Privacy in Location-Based Applications: Research Issues and Emerging Trends. Springer, Berlin (2009)
- Bindschaedler, V., Shokri, R.: Synthesizing plausible privacy-preserving location traces. In: IEEE S&P'16, pp. 546–563. IEEE (2016)
- Bindschaedler, V., Shokri, R., Gunter, C.: Plausible deniability for privacy-preserving data synthesis. VLDB Endow. **10**(5) (2017)
- Bollobás, B.: Random Graphs, 2nd edn. Cambridge University Press, Cambridge (2001)
- Chow, C.Y., Mokbel, M.F.: Trajectory privacy in location-based services and data publication. ACM SIGKDD Explor. Newsl. **13**(1), 19–29 (2011)
- DEVELOPERS, F.: Venue categories | build with foursquare. <https://developer.foursquare.com/docs/build-with-foursquare/categories/> (2020). Accessed 25 Oct 2020
- Dwork, C., McSherry, F., Nissim, K., et al.: Calibrating noise to sensitivity in private data analysis. In: TCC'06, pp. 265–284. Springer (2006)
- Dwork, C., Naor, M., Pitassi, T., et al.: Differential privacy under continual observation. In: STOC'10, pp. 715–724 (2010)
- Dwork, C., Roth, A.: The Algorithmic Foundations of Differential Privacy. Now Publishers (2014)
- Fettweis, G., Meyr, H.: Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck. IEEE Trans. Commun. **37**(8), 785–790 (1989)
- Gams, S., Killijian, M.O., Núñez del Prado Cortez, M.: De-anonymization attack on geolocated data. J. Comput. Syst. Sci. **80**(8), 1597–1614 (2014)
- Gursoy, M.E., Liu, L., Truex, S., Yu, L., Wei, W.: Utility-aware synthesis of differentially private and attack-resilient location traces.

- In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18), pp. 196–211 (2018)
14. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Proceedings of the 7th Python in Science Conference (SciPy'08), pp. 11–15 (2008)
 15. Hay, M., Li, C., Miklau, G., Jensen, D.: Accurate estimation of the degree distribution of private networks. In: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (ICDM'09), pp. 169–178 (2009)
 16. He, X., Cormode, G., Machanavajjhala, A., et al.: DPT: differentially private trajectory synthesis using hierarchical reference systems. *PVLDB* **8**(11), 1154–1165 (2015)
 17. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983)
 18. Hoshino, N.: A firm foundation for statistical disclosure control. *Jpn. J. Stat. Data Sci.* **3**, 721–746 (2020)
 19. Imola, J., Murakami, T., Chaudhuri, K.: Locally differentially private analysis of graph statistics. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security'21), pp. 983–1000 (2021)
 20. Iwata, T., Shimizu, H.: Neural collective graphical models for estimating spatio-temporal population flow from aggregated data. In: AAAI'19, vol. 33, pp. 3935–3942 (2019)
 21. Kearns, M., Roth, A., Wu, Z.S., Yaroslavtsev, G.: Private algorithms for the protected in social network search. *Proc. Natl. Acad. Sci.* **113**(4), 913–918 (2016)
 22. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD'11), pp. 193–204 (2011)
 23. Li, C., Miklau, G.: An adaptive mechanism for accurate query answering under differential privacy. *PVLDB* **5**(6), 514–525 (2012)
 24. Lichman, M., Smyth, P.: Modeling human location data with mixtures of kernel densities. In: KDD'14, pp. 35–44 (2014)
 25. Liu, Y., Wei, W., Sun, A., Miao, C.: Exploiting geographical neighborhood characteristics for location recommendation. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM'14), pp. 739–748 (2014)
 26. Liu, Z., Wang, Y., Smola, A.: Fast differentially private matrix factorization. In: RecSys'15, pp. 171–178 (2015)
 27. Matsuo, Y., Okazaki, N., Izumi, K., Nakamura, Y., Nishimura, T., Hasida, K., Nakashima, H.: Inferring long-term user properties based on users' location history. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007, pp. 2159–2165 (2007). <http://ijcai.org/Proceedings/07/Papers/348.pdf>
 28. Murakami, T., Arai, H., Hamada, K., Hatano, T., Iguchi, M., Kikuchi, H., Kuromasa, A., Nakagawa, H., Nakamura, Y., Nishiyama, K., Nojima, R., Oguri, H., Watanabe, C., Yamada, A., Yamaguchi, T., Yamaoka, Y.: Designing a location trace anonymization contest. In: Proceedings of Privacy Enhancing Technologies, pp. 225–243 (2023)
 29. Murakami, T., Hamada, K., Kawamoto, Y., et al.: Privacy-preserving multiple tensor factorization for synthesizing large-scale location traces. *PoPETs* **2021**(2), 5–26 (2021)
 30. Murakami, T., Kanemura, A., Hino, H.: Group sparsity tensor factorization for de-anonymization of mobility traces. In: 2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20–22, 2015, vol. 1, pp. 621–629 (2015). <https://doi.org/10.1109/Trustcom.2015.427>
 31. Murakami, T., Kanemura, A., Hino, H.: Group sparsity tensor factorization for re-identification of open mobility traces. *IEEE Trans. Inf. Forensics Secur.* **12**(3), 689–704 (2017). <https://doi.org/10.1109/TIFS.2016.2631952>
 32. Narita, J., Suganuma, Y., Nishigaki, M., Murakami, T., Ohki, T.: Synthesizing privacy-preserving location traces including co-locations. In: Proceedings of the 16th DPM International Workshop on Data Privacy Management (DPM'21), pp. 20–36 (2021)
 33. Nightley, for Spatial Information Science at the University of Tokyo (CSIS), C.: SNS-based people flow data. <http://nightley.jp/archives/1954> (2014). Accessed 25 Feb 2021
 34. Ninghui, L., Min, L., Dong, S.: *Differential Privacy: From Theory to Practice*. Morgan & Claypool Publishers, San Rafael (2016)
 35. Olteanu, A., Huguenin, K., Shokri, R., et al.: Quantifying the effect of co-location information on location privacy. In: PETS'14, pp. 184–203. Springer (2014)
 36. Olteanu, A., Huguenin, K., Shokri, R., et al.: Quantifying interdependent privacy risks with location data. *IEEE Trans. Mob. Comput.* **16**(3), 829–842 (2016)
 37. Olteanu, A., Humbert, M., Huguenin, K., et al.: The (co-)location sharing game. *PoPETs* **2019**(2), 5–25 (2019)
 38. PWS Cup 2019. https://www.iwsec.org/pws/2019/cup19_e.html (2019)
 39. Pyrgelis, A., Troncoso, C., Cristofaro, E.D.: Knock knock, who's there? Membership inference on aggregate location data. In: NDSS (2018)
 40. Qardaji, W., Yang, W., Li, N.: Understanding hierarchical methods for differentially private histograms. *PVLDB* **6**(14), 1954–1965 (2013)
 41. Qin, Z., Yu, T., Yang, Y., Khalil, I., Xiao, X., Ren, K.: Generating synthetic decentralized social graphs with local differential privacy. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17), pp. 425–438 (2017)
 42. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **77**(2), 257–286 (1989)
 43. Robins, G., Pattison, P., Kalish, Y., Lusher, D.: An introduction to exponential random graph (p^*) models for social networks. *Soc. Netw.* **29**(2), 173–191 (2007)
 44. Shokri, R., Stronati, M., Song, C., et al.: Membership inference attacks against machine learning models. In: S&P'17, pp. 3–18 (2017)
 45. Shokri, R., Theodorakopoulos, G., Le, B., et al.: Quantifying location privacy. In: IEEE S&P'11, pp. 247–262. IEEE (2011)
 46. Sofya, R., Adam, S.: *Differentially Private Analysis of Graphs*, pp. 543–547. Springer, Berlin (2016)
 47. Song, L., Kotz, D., Jain, R., et al.: Evaluating next-cell predictors with extensive Wi-Fi mobility data. *IEEE Trans. Mobile Comput.* **5**(12), 1633–1649 (2006)
 48. Stadler, T., Oprisanu, B., Troncoso, C.: Synthetic data—anonymisation groundhog day. *CoRR* 2011.07018 (2022). <https://arxiv.org/abs/2011.07018>
 49. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.* **23**(8), 1200–1214 (2010)
 50. Yang, D., Qu, B., Yang, J., et al.: Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach. In: WWW'19, pp. 2147–2157 (2019)
 51. Ye, M., Shou, D., Lee, W.C., Yin, P., Janowicz, K.: On the semantic annotation of places in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11), pp. 520–528 (2011)
 52. Yuan, G., Zhang, Z., Winslett, M., Xiao, X., Yang, Y., Hao, Z.: Low-rank mechanism: optimizing batch queries under differential privacy. *PVLDB* **5**(11), 1352–1363 (2012)
 53. Zheng, F.W., Zheng, Y., Yang, Q.: Joint learning user's activities and profiles from GPS data. In: Zhou, X., Xie, X. (eds.) Proceedings of the 2009 International Workshop on Location Based Social Networks, LBSN 2009, November 3, 2009, Seattle, Washington,

- USA, Proceedings, pp. 17–20. ACM (2009). <https://doi.org/10.1145/1629890.1629894>
54. Zheng, Y., Zhang, L., Xie, X., et al.: Mining interesting locations and travel sequences from GPS trajectories. In: WWW'09, pp. 791–800 (2009)

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