Research Article

Releasing Differential Private Trajectory Datasets Without Revealing Trajectory Correlations

Hong Zhu (1), Yunkai Yu (2), and Meiyi Xie (1)

1 School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
2 School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, China

Correspondence should be addressed to Meiyi Xie; xiemeiyi@hust.edu.cn

Received 25 August 2021; Revised 30 January 2022; Accepted 12 May 2022; Published 29 June 2022

Copyright © 2022 Hong Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the prevailing use of smartphones and location-based services, vast amounts of trajectory data are collected and used for many applications. When trajectory data are sent to a third-party research institute for analytical applications, the privacy of users would be severely disclosed. For example, the relationship between users will be revealed from the correlation between trajectories. In this paper, we propose a method for releasing trajectory datasets without revealing correlation between trajectories, called RDPT. In RDPT, we first quantify the trajectory correlation and convert the problem of protecting trajectory correlations into reducing the trajectory similarities of users and preserving the utility of the perturbed trajectories. Based on the insight, we model a multi-objective optimization problem and solve the problem with the particle swarm optimization algorithm modified to satisfy differential privacy. Then we generate synthetic trajectories and the correlations between trajectories are reduced. We conduct extensive experiments on three real trajectory datasets. The experimental results show that RDPT achieves almost equivalent data utility to and better privacy than the existing methods.

1. Introduction

With the extensive use of smart mobile devices furnished with location-based applications, such as WeChat and Twitter, vast amounts of trajectory data are being collected by location-based service (LBS) providers. Trajectory data can be widely used in many analytical applications, such as intelligent transportation, smart cities, infectious disease surveillance and epidemiological investigations. When LBS providers want to know whether a traffic congestion or certain events will occur in a certain area in the future, they often send the trajectories to a third-party research institute for an offline analysis. However, the third-party research institute is not always trusted and is likely a potential adversary. Moreover, since trajectories are formed by users’ temporal and spatial behavior as well as their social relationships [1], the trajectories can reflect the features of users’ behaviors. For example, an adversary could infer the social relationships of a user by analyzing correlation between trajectories of users. If two trajectories are correlated and have many check-in points close or even coincident that indicates they often visit close or the same locations, the attacker could infer the users of the two trajectories are friends with high probability because the two users often have activities together. From the social relationships, the attacker can even infer the health status, habits of daily life and occupations of the users with a high probability [2]. Therefore, the privacy of users is seriously revealed.

There are many privacy-preserving methods for trajectories, and these approaches can be summarized into three categories: methods without considering protection correlation [3–5], mechanisms considering correlations within a single trajectory [6–8] and approaches considering the trajectory correlations [9–11]. For the previous two categories, correlation between trajectories is not taken into consideration in these methods, causing the
There are also several researchers [9–11] who have proposed privacy-preserving mechanisms considering trajectory correlation. However, approaches in the literature [9, 10] are restricted to scenarios of publishing two trajectories. For example, these methods work well when two passengers in DiDi want to hide their trajectory correlation. However, there are more than two users in real social applications, so the number of trajectories published offline is usually greater than two. Moreover, even if we can reduce the correlation between two trajectories in a dataset, there is still a risk that the correlations amongst three or over three real trajectories are not reduced. This would still reveal the privacy of those users. Under scenarios in which a large number of trajectories need to be released, the research in the literature [9, 10] is no longer effective. In addition, although Zhao et al. [11] proposed a location privacy-preserving mechanism considering trajectory correlation, their approach was derived from k-anonymity and cannot resist location inference attacks [12–14]. Therefore, even if the trajectories are perturbed by existing methods, the problem of privacy leakage caused by trajectory correlation still persists.

To solve the problem of privacy leakage caused by trajectory correlation when LBS providers release a large number of trajectories offline, we propose a method for releasing differential private trajectories, called RDPT. First, the continuous geographical space of real trajectories is discretized via an adaptive grid partition method. The continuous geographical space is divided into $N \times N$ top-level cells, and the original location trajectories are converted into cell trajectories. For each top-level cell, to preserve the privacy when we divide each top-level cell into bottom-level cells, we add Laplacian noise [15] to the frequencies of locations in the top-level cell. Second, as the spatial distribution and frequencies of visiting to the top-level cells are important features for calculating correlation of trajectories, we extract the cell visit probability vector from each cell trajectory of a user. Then we quantify the trajectory correlation and convert the problem of protecting the trajectory correlation into reducing the similarity of each pair of obfuscated cell visit probability vectors and retaining the similarity between the real cell visit probability vector and the corresponding perturbed cell visit probability vector to preserve the utility of the perturbed trajectories. Based on this insight, we model a multi-objective optimization problem to find the balance between the security and data utility. By solving the problem with the particle swarm optimization algorithm [16], which is modified to satisfy differential privacy (PSO-DP), we obtain a perturbed cell visit probability vector for a cell trajectory of a user and reduce the correlations between the obfuscating trajectory of a user and the perturbed trajectories of other users. Finally, based on the perturbed cell visit probability vector, we generate a synthetic trajectory. Thus, the correlation between trajectories of users is reduced and the social relationship between users is protected.

To solve multi-objective optimization problem, there are generally several methods, such as derivative-based methods and evolutionary algorithms (EAs). Derivative-based methods are useful for solving problems with low dimensions, which are not suitable for our multi-objective optimization problem because our problem is complexity and high dimensions $N^2$. Conversely, EAs are widely used optimization methods that can effectively deal with complex problems. The genetic algorithm (GA) [17], particle swarm optimization algorithms (PSO) [16, 18, 19] and differential evolution (DE) algorithms [20, 21] are widely used in the proposed EAs. However, these algorithms cannot be used directly to solve our problem. No matter what algorithms in literature [16, 18–21] are selected, we need to add noise into these algorithms to satisfy differential privacy because reading an original trajectory would lead to a potential privacy leakage. Furthermore, compared with GA and DE, PSO has a simpler principle and fewer parameters, and we need not process the crossover and mutation for the trajectories such as genetic algorithm with differential privacy [17], thus ensuring its efficiency in different situations. Moreover, since users usually interact with other users in social networking, the trajectory of a user is formed by the user’s social ties in LBS applications, which is similar to the idea of PSO. Therefore, we choose a classical PSO to solve our problem.

The difference between our PSO-DP and other multi-objective optimization algorithms is that we modify a classical PSO algorithm with exponential mechanism of differential privacy in selection procedure. When we select a local optimal particle, since we read a real trajectory we need add noise to preserve privacy. So, we select a particle as a local optimal particle according to the probabilities of local particles that are calculated by utility scores. Usually, the local optimal particle would be selected with higher probability, but we may or may not select the local optimal particle because of randomness. Thus, the privacy is preserved. To the best of our knowledge, this is the first work about PSO with differential privacy in trajectory privacy-preserving.

The main contributions of this paper are summarized as follows:

1. We propose a method of releasing differential private trajectory datasets i.e., RDPT. We firstly model the problem of reducing trajectory correlations between trajectories of users and retaining data utility as a multi-objective optimization problem. By solving the problem, we can synthesize trajectories for users.

2. We adapted a classical PSO algorithm to solve the modelled multi-objective optimization problem. By modifying the selection procedure of local optimal particles in each iteration to satisfy differential privacy, we obtain a PSO-DP algorithm and prove the security of PSO-DP.

3. We evaluate our method on three real datasets, five different metrics and two specific implementations of quantifying the trajectory correlation. The
experimental results demonstrate that our method achieves almost equivalent data utility to and better privacy than the existing methods.

The remainder of this paper is organized as follows. In section 2 we review the related work in the literature. We present preliminaries in section 3 and formally define the problem in section 4. In section 5 we give a detailed description of RDPT. We show the experimental results and detailed analysis in section 6. Finally, we conclude this paper in section 7.

2. Related Work

In this section we briefly review three existing categories of privacy protection methods for the trajectories. Then we analyze the evolutionary algorithm satisfying differential privacy.

Trajectory privacy protection methods without considering the correlation. We can divide the approaches into: suppression methods [4], bounded perturbation [22], K-anonymity and its derivation approaches [3, 23, 24], and differential private methods [25–30]. Hasan et al proposed a privacy architecture with a bounded perturbation technique to protect user’s trajectory from the privacy breaches [22]. Huo et al proposed k-anonymity called “YCWA” [23]. They first extracted the sensitive locations in the trajectory according to the time interval and location density. Then they partitioned the geographic space of trajectories into discrete k-anonymity regions, and the sensitive locations in the trajectories were replaced with the corresponding k-anonymity regions to protect the sensitive locations. The YCWA method reduces the complexity and information loss of k-anonymity over complete trajectories and preserves the privacy of the individuals. Zhang et al. proposed a dual-K mechanism (DKM) to protect users’ trajectory privacy [24]. DKM first inserts multiple anonymizers between the user and the location service provider (LSP), and K query locations are sent to different anonymizers to achieve K-anonymity. Simultaneously, they combined a dynamic pseudonym and location selection mechanisms to improve user trajectory privacy. Although k-anonymity has been widely applied in many applications, the datasets published by these methods still face combination attacks and background knowledge attacks. Differential privacy has become a widely adopted privacy protection mechanism because of its strong guarantee of privacy. In literature [28], Ding et al. proposed a stream processing framework with differential privacy that contains two modules for trajectories. One module can concurrently receive real-time queries from individuals and release new sanitized trajectories, and another module comprises three algorithms based on differential privacy to facilitate publication of the distribution of location statistics. Wang et al. developed a privacy-preserving reference system that can extract privacy-demanding feature-based anchors that can subsequently be used to calibrate sequences from raw trajectories [29]. They provided a private trajectory data sensitization approach that scales to large spatial domains reflecting realistic trajectories.

In the literature [30], the authors presented an algorithm for protecting sensitive place visits in privacy-preserving trajectory publishing. By generalizing sensitive places using sensitive zones and distorting the sub-trajectories within the sensitive zones based on differential privacy, their method not only prevents leakages of sensitive place visits, but also preserves individual movement features. These privacy-preserving approaches for trajectories focus on sensitive information of locations and do not consider the privacy leakage caused by trajectory correlations contained in the trajectories.

Trajectory privacy protection methods considering correlations within a single trajectory. The spatiotemporal correlation contained in the trajectory data easily leads to the privacy leakage problem of the users [31]. Many researchers have proposed trajectory privacy-preserving methods considering the temporal and spatial correlation within a trajectory. He et al. proposed a differential private approach based on the spatiotemporal correlation in a trajectory, called DPT [6]. They first constructed a hierarchical index system to capture users’ mobility features. They then constructed a prefix tree to represent the spatial transfer features between adjacent locations of trajectories and added Laplacian noise to the visit frequencies of each node in the prefix tree. Finally, the perturbed trajectory is synthesized according to the obfuscated prefix tree to protect the spatial correlation contained within the trajectory. In the literature [7], the authors presented a differential private method called TGM for publishing trajectories. In TGM, they partitioned the geographical space and constructed a prefix sequence graph to model the spatial transfer features between grids in trajectories. Then the trajectories were iteratively synthesized using an exponential mechanism. Gursoy et al. [8] proposed a differentially private trajectory synthesis method. They extracted the Markov transition matrix, trajectory length probability distribution and journey probability distribution, and obfuscated the three features by the Laplacian mechanism. During the synthesis of trajectories, the trajectories were processed considering against Bayesian attacks, area (sub trajectory) sniffing attacks and abnormal location leakage attacks. These methods only consider spatiotemporal correlations within a single trajectory, and still do not focus on the trajectory correlation between different trajectories, which could still cause serious privacy leakage.

Trajectory privacy protection methods with correlation between different trajectories. Ou et al. [9] proposed a trajectory publication mechanism based on a hidden Markov model (HMM) to protect the correlation between trajectories. Similarly, in [10] the authors proposed an n-body Laplace framework. Then, under the framework, they presented two privacy protection methods for two types of data utilities. However, these methods are restricted to the scenario of releasing two trajectories and cannot provide an efficient privacy protection when publishing a large number of trajectories offline. To ease the social relationship attacks caused by trajectory correlation, Zhao et al. designed an effective model to simultaneously deal with social relationship attacks and reidentification attacks while
maintaining a high data utility [11]. In their model, they proposed a sliding-window algorithm that is a variant of K-anonymity, i.e., $K^m$-anonymity. The $K^m$-anonymity generates anonymized trajectories according to the social-aware distance, which concerns both the spatiotemporal distance and the social proximity. Moreover, the $K^m$-anonymity processes the anonymity with sub-trajectories in an $m$-length window instead of the entirety of the trajectories. However, the $K^m$-anonymity approach only satisfies k-anonymity and cannot resist homogeneous attacks or location inference attacks [12–14].

We summarize the three category works about privacy-preserving approaches in Table 1.

Evolutionary algorithm satisfying differential privacy. Evolutionary algorithms are widely used to solve multi-objective optimization problems. There are few works on the application of evolutionary algorithms in privacy protection. In [17], Zhang et al. proposed PrivGene, a differentially private model fitting task using genetic algorithms. They modified the genetic algorithm and proposed a differential private version of the genetic algorithm. In PrivGene, the authors use an exponential mechanism to select parent individuals for crossing and mutation, thus enhancing the security of the selection process.

3. Preliminaries

In this section, we introduce the basic concepts, including differential privacy, global sensitivity, and particle swarm optimization.

**Definition 1 (Trajectory).** A trajectory $T$ is a time-series sequence of tuples (location, time), i.e., $T = \{ (\ell_1, t_1), (\ell_2, t_2), \ldots, (\ell_L, t_L) \}$, where $\ell_i$ is a location consisting of latitude and longitude, $t_i$ is the moment when location $\ell_i$ is generated, and $L$ is the number of locations in trajectory $T$.

**Definition 2 (Neighbor datasets).** Suppose $D_1$ and $D_2$ are two datasets. $D_1$ and $D_2$ are neighbor datasets if and only if $D_1 = D_2 \cup \{ R \}$ or $D_2 = D_1 \cup \{ R \}$, where $R$ is a record in a dataset.

**Definition 3 (Differential Privacy).** Let $A$ be a privacy protection mechanism, $O$ be any output of $A$, and $D_1$ and $D_2$ be neighbor datasets. $A$ will satisfy $\varepsilon$-differential privacy if we have:

$$\Pr(A(D_1) = O) \leq e^\varepsilon \ast \Pr(A(D_2) = O).$$

**Definition 4 (Global sensitivity).** Global sensitivity indicates the maximum difference of two query results over neighboring datasets. Suppose $Q$ is a query function, the definition of global sensitivity is:

$$\Delta Q = \max_{D_1, D_2} \| Q(D_1) - Q(D_2) \|_1.$$

There are two widely used mechanisms for achieving differential privacy, i.e., the Laplace mechanism [15] and the exponential mechanism [32]. The Laplace mechanism is suitable for perturbing numerical query results, and the exponential mechanism is suitable for perturbing nonnumerical query results. We use the Laplace mechanism and exponential mechanism in RDPT.

There are two common properties in differential privacy. The first property is sequential composition, indicating that a sequence of computations where each provides differential privacy in isolation also provides differential privacy in sequence, but the privacy budget $\varepsilon$ is accumulated. The second property is parallel composition, meaning that if the sequence of computations is performed on disjoint databases, then the privacy budget is not accumulated additively, but rather determined by the worst privacy guarantees of all computations. The following definitions formally describe these two properties.

**Definition 5 (Sequential Composition).** Suppose an algorithm $A$ runs $n$ randomized algorithms: $A_1; A_2; \ldots; A_n$, and each $A_i (1 \leq i \leq n)$ satisfies $\varepsilon_i$-differential privacy. When publishing the output $A$ that runs over a dataset $D$ in sequence: $A(D) = (A_1(D), A_2(D), \ldots, A_n(D))$, $A$ satisfies $\sum_{i=1}^n \varepsilon_i$-differential privacy.

**Definition 6 (Parallel Composition).** Suppose an algorithm $A$ runs $n$ randomized algorithms: $A_1; A_2; \ldots; A_n$, and each $A_i (1 \leq i \leq n)$ satisfies $\varepsilon_i$-differential privacy. When publishing the output $A$ that runs over disjoint dataset $D_i$ for $A_i$: $A(D) = (A_1(D_1), A_2(D_2), \ldots, A_n(D_n))$, and $D = D_1 \cup D_2 \cup \ldots \cup D_n$, $A$ satisfies $\max_{1 \leq i \leq n} \varepsilon_i$-differential privacy.

Particle Swarm Optimization (PSO). PSO is an evolutionary algorithm [16]. In PSO, each particle searches for the optimal solution as an extremum relative to the objective. The optimal individual extremum in the swarm is considered to be the current global optimal solution. Then, each particle adjusts its speed and position based on its extremum and the global optimal solution. The above process is iterated many times until PSO is convergent. Then the current global optimal solution is the final solution of a given optimization problem.

4. The problem

Attack hypothesis: Suppose a third-party research institute obtains a trajectory dataset for analysis applications. Since the third-party institute is not always trusted and is likely a potential adversary, if the data owners had not perturbed the trajectories before they released the trajectory dataset offline, the adversary could have extracted the correlations from the trajectories, and the social ties among users or other privacy would be revealed. Therefore, in this paper, we will suppose the adversary can obtain the following information:

(1) A published trajectory dataset $D' = \{ T'_i | i \in [1, N'] \}$, where $T'_i$ is a perturbed trajectory and $N'$ is the number of trajectories.
from location mode into cell mode, i.e., a cell trajectory translation using the high data utility. Much privacy of individuals as possible while maintaining a relationship between users. In a word, we can protect as much as possible, we model a multi-objective optimization problem. Then we solve the problem via a particle swarm optimization algorithm modified by an exponential mechanism and obtain a perturbed cell visit probability vector for $T_u^C$. The privacy budget consumed in this step is $\epsilon_2$.

5. Releasing correlated trajectory datasets

5.1. The overview. Before describing RDPT in detail, we provide a high-level description of our approach. Suppose that the overall privacy budget consumed by RDPT is $\varepsilon = \varepsilon_1 + \varepsilon_2$. There are the following three steps in RDPT:

Step 1. We divide the geographical space of $D$ into $N^2$ identical cells and obtain a grid $G$ via an adaptive grid partition method. Then the trajectory $T_u$ in $D$ is converted from location mode into cell mode, i.e., a cell trajectory $T_u^C$, and the dataset $D^C$ is obtained for cell trajectories. Then the cell visit probability vectors are extracted from the cell trajectories $D^C$, and we can quantify the trajectory correlation using the $N^2$-dimensional cell visit probability vectors. For each cell in grid $G$, we add Laplacian noise into the density for locations in the top-level cell for all trajectories when we divide the cell (in the following, we call the cell a top-level cell) into bottom-level cells. The density for locations in a top-level cell for a trajectory is calculated by the density for locations in the top-level cell for all trajectories. When we divide the cell into identical cells and obtain a grid space $G = \{C_1, C_2, \ldots, C_N\}$. We call a cell in $G$ a top-level cell. Then, each trajectory $T_u^C$ for user $u$ in $D$ is converted into a cell trajectory $T_u^C = \{(C_1, t_1), (C_2, t_2), \ldots, (C_N, t_L)\}$ (suppose the number of locations in $T_u$ is $L$). Then, the geographic space of $D$ is discretized, and the original dataset $D$ is converted into a set of trajectories in cell mode $D^C$. We have following definitions.

Definition 7 (Cell trajectory). For $N \times N$ partition over the geographic space in $D$, we have a grid space $G = \{C_1, C_2, \ldots, C_N\}$, $G = \{C_1, C_2, \ldots, C_N\}$. For an original trajectory in location mode $T = \{(\ell_1, t_1), (\ell_2, t_2), \ldots, (\ell_L, t_L)\}$ (the number of locations in $T$ is $L$), if $\ell_1$ is in cell $C_1$ and $\ell_k$ is in cell $C_k$, ..., $\ell_L$ is in cell $C_k$, then we have a trajectory in cell mode $T^C = \{(C_1, t_1), (C_2, t_2), \ldots, (C_N, t_L)\}$. We call $T^C$ a cell trajectory.

<table>
<thead>
<tr>
<th>Category</th>
<th>Literature</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods considering correlations within a single trajectory</td>
<td>[6–8, 31]</td>
<td>Do not consider the privacy leakage caused by trajectory correlations</td>
</tr>
<tr>
<td>Methods considering correlations between trajectories</td>
<td>Only releasing two trajectories [9, 10]</td>
<td>Not suitable for publishing trajectories more than three cannot resist homogeneous attacks or location inference attacks [12–14]</td>
</tr>
</tbody>
</table>

(2) A user set $U = \{u\} \subseteq \{1, N'\}$ corresponding to $D'$, where for any user $u \in U$ there is only one trajectory $T^u \in D'$.

(3) Quantification of the trajectory correlation and the trajectory privacy-preserving method in this paper.

Our goal is as follows: given a real trajectory dataset, we perturb the dataset to reduce the trajectory correlation between a trajectory and other trajectories to the greatest extent possible, while ensuring a high data utility. Even if the adversary obtains the perturbed dataset, the quantification method of the trajectory correlation and the privacy-preserving approach, the adversary still cannot infer the social relationship between users. In a word, we can protect as much privacy of individuals as possible while maintaining a high data utility.

5.2. Adaptive Grid Partition and Quantification of Trajectory Correlation

5.2.1. Adaptive Grid Partition. It is difficult for us to generate the location visit probability vectors in the same dimension from the original location dataset $D$. Therefore, we divide the geographic space of $D$ into $N^2$ identical cells and obtain the grid space $G = \{C_1, C_2, \ldots, C_N\}$. We call a cell in $G$ a top-level cell. Then, each trajectory $T_u$ for user $u$ in $D$ is divided into a cell trajectory $T_u^C = \{(C_1, t_1), (C_2, t_2), \ldots, (C_N, t_L)\}$ (suppose the number of locations in $T_u$ is $L$). Then, the geographic space of $D$ is discretized, and the original dataset $D$ is converted into a set of trajectories in cell mode $D^C$. We have following definitions.

Definition 7 (Cell trajectory). For $N \times N$ partition over the geographic space in $D$, we have a grid space $G = \{C_1, C_2, \ldots, C_N\}$, $G = \{C_1, C_2, \ldots, C_N\}$. For an original trajectory in location mode $T = \{(\ell_1, t_1), (\ell_2, t_2), \ldots, (\ell_L, t_L)\}$ (the number of locations in $T$ is $L$), if $\ell_1$ is in cell $C_1$ and $\ell_k$ is in cell $C_k$, ..., $\ell_L$ is in cell $C_k$, then we have a trajectory in cell mode $T^C = \{(C_1, t_1), (C_2, t_2), \ldots, (C_N, t_L)\}$. We call $T^C$ a cell trajectory.
We improve the adaptive grid partition method in the literature [8]. Suppose that the total number of locations in a top-level cell for a dataset $D$ is $X$. Intuitively, a large $X$ would lead to many bottom-level cells. However, in extreme cases, if all of the $X$ locations occur in one place (or several near places), then all locations would be in exactly one bottom-level cell, which means that there is no location in other bottom-level cells, thus resulting in too many useless bottom-level cells and affecting the efficiency of trajectory synthesis in the last step. Therefore, both the diversity and the number of locations need to be considered when constructing the query function for bottom-level cell partitioning. Therefore, after we divide $D$ into $N^2$ top-level cells, for each top-level cell $C_i$, we count the sum of normalized number of locations in that cell for all cell trajectories as follows:

$$q(C_i) = \sum_{j \in D^C} \frac{f_{C_i,j}^C}{\left| T_j^C \right|},$$

(3)

where $f_{C_i,j}^C$ is the number of different locations inside the top-level cell $C_i$ for a cell trajectory $T_j^C$ corresponding to original trajectory $T_j$, $\left| T_j^C \right|$ is the number of locations in $T_j^C$, and $q(C_i)$ is the density of locations in the top-level cell $C_i$ for cell trajectories in $D^C$.

$q(C_i)$ is used for the adaptive partition of cell $C_i$. To enhance the security of adaptive grid partitioning, we add Laplacian noise to $q(C_i)$ to obtain a stochastic partition for the bottom-level cells. The important parameter for Laplacian noise is global sensitivity. Then we have the following theorem.

**Theorem 1.** The global sensitivity of the query $q(C_i)$ is 1.

**Proof.** Suppose that the neighboring datasets for the query are $D^C_1$ and $D^C_2 = D^C_1 \cup \{T^C\}$, where $T^C$ is a cell trajectory. Then we have the following deduction.

$$\Delta q = \max_{D_1,D_2} \left\| q(D_2) - q(D_1) \right\|_1,$n

$$= \max \left( \sum_{T \in D_1} f_{C_i,j}^C - \sum_{T \in D_2} f_{C_i,j}^C \right),
$$

$$= \max \left( \sum_{T \in D_1} f_{C_i,j}^C + f_{C_i,j}^C - \sum_{T \in D_2} f_{C_i,j}^C \right),
$$

$$= \max \left( \frac{f_{C_i,j}^C}{\left| T_j^C \right|} \right) \leq 1.$$

According to Definition 4, the global sensitivity of the query $q(C_i)$ is 1.

Therefore, it suffices to add $\text{Lap}(1/\epsilon_1)$ noise to each query answer to obtain the noisy answers $\bar{q}(C_i) = q(C_i) + \text{Lap}(1/\epsilon_1)$, where $\epsilon_1$ is the privacy budget. Then, each top-level cell $C_i$ is further divided into $B_i^2$ bottom-level cells according to $\bar{q}(C_i)$, where $B_i$ is proportional to $\bar{q}(C_i)$. $B_i$ is the same as that in the literature [8], i.e., $B_i = \lceil \frac{1}{\epsilon_1} \rceil$, where $N'$ is the number of trajectories in $D^C$.

5.2.2. Quantifying Trajectory Correlation. Quantifying trajectory correlation is a fundamental problem. There are three categories of methods: extraction of features from original check-in data [10, 33, 34], machine learning [35, 36], and statistical information of trajectories [31, 37]. The first category of methods has limitations: the lengths of different trajectories are different, and we need to preprocess the trajectories via interpolation to align the trajectories. Then, unnecessary errors are introduced in quantifying the trajectory correlation. In addition, these methods have strict requirements on length of trajectories, for instance, the length of trajectory cannot be too long. For the second category, the time cost is too massive to be suitable for applications in which the trajectory correlation needs to be computed many times. Conversely, the methods based on statistical information of trajectories consume less time and are more efficient, which is proportional to trajectory length or to the number of top-level cells. For our problem, the frequencies of visiting locations in the top-level cells are important feature for a trajectory, and the sequence of cells would describe the space distribution feature and transitions amongst check-ins within the trajectory. Then we use cell visit probability vector of a trajectory as the statistical information to describe the feature of a trajectory. In addition, we can align the trajectories in top-level cells by the adaptive grid partition. Therefore, we compute the trajectory correlation using the third category of method, and the statistical information we use in RDPT is the cell visit probability vector of a trajectory.

**Definition 8** (The cell visit probability vector). After we partitioned the geographic space of a trajectory dataset $D$ to $N \times N$ identical cells, for a cell trajectory $T^C$, we define a cell visit probability vector $P^C$, which is a $N \times N$-dimension vector. The k-th component of $P^C$ is then calculated as follows: for a cell $C_k$, if $T^C$ has $X$ locations within $C_k$ and the number of locations in $T^C$ (or the length of $T^C$) is $L$, then the k-th component of $P^C$ is $P^C[k] = X/L$.

**Definition 9** (Trajectory Correlation). The trajectory correlation $S$ is a measure of the correlation between two trajectories. Suppose $P^C_1$ and $P^C_2$ are two cell visit probability vectors of two trajectories $T^C_1$ and $T^C_2$, respectively, and a function $\text{sim}$ is a type of method calculating the vector similarity. We define $S$ as follows:

$$S(T^C_1, T^C_2) = \text{sim}(P^C_1, P^C_2).$$

(5)

There are many common implementations for calculating the similarity between vectors, such as cosine similarity and Pearson correlation coefficient. Therefore, our method can be applied to different specific implementations of trajectory correlation.

5.3. Modeling a multi-objective optimization problem. From this subsection, we will describe the second step in RDPT. We first model a multi-objective optimization
problem. Then, we perturb the real cell visit probability vector of $T_{u}^{C,act}$ for user $u$ to $T_{u}^{C,syn}$ and reduce trajectory correlations between cell visit probability vector $T_{u}^{C,syn}$ and the perturbed cell visit probability vectors of other users.

When we perturb the trajectories in the real dataset $D$ and obtain a perturbed dataset $D'$, we need to ensure a high data utility and security for $D'$. Therefore, for the cell trajectory $T_{u}^{C,syn}$ being perturbed, we should reach two goals: (1) $T_{u}^{C,syn}$ should have a high similarity with the real cell visit probability vector of $T_{u}^{C,act}$ to maintain data utility; and (2) $T_{u}^{C,syn}$ should have a low similarity with the perturbed cell visit probability vectors of cell trajectories of other users to preserve privacy. Then, we can model the two goals in the following two objective functions:

\[
\begin{align*}
O(p^{C}, p_{u}^{C,act}) &= \sum \text{sim}(p_{u}^{C,act}, p^{C}), \\
O'(p^{C}, \Psi) &= \frac{1}{\sum p_{v}^{C,syn} \text{sim}(p_{v}^{C,syn}, p^{C})}
\end{align*}
\]

(6)

where $p^{C}$ is a $N^2$-dimensional vector to be solved, $p_{u}^{C,act}$ is the cell visit probability vector of real cell trajectory $T_{u}^{C,act}$, and $\Psi = \{p_{v}^{C,syn} | v \neq u\}$ denotes the set of perturbed cell visit probability vectors of trajectories for other users. By solving the extremum for two functions in formula (6), we can obtain a solution $p^{C}$ that is the perturbed cell visit probability vector $p_{u}^{C,syn}$ corresponding to $T_{u}^{C,act}$.

Since $p^{C}$ is a cell visit probability vector, the sum of all dimensions in $p^{C}$ is equal to 1. Formula (6) has following constraint:

\[
\sum_{k=0}^{N^2-1} p^{C}[k] = 1.
\]

(7)

To preserve data utility, we restrict the lower bound and upper bound of each dimension in $p^{C}$. If a cell in a real cell trajectory $T_{u}^{C,act}$ in time slot $i$ is $C_{j}$, the actual scope of activities for a user $u$ is within $C_{j}$ and the adjacent eight cells of $C_{j}$. Therefore, if $C_{j}$ is perturbed to its 9 adjacent cells (including $C_{j}$ itself) while solving the multi-objective optimization problem in formula (6), the data utility for the trajectory of the user $u$ will not be largely lost. We can calculate the maximum count of locations from the 9 adjacent cells of $C_{j}$, and the count is divided by the number of locations in $T_{u}^{C,act}$. Then the $j$-th component of upper bound vector $U'$ of $p^{C}$ for $C_{j}$ is obtained. Therefore, after traversing each top-level cell $C_{j}$ ($1 \leq i \leq N^2$) corresponding to each cell in trajectory $T_{u}^{C,act}$, we obtain the upper bound vector $U'$ for each dimension of $p^{C}$, where the lower bound vector of $p^{C}$ is a zero vector.

We choose particle swarm optimization algorithm (PSO) to solve the problem in formula (6). Since a particle is the basic object in the iterative process of PSO, we need to integrate the cell visit probability vector $p^{C}$ into a particle. We will have following definition for a particle.

**Definition 10 (Particle).** A particle is a four-tuple $W = (p^{C}, V, \Lambda, F)$, where $p^{C}$ is the cell visit probability vector to be solved, $p^{C}$ denotes the position vector of a particle, $V$ is the speed vector of a particle, $F$ is the utility score function of a particle, which is determined by objective functions in formula (6), and $\Lambda$ is the degree of violation of constraint:

\[
\Lambda = \left(1 - \sum_{k=0}^{N^2-1} p^{C}[k]\right).
\]

(8)

The particle swarm optimization algorithm with a linearly decreasing inertia weight (PSO-IW) in the literature [16] is a commonly used version. The C-th round iteration of PSO-IW has two important steps: select and update. In the select step, the new local optimal particles and global optimal particles are chosen from all the particles in particle swarm $\Omega$, the historically global optimal particles and the historically local optimal particles. In the update step, the velocity vector $V$ and position vector $p^{C}$ of each particle are modified. However, when calculating the objective function value of each particle in the select step, we need to read the real cell visit probability vector from the cell trajectory dataset $D'$ that would lead to a potential privacy leakage. Therefore, we modified the select step and present a particle swarm optimization algorithm with differential privacy to enhance the security in solving the multi-objective optimization problem in formula (6).

5.4. The particle swarm optimization algorithm with differential privacy. In this subsection, we will describe the particle swarm optimal algorithm with differential privacy (PSO-DP) in detail.

5.4.1. The privacy budget for the C-th round iteration. In PSO-DP, the noise of differential privacy is added in the select step. Specifically, the original step select is changed into step em_select, which satisfies differential privacy by introducing an exponential mechanism into selecting new local optimal particles.

In PSO-DP, suppose the step em_select is executed $M$ times, and the privacy budget $\epsilon_{2}$ is divided into $M$ parts. When the number of iterations $C$ are small, the randomness of particles is strong and the difference between each particle in the particle swarm and the final solution is large. At this time, the particle swarm optimization algorithm can search the solution space thoroughly, and we do not need to allocate too much privacy budget to determine the global and local optimal particles earlier. With an increasing $C$, the particles in particle swarm $\Omega$ tend to be steady, and we need to allocate more privacy budget to reduce the randomness caused by differential privacy to avoid deteriorating the convergence of the particle swarm optimization algorithm again. Therefore, we use reciprocals of triangular numbers with the following series, which elegantly provides this property: $1/3, 1/6, \ldots, 2/M (M + 1)$, where the sum of the series converges to 1. In the series, the total privacy budget consumed by $M$ iterations is less than $\epsilon_{2}$. Next, we divide the remaining privacy budget $2\epsilon_{2}/(M + 2)$ into $M$ iterations, and then the privacy budget for the C-th iteration is computed as follows:
\[
\epsilon_2^C = \frac{2\epsilon_2}{(M - C + 1)(M - C + 2)} + \frac{2\epsilon_2}{M(M + 2)}
\]  

(9)

5.4.2. The utility score function for the exponential mechanism. In step \textit{em\_select}, the selection of local optimal particles are perturbed by an exponential mechanism. To select the local optimal particle with the highest probability by using an exponential mechanism, each particle needs to be evaluated by the utility score function \( F \). The evaluation of a particle is then determined by objective functions. Therefore, according to formula (6) we perturb each particle with an exponential mechanism by the utility score function as follows.

\[
F(P^C) = (1 - \gamma) \times \text{Rank}(O(P^C, P_{u}^{PC, act})) + \gamma \times \text{Rank}(O'(P^C, \Psi)),
\]

(10)

In formula (10), \( F(P^C) \) is the utility score for cell visit probability vector \( P^C \), and \( O \) and \( O' \) denote the values of the two objective functions when substituting \( P^C \) into formula (6). The function Rank denotes the rank for the objective function value \( P^C \) in all candidate particles. For example, \( \text{Rank}(O(P^C, P_{u}^{PC, act})) \) denotes the ascending rank of the objective function value \( (O(P^C, P_{u}^{PC, act})) \) in that of all candidate cell visit probability vectors.

In the formula (10) for \( F(P^C) \), the former part (before \(+\)) evaluates the preservation degree of the perturbing cell visit probability vector for the real cell trajectory \( T_{u}^{C, act} \) of user \( u \), i.e., the data utility, while the latter part (after \(+\)) evaluates the reduction degree of trajectory correlations for the perturbing cell visit probability vector \( P^C \) and perturbed cell visit probability vectors of other users, i.e., the data security. \( \gamma \) is a weight, and we define \( \gamma = 1/\sqrt{\epsilon_2} \). Then we have: (1) \( \gamma \) is inversely proportional to the privacy budget \( \epsilon_2 \) for PSO-DP. When \( \epsilon_2 \) is small, \( F(P^C) \) focuses on security, and when \( \epsilon_2 \) is large, \( F(P^C) \) focuses on data utility; (2) the maximum value of \( \gamma \) will not exceed 1; and (3) two square root operations over \( \epsilon_2 \) and \( \sqrt{\epsilon_2} \) avoid \( y \) being too small (i.e., \( \epsilon_2 \) is too large) or too large (i.e., \( \epsilon_2 \) is too small). Then, \( F(P^C) \) can balance the data utility and security to an appropriate extent.

According to the constraint condition in formula (7), when the constraint condition is satisfied, the trajectory correlation between the solution \( P^C \) of user \( u \) in equation (6) and the perturbed cell trajectories of other users can be reduced. Namely, the security for the output of PSO-DP would be increased.

To reduce the noise added to the particle swarm optimization algorithm, after the utility scores \( F \) for the cell visit probability vectors of all candidate particles are calculated, the utility scores \( F \) of all particles are normalized, and then the global sensitivity of the exponential mechanism is 1. We thus have following theorem.

\textbf{Theorem 2.} After the utility scores \( F \) in equation (10) for all candidate particles are normalized, the global sensitivity of queries based on \( F \) is 1.

\textbf{Proof.} Suppose the set of cell visit probability vectors in candidate particles is \( \Gamma = \{P^C_i\} \ i \in [1, |\Gamma|]\). The cell trajectory datasets \( D_1^C \) and \( D_2^C = D_1^C - \{T_{1}^{C}\} \) are neighbor datasets, where \( T_{1}^{C} \) is a cell trajectory. The sets of cell visit probability vectors for \( D_1^C \) and \( D_2^C \) are \( D_1^P \) and \( D_2^P \), respectively. According to equation (10), we can calculate the utility scores for all cell visit probability vectors as \( E = \{F_i\ i \in [1, |\Gamma|]\}\). Utility scores are the weighted sum of Rank\((\cdot)\). Then we have the denominator of the normalizing utility scores:

\[
NF = \sum_{i=1}^{|\Gamma|} F_i = \sum_{i=1}^{|\Gamma|} F(P^C_i) = \sum_{i=1}^{|\Gamma|} ((1 - \gamma) \\
\times \text{Rank}(O(P^C_i, P_{u}^{PC, act}))) + \gamma \sum_{i=1}^{|\Gamma|} \text{Rank}(O'(P^C_i, \Psi))
\]

(11)

\[
= (1 - \gamma) \sum_{i=1}^{|\Gamma|} \text{Rank}(O(P^C_i, P_{u}^{PC, act})) + \gamma \sum_{i=1}^{|\Gamma|} \text{Rank}(O'(P^C_i, \Psi))
\]

\[
= (1 - \gamma) \|\Gamma\| \frac{|\Gamma| + 1}{2} + \gamma |\Gamma| \frac{|\Gamma| + 1}{2} = \frac{|\Gamma|(|\Gamma| + 1)}{2}.
\]

According to the sets of cell visit probability vectors \( D_1^P \) and \( D_2^P \), as well as formula (10) and \( NF \), we have:

\[
\frac{\|F(P^C_{i}) - F(P^C_{j})\|}{NF_{i}} = \frac{1}{NF} \left\| F(P^C_{i}) - F(P^C_{j}) \right\|_{1}\]

\[
= \frac{1}{NF} \left\| (1 - \gamma) \left( \text{Rank}(O(P^C_i, P_{u}^{PC, act})))_{D_1^P} - \text{Rank}(O(P^C_j, P_{u}^{PC, act})))_{D_1^P} \right) \right\|_{1} + \gamma \left( \text{Rank}(O'(P^C_i, \Psi)))_{D_1^P} - \text{Rank}(O'(P^C_j, \Psi)))_{D_1^P} \right) \right\|_{1}\]

\[
\leq \frac{1}{NF} \left\| (1 - \gamma) |\Gamma| + \gamma |\Gamma| \right\| = \frac{1}{NF} |\Gamma| = \frac{2}{|\Gamma| + 1} \leq 1.
\]
Therefore, according to Definition 4 the global sensitivity of normalized utility score \( F \) is 1.

According to Theorem 2, when we add noise to utility score \( F \) with an exponential mechanism, the global sensitivity is 1.

5.4.3. PSO-DP algorithm. We modify the particle swarm optimization algorithm with differential privacy and obtain the new algorithm PSO-DP shown in Algorithm 1. The output of Algorithm 1 is the perturbed cell visit probability vector \( P_{u}^{new} \).

In Algorithm 1, the step \( em\_select \) that adds noise for PSO-DP to enhance the security of Algorithm 1 is shown in Algorithm 2. After \( P_{u}^{C,\text{act}} \) is perturbed in Algorithm 1, user \( u \) is added into \( \Psi \) to update \( \Psi \) when we use Algorithm 1 to perturb the cell visit probability vector of the next user.

In Algorithm 2, when selecting the local optimal particles, an exponential mechanism is introduced. We illustrate the formula following exponential mechanism to calculate the probabilities for particles in step 4. In step 5, we select a local optimal particle according to the probabilities, and thus the security is enhanced.

5.5. Trajectory Synthesis. After we obtain the perturbed cell visit probability \( P_{u}^{C,\text{syn}} \), we synthesize the perturbed location trajectory \( T_{u}^{new} \) for user \( u \).

We first obtain a top-level cell set \( \text{Set}^{C} \) from \( P_{u}^{C,\text{syn}} \), in which the visit probability of each cell is greater than 0. Then we randomly select a cell from \( \text{Set}^{C} \) for each time slot of \( T_{u} \), and from the cell we select a bottom-level cell with a maximum density of check-ins. We randomly generate a location in the bottom-level cell as the perturbed location. Then the perturbed location trajectory \( T_{u}^{new} \) is formed. This is the post-processing of RDPT and will not consume privacy budget and can preserve the privacy of individuals [39] because when we partition a cell into bottom-level cells we add Laplacian noise.

In the following, we illustrate Algorithm 3 to process all trajectories in a dataset. In Algorithm 3 we have two stages: the adaptive grid partition in line 2 - 8, and calling PSO-DP in line 14 to perturb trajectories in the original dataset one by one. The privacy budgets consumed in each stages are \( \varepsilon_{1} \) and \( \varepsilon_{2} \), respectively.

5.6. The Parameters and Convergence of RDPT. In Algorithm 3, several parameters will influence the solution of the multi-objective optimization problem in formula (6). These parameters are the population size \( \alpha \) of particle swarm \( \Omega \), the number of new global optimal particles selected in each iteration \( \beta \), and the parameter \( M \) that controls the maximum number of iterations. According to the literature [40, 41], since there are two objective functions and one constraint in formulas (6) and (7), we will select an appropriate value in the range [70, 300] for the population size \( \alpha \). Usually we let \( \beta = 1 \). We empirically select appropriate \( M \) to control the maximum number of iterations to make the PSO-DP convergent for each trajectory, usually we set the \( M \) in \([400,1000]\). When \( \varepsilon_{1} \) is larger, \( M \) is less. When \( \varepsilon_{1} \) is less, for example \( \varepsilon_{2} = 0.1 \), we need more iterations for PSO-DP(\( M \) is set 1000). Usually, for different datasets, these parameters should be analyzed and empirically adjusted to make the results better. Other parameters in PSO-DP, we did not adjust them.

It is difficult to prove the convergence of Algorithm 2 (the PSO-DP algorithm). Usually, as shown in Algorithm 1 when the difference of values of objective function between two iterations is less than \( 10^{-3} \) in several consecutive iterations and the number of iterations is larger than \( M/2 \), we consider the PSO-DP is convergent. The exponent mechanism will interfere the convergence of PSO-DP, but the general trend of PSO-DP is convergent. In fact, the randomness introduced by exponent mechanism may let the PSO-DP algorithm jump out of a local optimal solution and get a better local optimal solution. Even so, there are several trajectories not convergent. At this time, we can re-execute the Algorithm 2 for the trajectory, and then the PSO-DP usually will be convergent. In our experiments, the Algorithm 2 is convergent for all trajectories in \( D \).

5.7. Privacy Analysis. The need for data privacy appears in two different scenarios. One is the data collection scenario in which individuals regard data collectors are untrusted and send their check-ins with local differential privacy (e.g. voluntarily on social network sites). The second is the data releasing scenario in which datasets are released to a third-party research institute for analysis applications and differential privacy protection is used over the centralized datasets. RDPT solved the problem of privacy disclosure in the second scenario before differential privacy implementation.

Next, we prove that the RDPT satisfies differential privacy, we first prove that Algorithm 1 satisfy differential privacy.

**Theorem 3.** Algorithm 1 satisfies \( \varepsilon_{2} \)-differential privacy.

**Proof.** We first analyze Algorithm 2, in which the important step is selecting a local optimal particle. When Algorithm 2 is called in each iteration, the exponential mechanism selects a new local optimal particle in the particle swarm, and the corresponding candidate particle set \( \Omega \) disjoins from that of other iteration calls. According to the parallel composition property of differential privacy in Definition 6, the privacy budget consumed by selecting a new local optimal particle for each particle is \( \varepsilon_{2}^{C} \). Therefore, the process for selecting local optimal particles in Algorithm 2 is \( \max\{\varepsilon_{1}^{C}\} = \varepsilon_{2}^{C} \)-differential privacy.

For the selection of global optimal particles, we only select \( \beta \) global optimal particles from the union of the newly local optimal particles and the historically global optimal particles; we do not need to add noise anymore. Therefore, Algorithm 2 satisfies \( \varepsilon_{2}^{C} \)-differential privacy as a whole.

In Algorithm 1, Algorithm 2 is sequentially called with maximum \( M \) times; hence, the privacy budget consumed in Algorithm 1 is no more than \( \sum_{M=1}^{M} \varepsilon_{2}^{C} = \varepsilon_{2} \)-differential privacy, according to the sequential composition property in Definition 5.

\( \square \)
Input: privacy budget $\epsilon$, real cell visit probability vector $P_{C}^{\alpha\text{syn}}$ for current user $u$, the set of other users $\Psi = \{v | v \neq u\}$ that the cell visit probability vectors are perturbed by RDPT (If $u$ is the first user to be processed, then $\Psi = \emptyset$), upper bound vector $U'$ for particle position vector, the number of global optimal particles $\beta$, population size $\alpha$, and the maximum number of iterations $M$.

Output: perturbed cell visit probability vector $P_{C}^{\alpha\text{syn}}$ of a user $u$

(1) $C = 1$

(2) Initialize particle swarm $\Omega = \{W_{i} = (P_{C}^{\alpha\text{syn}}, V_{i}, \Lambda, F_{i}) | 1 \leq i \leq \alpha\}$, where $P_{C}^{\alpha\text{syn}}$ is randomly initialized such that each component of $P_{C}^{\alpha\text{syn}}$ is not higher than the upper bound of its corresponding dimension in $U'$ and the sum of all dimensions of $P_{C}^{\alpha\text{syn}}$ is 1. $V$ is a zero vector. $\Lambda$ is calculated by equation (8) and $F_{i}$ is calculated by equation (10), respectively. When $F_{i}$ is calculated, the values $O_{i}, O_{i}'$ of the objective function are calculated by equation (6).

(3) $\epsilon_{2}^{\alpha} = 2\epsilon_{2}/(M - C + 1)(M - C + 2) + 2\epsilon_{2}/(M + 2)$

(4) Initialize two sets: $\Omega^{\ast} = \Omega$ to save global optimal particles, $\Theta$ to save global optimal particles.

(5) $\Omega^{\ast}, \Theta = \text{em\_select}(\epsilon_{2}^{\alpha}, \Omega^{\ast}, \Theta, \alpha, \beta)$ //call algorithm 2

(6) $u_{\text{min}} = 0.4, u_{\text{max}} = 0.9$ //the bounds of the weight $w$ for updating the velocity of the particle and are obtained from the literature [16].

(7) $c_{1} = c_{2} = 2$ //two accelerating constants obtained from the literature [16].

(8) select the global best particle from the set $\Theta$ and save the values of the objective function of the particle to $O_{1}$ and $O_{1}'$

(9) while $C \leq M - 1$ do

(10) $w = (w_{\text{max}} - w_{\text{min}})/(M - C)/M + w_{\text{min}}$ //the inertia weight computed according to the literature [16].

(11) for $i = 0$ to $\alpha - 1$ do

(12) let $W_{i} = [\Omega[i], W_{i}'\text{Best}], i = 1, 2$, and select the global best particle $W_{i}'\text{Best}$ from the set $\Theta$

(13) // $P_{C}^{\alpha\text{syn}}$ and $P_{C}^{\alpha\text{syn}}$ are the position vectors of $W_{i}'\text{Best}$ and $W_{i}'\text{Used}$, respectively

(14) $F_{i} = W_{i}'\text{Best} - F_{i}$

(15) $V_{i} = \omega \times V_{i} + c_{1} \times \text{rand}(0, 1) \times (P_{C}^{\alpha\text{syn}} - F_{i}) + c_{2} \times \text{rand}(0, 1) \times (P_{C}^{\alpha\text{syn}} - P_{C}^{\alpha\text{syn}})$

(16) $P_{C}^{\alpha\text{syn}} = P_{C}^{\alpha\text{syn}} + V_{i}$

(17) recompute $\Lambda$ and $F_{i}$ according to formulas (8) and (10), respectively. When $F_{i}$ is calculated, the values $O_{i}, O_{i}'$ of objective function are calculated by formula (6).

(18) end for

(19) $C = C + 1$

(20) $\epsilon_{2}^{\alpha} = 2\epsilon_{2}/(M - C + 1)(M - C + 2) + 2\epsilon_{2}/(M + 2)$

(21) if $C = M$ then

(22) $\beta = 1$

(23) end if

(24) $\Omega^{\ast}, \Theta = \text{em\_select}(\epsilon_{2}^{\alpha}, \Omega^{\ast}, \Theta, \alpha, \beta)$ //call algorithm 2

(25) compare the values of objective function $O_{1}$ and $O_{1}'$ of the global best particle from $\Theta$ with $O_{1}$ and $O_{1}'$

(26) if abs($O_{1} - O_{2}$) < $10^{-3}$ and abs($O_{1}' - O_{2}'$) < $10^{-3}$ then

(27) break

(28) end if

(29) end while

(30) Assign the position vector in $\Theta[0]$ to $P_{C}^{\alpha\text{syn}}$

(31) Return $P_{C}^{\alpha\text{syn}}$

Algorithm 1: PSO-DP.

Theorem 4. Algorithm 3 (RDPT) satisfies $\epsilon$-differential privacy.

Proof. Suppose the total privacy budget is $\epsilon$. We split it into two parts in RDPT, denoted by $\epsilon_{1}$, and $\epsilon_{2}$, $\epsilon_{1}$ is for adaptive grid partitioning, while $\epsilon_{2}$ is for perturbing the cell visit probability vector in Algorithm 3.

In line 2-8 in Algorithm 3, we add Laplacian noise to the result of formula (3) to ensure the privacy of dividing a top-level cell into bottom-level cells. As shown in Theorem 1, the sensitivity of formula (3) is 1, which means that the added Laplacian noise follows a distribution of Lap($1/\epsilon_{1}$). Therefore, the adaptive grid partitioning in Algorithm 3 satisfies $\epsilon_{1}$-differential privacy.

When we perturb a real trajectory in a dataset, Algorithm 1 is called in Algorithm 3 in line 14. The Algorithm 1 only reads the real trajectory of one user and perturbs cell visit probability vector. According to the sequential composition property in Definition 5 and Theorem 3, the process of perturbing a real trajectory satisfies $\epsilon_{2}$-differential privacy for Algorithm 1. When next real trajectory is processed, the real trajectory that the Algorithm 1 reads is disjointed with previous trajectory. Therefore, when we perturb real trajectories in a dataset, although Algorithm 1 are called $N'$ times, perturbing real trajectories in a dataset still satisfies ($\epsilon_{2}$)-differential privacy according to parallel composition property in Definition 6.

As a result, Algorithm 3 (RDPT) satisfies $\Sigma_{i=1}^{2} \epsilon_{i} = \epsilon$-differential privacy according to the sequential composition property in Definition 5.

RDPT not only satisfies differential privacy theoretically but also reduce trajectory correlation, and then the privacy is then preserved. \qed
6. Experiments and Analysis

Extensive experiments are conducted on three real datasets to verify the effectiveness of RDPT. By adjusting the total privacy budget $\epsilon$, we compare RDPT with several methods over five metrics to verify that RDPT achieves almost the same data utility and better security. We also verify the stability of RDPT over two specific implementations for trajectory correlation.

6.1. Datasets. We use three real datasets in our experiments: Gowalla [42], Yonsei [10] and Geolife [8]. The three datasets denote different applications for releasing trajectory data offline.
Gowalla is a location-based social networking website where users share their locations by checking-in. The dataset Gowalla-New York (GNY) in our experiments is a subset of check-in records in the Gowalla dataset, in which latitude and longitude coordinates are located in the city of New York. The dataset depicts the relationships and daily activities of users in real social networking. In data preprocessing, the locations where latitude, longitude or timestamp (time slot) are null are deleted from these two datasets. In addition, if the number of locations in a trajectory is less than 80, the trajectory is also deleted from the datasets because the trajectory correlation between two users are formed in a relative long time. Thus the experiments over the dataset can simulate a real scene and patterns in releasing a trajectory dataset offline.

The Yonsei is a dataset collected by Yonsei University, Seoul, Korea [10]. The trajectories of nine graduate students at Yonsei University were collected using the mobile location service application SmartDc over two months in 2011. The dataset Yonsei-Seoul (YSO) is a subset of locations in the Yonsei dataset in which latitude and longitude coordinates are located in Seoul. Different from the GNY dataset, all users in Yonsei dataset have relationships and the data are denser than GNY dataset. Yonsei dataset also records the behavior of users, and simulate a real scene in releasing trajectories offline.

The Geolife dataset contains trajectories from 182 users over three years. Locations are represented by GPS latitude and longitude coordinates, the date and time when the user visits the locations. We selected trajectories of 33 users from October 23 to November 6, 2008 and all the locations are in Beijing. For each user, we connect multiple trajectories of the user into one trajectory, then we obtain a datasets (GEO) for our experiments. The data are much more dense than other two datasets.

The statistical information of the dataset after preprocessing is summarized in Table 2.

6.2. The methods to be compared. We used DPT [6], TGM [7], and AdaTrace [8] as three comparison methods. These methods consider the correlations within a trajectory. In addition, these methods are privacy-preserving approaches with differential privacy. Although the correlations within a trajectory in DPT, TGM and AdaTrace are different from the trajectory correlation between trajectories in RDPT, we still choose them for a comparison because their aim is to generate synthetic trajectories that can provide efficient protection while publishing a large number of trajectories offline. As for the methods in literature [9, 10], these methods only perturbed two trajectories with the same length, and require the two trajectories are not too long. It is difficult to find a real dataset for the experiments to compare with them. So we do not compare with the two methods. For the comparison approaches, we use the parameters recommended by the literature or the parameters that improve the experimental results.

6.3. The metrics. We select five metrics in our experiments to evaluate the data utility and privacy of RDPT. The metrics for evaluating the data utility are the Jensen-Shannon divergence of the location visit probability vector (JS) [6–8], the Kendall coefficient of the location visit probability (Ken) [8], and the query error (Qerr) [8]. The ability to a resist Bayes inference attack (Qerr) [8] and the ability to protect the trajectory correlation (δ) are two metrics for privacy.

The Jensen-Shannon divergence (JS) of the location visit probability vector. This metric evaluates the degree of preserving the location visit frequency vectors between two trajectories in the perturbed dataset. The smaller the value is, the better preserved the spatial distribution feature is, and the smaller the utility loss of the dataset is. The equation for calculating JS is:

\[
JS = \frac{1}{2} KL(P, P + Q + 2) + \frac{1}{2} KL(Q, P + Q + 2),
\]

where \(P, Q\) denote global location visit probability vectors in the real dataset and perturbed dataset, and KL(·) is the Kullback-Leibler divergence.

The Kendall coefficient (Ken) of the location visit frequencies. The Kendall coefficient (Ken) evaluates the statistical results for any pair of locations in the datasets, such that the order of the visit frequencies of the two locations are not changed in the real dataset or perturbed dataset. The larger the value is, the more relative "hot" locations are preserved, and the higher the data utility is. To describe the metric, we first define "order preserving". "Order preserving" means that for two locations \(\ell_1\) and \(\ell_2\), if the visit frequencies of \(\ell_1\) are larger than (or less than) that of \(\ell_2\) in the real dataset, then we say that \(\ell_1\) and \(\ell_2\) is a pair of "order preserving" locations. Suppose the number of "order preserving" pairs is \(X\) and the number of "order not preserving" pairs is \(X'\) over the real dataset and perturbed dataset, respectively. The equation for calculating Ken is:

\[
Ken = \frac{X - X'}{n(n - 1)/2},
\]

where \(n\) is the number locations in a dataset.

The query error (Qerr). We define a query \(q\) as follows: counting the number of trajectories passing through a specific region \(R\) in a dataset. Suppose \(q(D)\) denotes the query results over a dataset \(D\), and the query error (Qerr) is calculated as follows:

\[
Qerr = \frac{1}{|R|} \sum_{r \in R} \|q(D) - q(D^{sys})\|_1 / \max(q(D), b),
\]

where \(b\) controls the impact of the query results in extreme cases. In our experiments, we let \(b = 1\% \times |D|\) [8]. \(R\) is a set of 500 regions selected uniformly and randomly to avoid the query error influenced by accidental abnormal query results. The less Qerr is, the more number of trajectories is preserved that are traversed in different regions of the dataset, and the more conducive the dataset is for commercial block planning and other application scenarios.
6.4. Comparison of data utility. We will first compare relation calculated by Pearson correlation coefficient and Pearson and RDPT-Cos are the results for trajectory correlation in RDPT. In following results, the RDPT-efficient and cosine similarity to calculate the trajectory correlation is almost equivalent to DPT. However, the difference of Qerr, and RDPT-Cos are better than TGM and AdaTrace, and a much better than TGM. From Figure 3(b), RDPT-Pearson is a slight worse than DPT and are very close to AdaTrace, but much better than TGM. From Figure 3(b), RDPT-Pearson and RDPT-Cos are better than TGM and AdaTrace, and a slight worse than DPT. However, the difference of the Qerr, Qerr s between RDPT-Pearson is less than 0.04. Therefore, the results of Ken over Geolife dataset when \( \epsilon = 1 \) and the difference of Ken s between RDPT-Pearson and AdaTrace all focus on features of global spatial distribution, for RDPT are better or almost equivalent to other three methods.

In the following, we compare RDPT with DPT [6], TGM [8], and AdaTrace [9] over above five metrics, including the Jensen-Shannon divergence (JS), the Kendall coefficient (Ken), the query error (Qerr), the ability to resist a Bayes inference attack (\( \delta \)), and the ability to protect trajectory correlation (\( \delta \)) over three datasets.

### 6.4. Experimental results and analysis

We implement RDPT in Java, and the experiments are conducted on a computer with a 3.60 GHz Intel(R) Core(TM) i5-8600K, and 16 GB of memory. We adjust the parameters in our experiments to guarantee the Algorithm 1 is convergent. For each experiment, we repeat several times and obtained the average values for the five metrics. The parameters for our experiments are listed in Table 3.

In the following subsections, we compare the data utility when \( \epsilon \) is increasing. We use the Pearson correlation coefficient and cosine similarity to calculate the trajectory correlation in RDPT. In following results, the RDPT-Pearson and RDPT-Cos are the results for trajectory correlation calculated by Pearson correlation coefficient and cosine similarity respectively.

#### 6.4.1. Comparison of data utility

We will first compare RDPT with DPT, TGM, and AdaTrace over three datasets for three data utility metrics: JS, Ken, and Qerr.

The experimental results. Figure 1, Figure 2 and Figure 3 show the results of RDPT, DPT, TGM and AdaTrace on different \( \epsilon \) over three datasets for three different metrics: JS, Ken and Qerr, respectively. From over the perturbed dataset \( D' \), and the attacker also knows the vector. We will evaluate the difference of \( P(Z|D) \) and \( P(Z|D') \) using the Jensen-Shannon divergence as follows.

\[
\vartheta = \max\{JS(P(Z|D), P(Z|D')|Z_i \in Z)\}.
\]

The less \( \vartheta \) is, the less the difference between \( P(Z|D) \) and \( P(Z|D') \) is, the less privacy the attacker infers, and the higher the security is. To avoid being influenced by the specific sensitive area \( Z \), we regard the maximum value of the calculation results of all regions in the sensitive region set \( Z \) as the final result.

The ability to protect the trajectory correlation (\( \delta \)). The metric evaluates the protection degree of trajectory correlation for a privacy-preserving method. The larger the value is, the better the trajectory correlation is protected, and the lower the probability that an attacker infers the social relationship through trajectory correlation is. The metric is:

\[
\delta = \frac{\sum_{T_i^C \in D'} \sum_{j \in T_i^C, j \neq i} S(T_i^C, T_j^C) - \sum_{T_i^C \in D'} \sum_{j \in T_i^C, j \neq i} \sum_{T_i^C \in D'} \sum_{j \in T_i^C, j \neq i} S(T_i^C, T_j^C)}{|D'|(|D'|-1)/2}
\]

Table 2: Detail information about the Dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GNY</th>
<th>YSO</th>
<th>GEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>New York</td>
<td>Seoul</td>
<td>Beijing</td>
</tr>
<tr>
<td>number of user</td>
<td>328</td>
<td>9</td>
<td>33</td>
</tr>
<tr>
<td>number of check-in</td>
<td>67217</td>
<td>36851</td>
<td>770730</td>
</tr>
<tr>
<td>average number of check-in per user</td>
<td>204.9</td>
<td>4094.5</td>
<td>23355.5</td>
</tr>
</tbody>
</table>

In Figure 1, the results of JS for RDPT are better than other three methods. In Figure 2, the results of Ken. Ken for RDPT are better than other three methods over GNY and YSN datasets, but the results of Ken for RDPT-Pearson is less than Adatrace over Geolife dataset when \( \epsilon = 1 \) and the difference of Ken s between RDPT-Pearson and AdaTrace is less than 0.04. Therefore, the results of Ken over Geolife for RDPT are better or almost equivalent to other three methods.

In Figure 3(a), the results of Qerr, Qerr for RDPT are a slight worse than DPT and are very close to AdaTrace, but much better than TGM. From Figure 3(b), RDPT-Pearson and RDPT-Cos are better than TGM and AdaTrace, and a slight worse than DPT. However, the difference of the Qerr, Qerr s between DPT and RDPT is about 1%. In Figure 3(c), the results of RDPT are a slight worse than TGM, and are better than AdaTrace. However, the results of RDPT-Pearson are better than DPT and the results of RDPT-Cos are almost equivalent to DPT. After all, the Qerr, Qerr s for RDPT, DPT, and TGM are approximate.

From Figure 1 to Figure 3, RDPT preserves almost equivalent data utility to the comparison methods on two different implementations of trajectory correlation.

Analysis of the results. RDPT perturbs the trajectories one by one, and preserves the feature of spatial distribution of locations for each trajectory. Although DPT, TGM, and AdaTrace all focus on features of global spatial distribution, they replace the spatial distribution feature of each trajectory with the common spatial distribution feature of all trajectories. Then, when the spatial distribution feature is
6.4.2. Comparison of privacy. The experimental results.

Since the ability to protect trajectory correlation ($\delta$) will be influenced by the specific implementations of trajectory correlation for DPT, TGM, and AdaTrace, then we show the results of DPT, TGM, and AdaTrace with different trajectory correlation for Pearson correlation coefficient and cosine similarity. The experimental results are shown in Figure 4 and Figure 5 with different privacy budgets $\varepsilon$ over three datasets and two metrics: the ability to resist Bayes inference attacks ($\bar{d}$) and the ability to protect trajectory correlation ($\delta$), respectively. As shown in Figure 4, for $\bar{d}$, RDPT-Pearson and RDPT-Cos are better than DPT, TGM, and AdaTrace on different privacy budgets $\varepsilon$. The results indicate that RDPT

generalized, JS and Ken metrics are worse than or equivalent to RDPT. Moreover, as shown in the multi-objective functions in (6), the first part (i.e., $O(PC, PC, \text{act})$) is formulated for preserving the cell visit probability vectors of the trajectories. After solving it by PSO-DP, we can obtain its extremum and achieve the purpose, thus preserving a better JS and Ken metrics than DPT, TGM, and AdaTrace. As for the Qerr metric, the post-processing in step 3 introduced noise, then the query for the number of trajectories that are traversed in different regions would be changed slightly, then the Qerr, Qerr metric is a slight worse than DPT or TGM. As a result, the data utility for RDPT is almost equivalent to DPT, TGM, and AdaTrace.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>GNY</th>
<th>YSO</th>
<th>GEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>the scale N for dividing top-level cells</td>
<td>18</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>proportion of $\varepsilon_1$ and $\varepsilon_2$ over total privacy budget</td>
<td>0.2, 0.8</td>
<td>0.2, 0.8</td>
<td>0.2, 0.8</td>
</tr>
<tr>
<td>population size $\alpha$ of particle swarm on PSO-DP</td>
<td>140</td>
<td>140</td>
<td>150</td>
</tr>
</tbody>
</table>

Table 3: Parameters in the experiments.

Figure 1: Comparison of JS with DPT, TGM and AdaTrace. (a) GYN. (b) YSN. (c) GEO.
Figure 2: Comparison of Ken with DPT, TGM and AdaTrace. (a) GYN. (b) YSN. (C) GEO.

Figure 3: Continued.
Figure 3: Comparison of $Q_{err}$ with DPT, TGM and AdaTrace. (a) GYN. (b) YSN. (C) GEO.

Figure 4: Comparison of $\theta$ with DPT, TGM and AdaTrace. (a) GYN. (b) YSN. (C) GEO.
could preserve the features of Markov transfer matrix in sensitive regions on datasets better than comparison methods, and RDPT could avoid the leakage of privacy in sensitive areas after the dataset being perturbed. For δ, the results of RDPT are better than DPT, TGM, and AdaTrace. In addition, the smaller ε is, the better the metric is.

Analysis of the results. As the metric ϑ reflects the spatial feature of the Markov transfer matrix on trajectories, and the results from Figure 1 to Figure 2 show that RDPT better preserves the location visit probability vector (i.e., spatial feature). Therefore, from the aspect of the privacy metric δ, the metric ϑ is also better than compared methods. For the metric δ, since RDPT focuses on preserving the spatial distribution feature on the trajectory of each user, the differences for the spatial distribution features between perturbed trajectories are large. While DPT, TGM and AdaTrace preserve the global spatial distribution features of all users, they generate perturbed trajectories by constructing mobile models based on global spatial distribution features. Therefore, the spatial distribution features between perturbed trajectories are similar in the perturbed dataset, so the average value for trajectory correlations between perturbed trajectories and original trajectories has less reduction. Hence, their δ performance is worse than RDPT. Moreover, as shown in the multi-objective functions in equation (6), the second part (i.e., $O'(P^C, \Psi)$) is formulated for protecting the trajectory correlation of different trajectories. After solving the optimization problem via PSO-DP, we can obtain its extremum and achieve the goal, which means that the trajectory correlation is well protected (i.e., δ). Consequently, RDPT naturally outperforms DPT, TGM and AdaTrace over δ.

From the Figure 5, the results of δ over Geolife dataset for RDPT is not better than GNY and YSN, the reason is that the GNY and YSN datasets are for real social networking and some users have relationships, then trajectory relations between users are higher, while the users in Geolife dataset usually have not relationships in social networking, therefore the descent degree of trajectory correlation is not obvious.

![Figure 5: Comparison of δ with DPT, TGM and AdaTrace. (a) GYN. (b) YSN. (C) GEO.](image-url)
7. Conclusion
In this paper, we proposed a differentially private trajectory publication method, named RDPT, to protect the trajectory correlation. We designed a multi-objective optimization problem that aims to reduce the trajectory correlation between a given trajectory and other trajectories. We integrated the optimization problem into RDPT and solved it through a modified particle swarm optimization algorithm with differential privacy. The experimental results on three real datasets show that RDPT achieves almost equivalent data utility to existing methods, as well as on two different specific implementations of trajectory correlation. Moreover, RDPT achieves a better privacy insurance than existing methods, and RDPT is more suitable for preserving privacy of long trajectories. For the dense dataset like Geolife which has overlength trajectories, how to improve our method, is our future work.

Conflicts of Interest
The authors declare that they have no conflict of interest.

Acknowledgments
This research is supported by National Science Foundation of China (the project number is 61772215) and the project supported by Wuhan Science and Technology Bureau (the project number is 201801040101274). This work is also the extension of the paper “CTP: Correlated Trajectory Publication with Differential Privacy” in 2021 IEEE the 6th International Conference on Computer and Communication Systems.

References


