A Differential Privacy-Based Privacy-Preserving Data Publishing Algorithm for Transit Smart Card Data

Yang Li, PhD
Associate Professor, Guangdong University of Technology, Guangzhou, China. 510006
Visiting Scholar, Missouri University of Science and Technology.
1401 N Pine St, Rolla, MO 65401, USA. liy6@mst.edu

Dasen Yang, Graduate student
Guangdong University of Technology, Guangzhou, China. 510006.
Email: 717805648@qq.com

Xianbiao Hu, PhD (Corresponding Author)
Assistant Professor, Department of Civil, Architectural and Environmental Engineering
Missouri University of Science and Technology.
1401 N Pine St, Rolla, MO 65401, USA. xbhu@mst.edu

Abstract

This manuscript is focused on transit smart card data and finds that the release of such trajectory information after simple anonymization creates high concern about breaching privacy. Trajectory data is large-scale, high-dimensional, and sparse in nature and, thus, requires an efficient privacy-preserving data publishing (PPDP) algorithm with high data utility. This paper describes the investigation of the publication of non-interactive sanitized trajectory data under a differential privacy (DP) definition. To this end, a new prefix tree structure, an incremental privacy budget allocation model, and a spatial-temporal dimensionality reduction model are proposed to enhance the sanitized data utility as well as to improve runtime efficiency. The developed algorithm is implemented and applied to real-life metro smart card data from Shenzhen, China, which includes a total of 2.8 million individual travelers and over 220 million records. Numerical analysis finds that, compared with previous work, the proposed model outputs sanitized dataset with higher utilities, and the algorithm is more efficient and scalable.

Keywords: Privacy-Preserving Data Publishing (PPDP), Differential Privacy (DP), Transit Smart Card, Trajectory Data;

Highlights:

- Quantitatively measure the privacy breach risks of transit smart card data
- A privacy-preserving data publishing (PPDP) algorithm is proposed
- The proposed algorithm outperforms two previous models on data utility and runtime efficiency

1. Introduction

With the rapid advancement of information and communication technologies (ICT), a variety of data collection methods have been developed to collect information on individual traveler's spatial-temporal movement. Commonly seen technologies include the Global Positioning System (GPS), social medias, smart card or IC card and so on. These data carry rich information on traffic conditions and traveler's activity patterns, and have been widely shared and used by researchers and transportation practitioners for various purposes.

Privacy breaches occur when users are re-identified from anonymous data. A prevailing assumption, nowadays, is that if the attributes that carry personal information (such as names and addresses) are removed before sharing, concerns about privacy leakage can be eliminated. However, it has been proved, in many fields, that the anonymity of personal information removal does not effectively protect privacy. For example, research has shown that 87% of the population in the United States have reported characteristics that likely made them unique, based only on a 5-digit ZIP code, gender, and date of birth (Sweeney 2000). This means that, even if an individual's name and address are removed, there is still a way to identify that person based on a combination of only a few attributes.

Privacy issues have also been a major concern in transportation engineering, as transportation datasets usually capture each individual traveler's spatial-temporal movements and, as a common practice, to make them publicly available after some simple attempts at anonymity. In this manuscript, we focus on data collected by a smart card (or IC card) that record the payment history of travelers who boarded and/or alighted from transit vehicles in Shenzhen, China. The dataset being analyzed includes a total of 2.8 million different travelers and over 220 million records. One would think that, with merely two boarding/alighting records for each trip, and without including personal information (such as names, home addresses, and dates of birth), such data would not impose a privacy concern. However, our analysis shows that, if a traveler's two travel records are known, and by using subway station names and departure times (with an accuracy of 10 minutes), 30.7% of users can be uniquely identified even though their personal information has been removed from the original dataset.

Figure 1 presents part of June 7, 2016, Shenzhen metro smart card dataset, which includes anonymous ID and ride records. Each line includes an anonymous identifier for the passenger, part of the trajectory records, and sensitive information that can be inferred from historical trajectories (such as home and work address). For example, as shown in the second line, a user with a pseudo-identifier ID 20016755 checked into "Bu Xin" station at 09:24am, and then "Fu Tian" station at 09:52am. The red line in Figure 1 represents the background knowledge owned by the attacker. The green line represents the sensitive information that an attacker may obtain. If an attacker has already known Alice has traveled to "Bu Xin station" on that day, around 7:20am-7:30am (i.e. with an accuracy of 10 minutes), and to "Long Cheng Center" station (on the same day) around 8:09am -8:19am, Alice's unique ID can easily be found to be 20015461 as she is the only passenger with these two travel records in the dataset. With this information, the attackers can discover all historical travel records for Alice, and use them to infer sensitive personal information (such as approximate home and work addresses and other living habits).

The degree for a privacy breach increases when more background knowledge of the trip or traveler becomes available. For example, if the attacker already knows that Frank has traveled to "Gangxia station" on June 7, 2016, at around 19:00-21:00 (i.e. with an accuracy of 2 hours), they are not sure if his identification number is 20160553 or 20099459. However, if they know that Frank rode on a bus right after the subway, then Frank's unique ID can easily be identified as 20099459. From our experiment, if we have a passenger's background information on a bus transfer, then the likelihood of him/her being uniquely identified in the dataset will increase to 41.4%. In other words, almost half of the people using

smart cards are identifiable and an attacker can use such information to view an individual's complete travel history in the dataset.

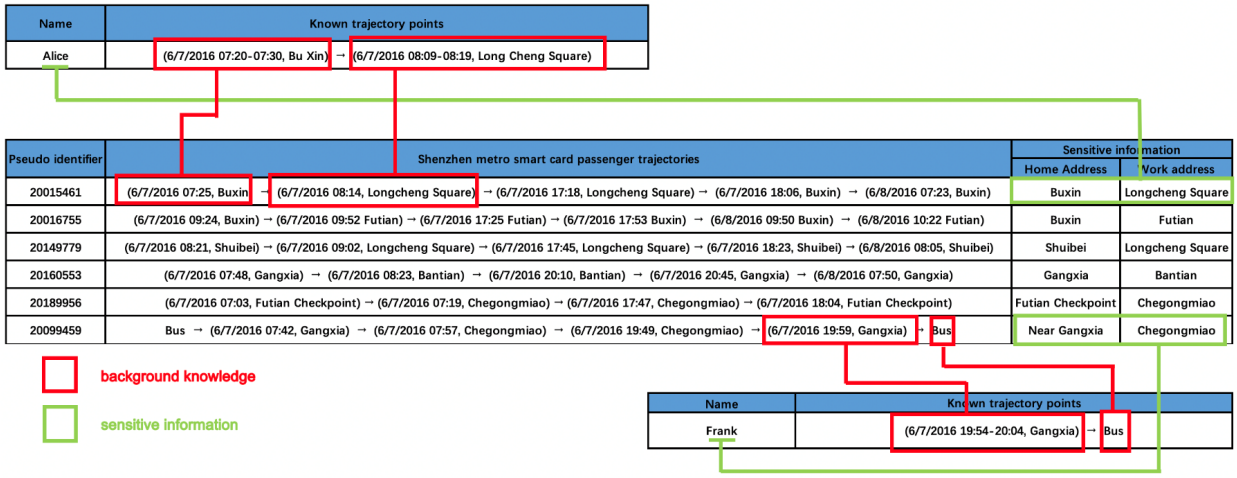

Figure 1. Examples of background knowledge attacks.

A differential privacy-based privacy protection algorithm for a transit smart card data is described in this manuscript. When compared with existing models, such as SeqPT (Chen et al. 2012) and SafePath (Khalil et al. 2018), it improves data utility while enhancing algorithm efficiency at the same time. Transit trajectory data is unique in a sense that it is large-scale and sparse in nature, as well as high-dimensional, as it includes not only spatial but also temporal information. The contributions of the proposed algorithm are mainly focused on the following three aspects.

- A **new prefix tree structure** without taxonomy tree as sublevel is developed to effectively utilize the privacy budget, with a goal of improving sanitized data utility. Previous research often built taxonomy trees for time and location dimensions at each level, which results in wasting the privacy budget. A new prefix tree, without a taxonomy tree at each level is proposed to save the privacy budget which, subsequently, will lead to data utility improvement. More budget can now be used to build a prefix tree with less noise added.

- An **incremental privacy allocation mechanism** is designed to improve sanitized data utility. Existing research allocates a privacy budget that is equal on each layer of the tree. However, due to the nature of the tree structure, as the depth increases, the number of nodes in each layer decreases and the random noise generated by the same amount of privacy budget becomes more significant. The proposed model reduces the impact of the noise on the prefix tree, with the same privacy budget amount, and brings a higher data utility, as compared with that of previous works.

- A **spatial-temporal domain reduction model** is developed to improve runtime efficiency. This is achieved by effectively filtering unreachable nodes, as the prefix tree grew and, thus, significantly reduce the computational workload. Compared with sequence data without the timestamp adopted in previous research, such as (Chen et al. 2012), the data investigated in this manuscript includes both timestamp and location information. A differential privacy model is needed that meets the randomness computation requirement when a prefix tree grows all possible timestamp and location combinations in the value domain. As such, the computation workload would increase dramatically and make critical efficiency improvement.

This paper is organized as follow. Work related to various privacy protection models is reviewed in Section 2. Some applications of privacy protection methods in transportation engineering are also summarized.

Section 3 introduces some preliminary concepts and formalizes the problem. Section 4 presents the proposed algorithm. The developed algorithm is implemented and compared with existing models in Section 5. Section 6 concludes this paper along with some discussion of future work.

2. Literature Review

With the rapid advancements in data collection technologies and applications, privacy-protected data mining has quickly attracted research attention, mainly in computer science, whereas algorithm development and studies of their applications in transportation have been very limited. In this section, related work is reviewed in two categories based different privacy principles. The first focuses on data indistinguishability, with k-anonymity and ℓ-diversity as the representative methods, and the second focuses on data uninformativeness, with differential privacy as the representative method.

In distinguishability models, private trajectories are published after anonymizing sensitive information. Generalization, which replaces individual values of attributes with a broader category, and suppression, which removes sensitive records from a dataset to meet specified anonymity constraints, are the most widely-used anonymization mechanisms. For generalization methods, Nergiz et al. (2008) redefined the notion of k-anonymity for sequences of spatial-temporal points, and released a randomly generated set of representative trajectories. A generalization-based k-anonymity approach was applied to trajectory data for the first time. Abul et al. (2008) proposed (k, δ)- anonymity for data publishing in a moving objects databases (MOBs) by considering the inaccuracy of sampling and positioning systems, where δ represents a possible location imprecision. Similarly, Yarovoy et al. (2009) considered timestamp as a fixed quasi-identifier (QID) attribute for all MOBs to avoid the combining of different anonymization groups by an attacker. Monreale et al. (2010) proposed a method for achieving anonymity by defining a transformation of the original GPS trajectories based on spatial generalization and k-anonymity. The novelty relied on finding a generalization scheme that depended directly on the input trajectory dataset instead of a fixed grid hierarchy. Hu et al. (2010) proposed a new generalization paradigm, called local enlargement, for a given sensitive event dataset, which guaranteed that user locations were enlarged enough so that each event was covered by at least k users. Virtual Trip Line (VTL) (Hoh et al. 2008) represents a concept of geographic marker that is placed to avoid specific privacy sensitive locations which allow aggregating and cloaking several location updates based on trip line identifiers, without knowing the actual geographic locations of these trip lines. Hoh et al. (2012) proposed a traffic monitoring system design based on VTLs, and Sun et al. (2013) proposed a VTL zone system for privacy protection in fine-grained urban traffic modeling applications.

For suppression methods, Terrovitis and Mamoulis (2008) defined an attacking model, in which different adversaries had different background knowledge, as a set of projections on a trajectory dataset. A greedy method was proposed that iteratively transformed long and detailed projections into smaller and simpler ones to suppress selected locations from the original trajectories until a privacy constraint was satisfied. Fung et al. (2009) proposed a *LKC*-privacy definition that could avoid attacking identity linkages and attribute linkages. The model transformed a raw dataset into an anonymous one by a sequence of suppressions. Based on that, Chen et al. (2013) first introduced local suppression to achieve a tailored privacy model for trajectory data anonymization, which allowed the adoption of various data utility metrics for different data mining tasks. Cicek et al. (2014) proposed a p –confidentiality model which centered on the probability of a user visiting a sensitive location with a p input parameter to ensure location diversity.

Data privacy-related research and applications in the transportation area have been very limited, and primarily belong to indistinguishability models. Ghasemzadeh et al. (2014) proposed a local suppression model, named *LK*-anonymity, for achieving anonymity in a trajectory database, which guaranteed that, in

a trajectory database, for any non-empty sub sequence with a length less than (or equal to) *L* must have a count greater than, or equal to, *K* in the database. Additionally, Gao et al. (2019) quantitatively measured the risk of privacy disclosure, in a license plate recognition (LPR) dataset, caused by re-identification attacks based on the concept of k-anonymity. A variety of factors were examined to determine the degree of anonymity of an individual, including temporal granularity and size of published data, local versus non-local vehicles, and continuous versus non-continuous observations. It was found that five spatiotemporal records were enough to uniquely identify about 90% of individuals. A suppression solution and a generalization solution were proposed to quantify the privacy-and-utility trade-off. He and Chow (2019) proposed a privacy control algorithm, based on information-theoretic k-anonymity for private operators, to safely share complex network-oriented data objects. The algorithm was proven to converge sub linearly toward a constrained maximum entropy under certain asymptotic conditions, with a measurable gap.

While these research efforts have been shown to be effective in some practical applications, indistinguishability models have required us to predefine or assume an attacker's background knowledge. However, it has become challenging (or even impossible) to enumerate an adversary's possible background knowledge before an attack occurs. As such, the indistinguishability privacy principle has been proven to be prone to privacy attacks, such as background knowledge attacks, and homogeneity attacks. To overcome these shortcomings, differential privacy was proposed as a strict definition of the uninformativeness privacy principle that makes no assumptions about the power or background knowledge of a potential adversary.

Based on differential privacy, Chen et al. (2012) first proposed PPDP algorithms for sequential data. A variable-length synthetic data, named n-grams, and based on the Markov assumption, was published, which described trajectories as transition probabilities based on a past history of (n-1) locations. Mir et al. (2013) introduced DP-WHERE, a differentially private synthetic trajectory generator that represented trajectory data as probability distributions, instead of directly modeling the sequential data at the level of an individual. He et al. (2015) presented DPT, a hierarchy reference system, to synthesize mobility data based on raw GPS trajectories of individuals. Xiao and Xiong (2015) proposed a definition of δ-location set to account for the temporal correlations in location data so that true locations could be hidden within a single trajectory. (Xiao and Xiong 2015) extended the definition of δ-location set and adopted a data release mechanism of an isotropic space, generated a "noisy" location, and transformed that noisy location back to the original space. Gursoy et al. (2018) proposed a probability-distribution-based approach, named DP-Star, which constructed a density-aware grid in order to preserve spatial densities. However, the works described above considered trajectories as sequences that did not include temporal information, although timestamp contained important information that was very useful for trajectory data analysis. Liu et al. presented VTDP, a fine-grained vehicle trajectory data sanitization framework which releases the attributes of IDs, positions, speeds, accelerations and timestamps under differential privacy guarantee (Liu et al. 2019). Due to the fine time granularity, the time duration of the continuous data that can be processed is usually short. Differential privacy of the mutual correlation of a trajectory pair can be found in (Lu et al. 2018).

Due to the high dimensionality of trajectory data that contain both timestamp and location information, it is challenging to achieve the goal of publishing actual trajectory data by directly adding noise with Laplacian or exponential mechanisms (McSherry and Talwar 2007). One way is to represent trajectory data as a tree. SeqPT (Chen et al. 2012) was the first tree-based trajectory PPDP algorithm to represent location sequence as a path from root node to leaf node, with a count of the sub-sequence frequent pattern stored in the tree node. Laplacian noise was added to the count to determine whether the subtree would keep growing on the corresponding node. In the process of growing the subtree, all locations had to be calculated. SeqPT obtained good utility for sequence data, but was limited in applying it to spatial-

temporal trajectory data. Due to the addition of timestamps, data dimensions grew exponentially and the amount of calculation also increased with the height of the tree. Our experiment shows that SeqPT is not suitable for high-dimensional spatial-temporal trajectory data and a large tree height (to be discussed later in this manuscript).

SafePath (Khalil et al. 2018) improved SeqPT by introducing a variable height and degree taxonomy tree. The purpose of such a design was to reduce the possibility of generating an empty node, and to speed up the process of filtering all possible timestamp and location combinations, when growing a subtree for each node. Although the algorithm efficiency improved, when compared with SeqPT, especially when dealing with real spatial-temporal trajectory data with high-dimensional attributes of timestamp and location, it is found that a taxonomy tree consumed part of the privacy budget and, at the same time, leads to a relatively low utility of sanitized trajectory data.

In summary, while many researchers have studied the problem of PPDP, research in the transportation area has been very limited. It remains a challenging task to balance algorithm efficiency and data utility under a strict privacy definition. Two models from the literature that are closest to our research, SeqPT and SafePath, are chosen as the benchmarks for testing the performance of our proposed algorithm.

3. Preliminaries

As a convenient reference, the mathematical notations used in this section are presented below.

$\mathcal{D}$: Trajectory dataset

$\widehat{\mathcal{D}}$: Sanitized trajectory dataset output by PPDP algorithm

$\mathcal{T}$: Timestamp domain

$\mathcal{L}$: Location domain

t: Timestamp, $t \in \mathcal{T}$

l: Location, $l \in \mathcal{L}$

$\mathcal{PT}$: Prefix tree

$Root$: Root of prefix tree $\mathcal{PT}$

E: Set of edges of prefix tree $\mathcal{PT}$, each edge represents a pair of timestamp and location

V: Set of nodes of prefix tree $\mathcal{PT}$, each node stores the count of a sub-trajectory

v_i: A tree node in set V

c_i: A count number on node v_i

e_{in}: A tree edge in set E and an in-edge of node v_i

e_{out}: A tree edge in set E and an out-edge of node v_i

$t_i l_i$: A trajectory point on edge e_{in}, $t_i \in \mathcal{T}$, $l_i \in \mathcal{L}$

$t_{i+1} l_{i+1}$: An adjacent trajectory point with $t_i l_i$ on edge e_{out}, $t_{i+1} \in \mathcal{T}$, $l_{i+1} \in \mathcal{L}$

h: Prefix tree height

θ: Threshold to determine if a noisy prefix tree node should be deleted or not

tr: Trajectory of a trip that include pairs of timestamp and location, represented by E in a prefix tree

ϵ: Privacy budget

δ: Parameter that relaxes differential privacy requirements

M: A differential privacy randomized mechanism

Ω: Every set of outputs of mechanism M

$\Pr[M(\mathcal{D}) \in \Omega]$: Probability of $M(\mathcal{D}) \in \Omega$

f: Any function

$\mathbb{R}^d$: A real value set, which is the output domain of function f

Δf: Sensitivity of function f

λ: Parameter of Laplacian distribution

$\tilde{l}$: The l-th level of the prefix tree

$\epsilon_{\tilde{l}}$: A function that represents the amount of privacy budget for the l-th level in prefix tree

σ: Parameter of privacy distribution function $\epsilon_{\tilde{l}}$

$\theta_{\tilde{l}}$: A function that represents the value of threshold for the l-th level in prefix tree

k, b: Parameters of function $\theta_{\tilde{l}}$

q: A query composed of several pairs of timestamp and location

$|q|$: Length of query q

$T(q)$: The set of user trajectories that contains q

$q(\mathcal{D})$: Count query function on dataset $\mathcal{D}$, it returns the count of query q on $\mathcal{D}$

$relative_{error}$: Relative error rate of count queries

s: A sanity bound used in calculating relative error

KL: A matrix representing travel time background knowledge between locations

kt_{ij}: Elements in matrix KL, represent the minimum arriving time between location l_i and l_j

$tval$: Time interval between in-edge and out-edge of a tree node

$\mathcal{L}_r$: A reduced location domain

$Pr_{\theta_{\tilde{l}}}$: The probability of an empty node having noisy count greater than or equal to $\theta_{\tilde{l}}$

$\mathbb{E}$: The expected value of number of empty nodes that were incorrectly selected

3.1. Trajectory data

This paper focuses on the PPDP problems of trajectory data collected by transit smart cards. When travelers board or alight from transit vehicles, their payment histories, which represent their travel trajectories are collected. Two basic features of such trajectory data are: 1) spatial-temporal, meaning such data contains both timestamp and location attributes, and 2) sequential, meaning the timestamp and location pairs are ordered. A definition of trajectory data is given below first, and some statistics of the data analyzed for this manuscript are described.

Definition 1 (Trajectory data) A trajectory tr is defined as a sequence of timestamp t_i in timestamp domain $\mathcal{T}$ and location l_i in location domain $\mathcal{L}$ that contains all travel records collected with an ID card.

$$tr = t_1 l_1 \rightarrow \cdots \rightarrow t_i l_i \rightarrow t_{i+1} l_{i+1} \rightarrow \cdots \rightarrow t_n l_n \quad (1)$$

Table 1 gives an example of a trajectory dataset which includes a total of eight trajectory data. Among them, the first trajectory data tr_1 travels from location Y at time slot 1 to location X at time slot 4. Note that t_i is strictly increasing in the sequence, so that we always have $t_i < t_{i+1}$. $|tr|$ denotes the trajectory

length which is the number of timestamp and location pairs in tr, for example we have $|tr_1| = 2$ in Table 1.

Table 1. Trajectory dataset

ID	Trajectory
tr_1	1Y→ 4X
tr_2	2X→ 3Z
tr_3	2X→ 3Z→ 4Y
tr_4	2Y→ 4X
tr_5	2Y→ 3Z
tr_6	3X→ 4Y
tr_7	1Z→ 2X→ 3Z
tr_8	1Z→ 4X

A trajectory $tr' = t'_1 l'_1 \rightarrow t'_2 l'_2 \rightarrow \cdots \rightarrow t'_{|tr'|} l'_{|tr'|}$ is a prefix of another trajectory $tr = t_1 l_1 \rightarrow t_2 l_2 \rightarrow \cdots \rightarrow t_{|tr|} l_{|tr|}$, denoted by $tr' \preceq tr$ if and only if $|tr'| \leq |tr|$ and $\forall 1 \leq i \leq |tr'|$, $t_i l_i = t'_i l'_i$. For example, in Table 1, tr_2 is a prefix of tr_3 but not tr_7.

We focus on the anonymized urban metro IC card data collected from Shenzhen, China in June 2016. The dataset includes six metro lines and a total of 137 stations, covering a majority of the geographic area of the city. The data attributes include a user ID, that has been anonymized and represents a unique passenger, the timestamp that the traveler checked into/out of a station, the location (the name of the station), fare paid, and other attributes that are less relevant for this research. Overall, the dataset contains 220 million metro IC card records, from 2.8 million passengers, over 29 days, which accounts for about 10% of the resident population in Shenzhen.

Figure 2 and Figure 3 show the geographic location and traffic flow distribution at each metro site included in our dataset. Figure 2 shows the metro network of Shenzhen, in which the lines (in six different colors) show different metro lines; the colored dots represent metro stations. Figure 3 shows traffic statistics for each station. Each column represents the number of check-in (purple) and check-out (yellow) trips per day at a subway station. We can see in Figure 3 that the trip distribution varied significantly among different stations. The top ten stations, with the most check-in and check-out trips, generate about 20% of the trip records, whereas some other stations have significantly lower traffic.

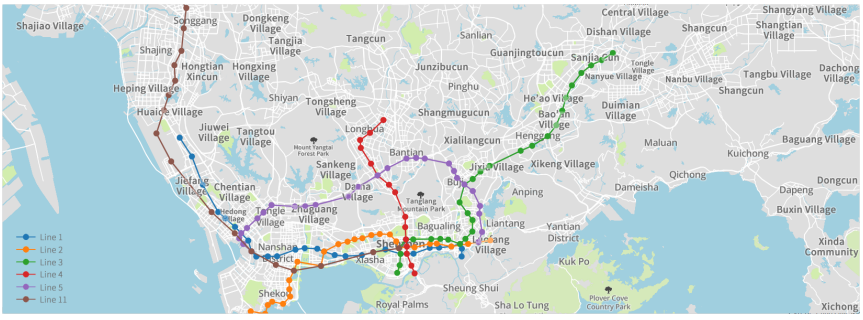

Figure 2. Metro network of Shenzhen.

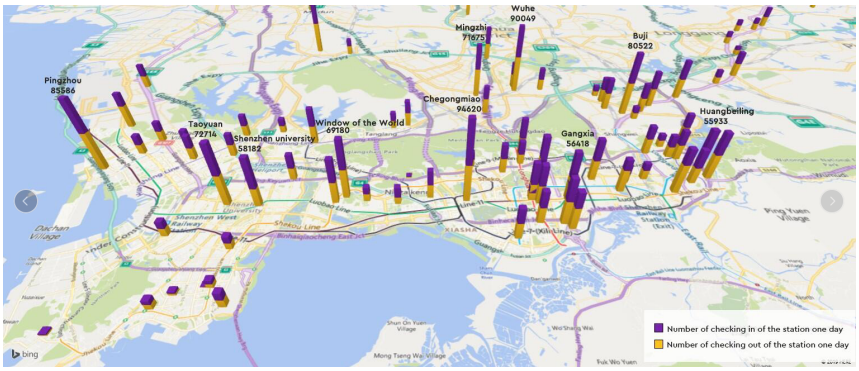

Figure 3. Traffic statistic for each station.

3.2. Prefix tree

A prefix tree is a kind of tree data structure that is often used to store a dictionary table or some sequence of characters. The trajectory data concerned in this manuscript is a kind of spatial-temporal sequence data, which makes a prefix tree a good match. A trajectory prefix tree is defined below.

Definition 2 (Trajectory Prefix Tree). A prefix tree PT of a trajectory dataset $\mathcal{D}$ is defined as a triplet $\mathcal{PT} = (Root, E, V)$, in which E is the set of edges with each edge representing timestamp and location $t_i l_i$ $pairs$ in a trajectory. Timestamp and location pairs, on a path from root to node, form a trajectory or a prefix of the trajectory. V is a set of nodes on which numbers represent a count of the trajectories on the path from root-to-node. The number on $Root \in V$ represents a count of the trajectories in dataset $\mathcal{D}$. An edge that connects a node and its parent node is called an **in-bound edge** (referred to as **in-edge**) of the node, whereas an edge that connects a node and its child node is called an **out-bound edge** (referred to as **out-edge**) of the node.

A trajectory prefix tree that corresponds to the dataset in Table 1 is given in Figure 4.

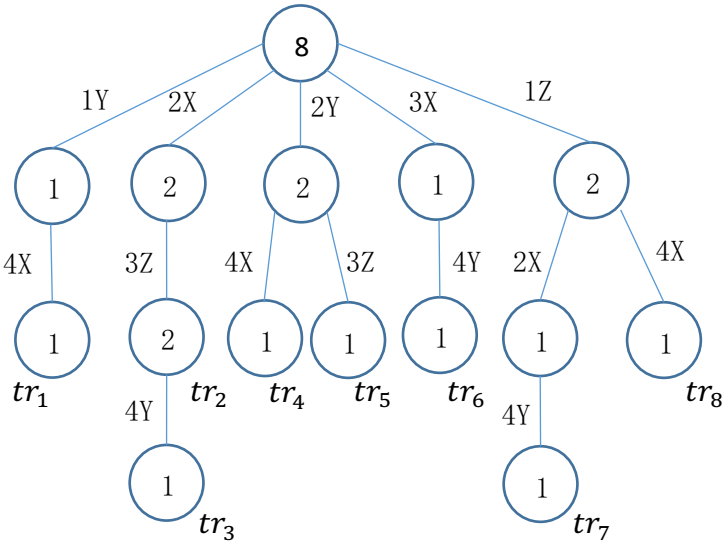

Figure 4. Prefix tree of the trajectories in Table 1.

3.3. Differential privacy

Differential privacy is a strong privacy definition. Assuming that two databases differ by only one record, the results of analyzing these two databases by using a differential privacy method will not show a significant difference. In other words, the results of an analysis would be independent from the presence of a particular record for a specific individual. Hence, it cannot be used in any way to violate an individual's privacy.

Differential private trajectory data publishing aims to output aggregated trajectories without disclosing any passenger's information. There are two scenarios of differential private data publishing, namely interactive and non-interactive. This manuscript focuses on non-interactive trajectory data publishing, in which all queries are submitted to a data owner at the same time, and the owner can provide answers with full knowledge of the query set. A non-interactive setting can generate more noise than an interactive setting, so it requires a more adaptive privacy budget allocation mechanism. Below is a formal differential privacy definition in a non-interactive setting:

Definition 3 (Differential Privacy). A randomized mechanism M gives ϵ-differential privacy if for any neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ differing by at most one record and for any possible sanitized dataset $\widehat{\mathcal{D}} \in Range(M)$,

$$\Pr[M(\mathcal{D}_1) = \widehat{\mathcal{D}}] \leq exp(\epsilon) \times \Pr\left[M(\mathcal{D}_2) = \widehat{\mathcal{D}}\right] \quad (2)$$

where the probability is taken over the randomness of M.(Chen et al. 2011)

The parameter ϵ refers to the privacy budget, which controls the level of privacy guarantee achieved by mechanism M. A smaller ϵ represents a stronger privacy level and can cause more noise to be added to the true answer. ϵ typically ranges $0 < \epsilon \leq 1$.

Differential privacy has two composition properties: sequential composition and parallel composition, which allow us to design more sophisticated algorithms. Sequential composition applies to situations where a sequence of differentially private computations takes place with the same set of data, and the entire sequence provides a privacy guarantee with the sum of all computations' privacy guarantees. Parallel composition is suitable for cases, where a sequence of differentially private computations takes place on a disjoint set of data, and the entire sequence gives the worst privacy guarantee among all of the computation's privacy guarantees.

3.4. Attacking model

Most attacks against trajectory data that are reported in the literature belong to the category of background knowledge attacks (also known as record linkage attacks) as defined by Fung et al. (2010). Background knowledge attacks aim at mapping records in a target trajectory dataset that is based on the background knowledge acquired by the attacker. The background knowledge can include personal travel habits or sensitive individual information, such as work location and home address. This information can be easily collected in many ways.

A successful attack enables the attacker to establish a link with records in a dataset, which leads to privacy leakages, when the records include sensitive information. Through these record linkages, one can analyze a traveler's work location, home address, activity pattern, and other sensitive information, based on the passenger's trajectories.

Our algorithm is based on a strict differential privacy definition. We define an attacking model by assuming that an attacker has the background knowledge of all records, except one on trajectory dataset $\mathcal{D} = \{tr_1, tr_2, \dots, tr_{|\mathcal{D}|}\}$, and that he (or she) cannot conclude the last trajectory from a differentially private

version of $\mathcal{D}$ (denoted by $\widehat{\mathcal{D}}$). As the random noise is added to the trajectory count, the sub-trajectories with counts, smaller than the threshold, will be deleted. An attacker cannot link background knowledge with one record, so the developed algorithm can effectively prevent record linkage attacks. $\widehat{\mathcal{D}}$ also guarantees its utility, in terms of count queries, which is widely used for various data analysis tasks.

To measure the utility of an output sanitized trajectory dataset $\widehat{\mathcal{D}}$, relative error is defined below through the accuracy of count queries $q(\mathcal{D})$. Count query, a common function on a trajectory dataset, returns the number of sub-trajectories in a trajectory dataset. It is also a basic operation of many data mining algorithms. For the quality of the sanitized trajectory data to be published, the accuracy of count queries is an important metric. A count query function is defined in the following. Query q is a sub-trajectory of trajectory tr, denoted by $q \subset tr$, if and only if $|q| \leq |tr|$ and $\forall t_i l_i \in q, t_i l_i \in tr$. $|q|$ is the length of query q, which represents the number of timestamp and location pairs in q. Count query q on a trajectory dataset D, denoted by function $q(\mathcal{D})$, returns the number of q in $\mathcal{D}$. Take the trajectory dataset in Table 1 as an example, and suppose $q = 2\mathrm{X} \rightarrow 3\mathrm{Z}$, $q(\mathcal{D}) = 3$, as tr_2, tr_3 and tr_7 all contain q.

Definition 4 (Relative Error) Relative error of count queries on synthetic dataset $\widehat{\mathcal{D}}$ is defined as follows:

$$relative_{error} = \left(\frac{\left| q\left(\widehat{\mathcal{D}}\right) - q(\mathcal{D}) \right|}{max\{q(\mathcal{D}), s\}} \right) \tag{3}$$

where s is a sanity bound and is suggested to take a value of 0.1% of the dataset size. (Khalil et al. 2018).

4. Methodology

In this section, we present an algorithm that publishes differential private trajectories. Section 4.1 gives an overview of our algorithm, while Section 4.2 describes each step of the algorithm. In the last part, Section 4.3, we present a theoretical analysis of privacy and the complexity of our algorithm.

4.1. Methodology overview

An algorithm is presented that can publish a differential private sanitized trajectory dataset. Compared with previous research, the developed algorithm features three improvements. First, as opposed to the commonly seen taxonomy tree structure at each level, a new prefix tree (without a taxonomy tree) utilizes privacy budgets better and improves the utility of output. The second is an incremental privacy budget allocation model which replaces the average privacy budget allocation scheme with a new model that distributes the privacy budget based on a distribution of count values at each prefix tree level. This module also helps with improving the utility of sanitized data under a fixed privacy budget. The last contribution is a spatial-temporal dimensionality reduction model, which narrows the timestamp and location combination domain and avoids dimension explosion without consuming any privacy budget. This module is intended to reduce the total run time.

The framework of our methodology (shown in Figure 5) mainly includes three modules. First, in the Initialization module, a raw dataset is scanned once to build an initial prefix tree following the new prefix tree structure. Then, in the HandleSubTree module, Laplacian noise is added to each tree node level of the developed incremental privacy budget allocation model and the spatial-temporal dimensionality reduction model. In the third part, the generated noise prefix tree is traversed and the sanitized dataset is generated as algorithm outputs.

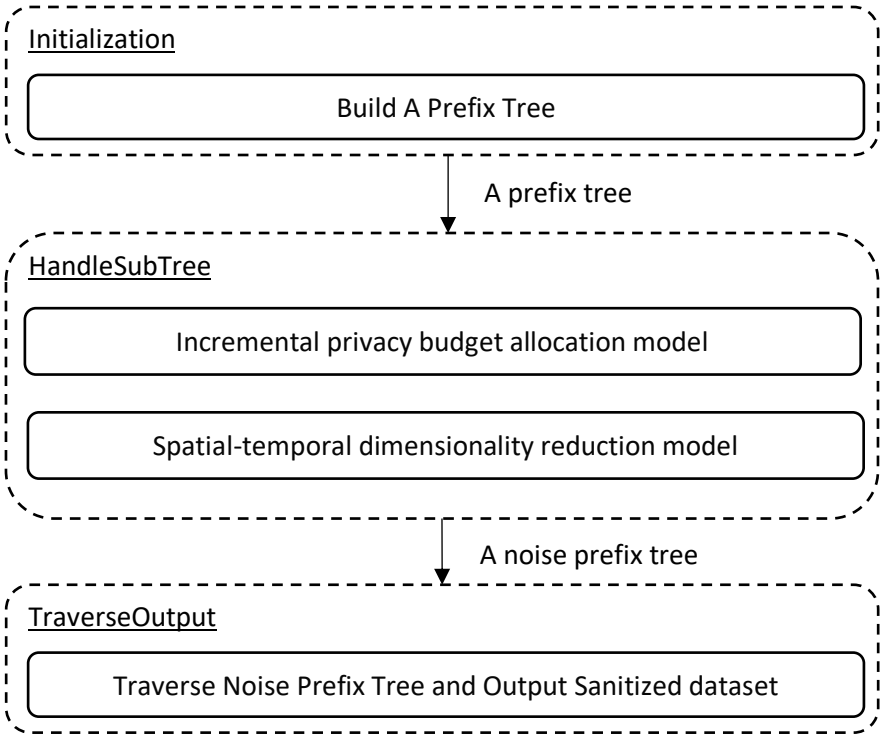

Figure 5. Algorithm framework.

A new prefix tree structure without taxonomy tree as sublevel**:** Due to the simultaneous presence of timestamp and location information in trajectory data, that data has high dimensionality and sparse characteristics. As such, a new prefix tree is used as a data structure to store the trajectory data and the sub-trajectory count. The developed model departed from existing research in that the commonly used taxonomy tree in each tree sub-level is not used, which helps with filtering empty nodes at the expense of consuming privacy budget. The rationale behind this is that, we believe that a taxonomy tree brings the problem of consuming the privacy budget and results in a reduction in the utility of sanitized trajectory data. Compared with previous research, that utilized a taxonomy tree to speed up the process of building the tree structure, our approach uses a new prefix tree, combined with a spatial-temporal background knowledge matrix, which does not consume extra privacy budget.

A prefix tree is described as $\mathcal{PT} = (Root, E, V)$, in which E represents a combination of pairs of timestamp and location and V represents a count of a sub-trajectory. Suppose there is a non-leaf and non-root node $v_i \in V$, an edge that connects v_i and its parent node is an **in-edge** $e_{in} \in E$ of v_i, whereas an edge that connects v_i and its child node is an **out-edge** $e_{out} \in E$. When growing new out-edge to build a new tree level, each level has a privacy budget of $\epsilon_{\tilde{l}}$ in our algorithm, as our model does not build any sub-level during the process of forming a new prefix tree level.

On the contrary, SeqPT and SafePath build both a timestamp taxonomy tree and a location taxonomy tree in the process of growing each new tree level. Suppose the height of the taxonomy tree is d, the privacy budget consumed on the taxonomy tree at each level is $\frac{(2\times d-1)}{2\times d} \times \epsilon_{\tilde{l}}$, and the privacy budget that is left to the real tree node is only $\epsilon_{\tilde{l}} - \frac{(2\times d-1)}{2\times d} \times \epsilon_{\tilde{l}} = \frac{\epsilon_{\tilde{l}}}{2\times d}$. Whereas, in the proposed model, we have the entire privacy budget ϵ_l allocable to all nodes at each level. So, although the structure of the taxonomy tree helps with filtering empty nodes, it consumes a majority of the privacy budget to build the taxonomy tree,

and the privacy budget left for the final tree node is only a very small proportion, leading to a large noise addition.

The new prefix tree is the basic data structure used to represent the trajectory dataset in our algorithm. An initial prefix tree is built first by inputting a raw trajectory dataset, and a noise prefix tree that satisfies the differential private definition is set up by adding noise to the counts on each tree node level, from top to bottom. Finally, the sanitized dataset is obtained by traversing a noise prefix tree once. The data structure of the new prefix tree is not only related to the problem of privacy budget allocation but, also, a combination node filtering. The previous problem effects output data utility, and the latter problem affects the efficiency of the algorithm.

Incremental privacy budget allocation model: A given privacy budget ϵ needed to be allocated level by level to construct a noise prefix tree to satisfy the differential privacy definition. As trajectories in each subtree are disjointed, satisfying a parallel composition feature of the differential privacy definition, nodes at the same level share the same privacy budget $\theta_{\tilde{l}}$. In existing research, such as SeqPT and SafePath, at each level, top down, the budget is evenly distributed $\theta_i = \theta_{i+1}$, $1 \leq i < h$. However, we argue that, as the statistical characteristics of each level are different, and the privacy budget determines the amount of noise added to each node, then the average budget allocation mechanism is not reasonable.

Suppose node $v_i \in V$ is the parent node of $v_{i+1} \in V$, c_i and $c_{i+1}$are count values on node v_i and v_{i+1}, in which $1 \leq i < h$. According to the characteristics of the count value on the prefix tree node, c_i is equal to the sum of count values on all of its child nodes. As such, it can be concluded that the count value on the node decreases by level, from top down, $c_{i+1} \leq c_i$. We propose an incremental privacy budget allocation model, based on such characteristics, in which both privacy budget $\epsilon_{\tilde{l}}$ and threshold $\theta_{\tilde{l}}$ of each level are the results of two functions of tree level $\tilde{l}$. σ, k, b are three adjustable parameters.

$$\epsilon_{\tilde{l}} = \frac{lg(\tilde{l}+\sigma)}{\sum_{l=1}^{h} lg(\tilde{l}+\sigma)} \times \epsilon, \sigma > 0 \quad (4)$$

$$\theta_{\tilde{l}} = k \times \tilde{l}^{-1} + b, k > 0, b > 0 \quad (5)$$

Privacy budget function $\epsilon_{\tilde{l}}$ is increasing by level, from the top down, $\epsilon_i < \epsilon_{i+1}$, $1 \leq i < h$. The reason is that, under the same privacy budget, the higher the count value is, the smaller the impact would be due to the added noise. On the other hand, the threshold function $\theta_{\tilde{l}}$ decreases by level, $\theta_i > \theta_{i+1}$, $1 \leq i < h$, as the average count value on nodes also decreases by level.

We implement the incremental privacy budget allocation model in the module of a HandleSubTree, as shown in Figure 5. The privacy budget $\epsilon_{\tilde{l}}$ and threshold $\theta_{\tilde{l}}$ are functions of tree level $\tilde{l}$, which are also determined by the total privacy budget ϵ and other parameters, such as k, b, and σ. The amount of noise added is $Lap\left(\frac{\Delta f}{\epsilon_{\tilde{l}}}\right) = Lap\left(\frac{1}{\epsilon_{\tilde{l}}}\right)$, in which $\Delta f = 1$ because the sensitivity of the counting function is 1, so the noise is inversely proportional to privacy budget $\epsilon_{\tilde{l}}$. Nodes, at the same level, share the same $\epsilon_{\tilde{l}}$ and $\theta_{\tilde{l}}$. All existing nodes are added noise to the raw count value c_i. If the noise value is greater than, or equal to $\theta_{\tilde{l}}$, that is $c_i + Lap\left(\frac{1}{\epsilon_{\tilde{l}}}\right) \geq \theta_{\tilde{l}}$, the node is retained but, otherwise, it is deleted. Nodes that do not previously exist are added noise to 0; nodes with results greater than, or equal to, $\theta_{\tilde{l}}$, that is $0 + Lap\left(\frac{1}{\epsilon_{\tilde{l}}}\right) \geq \theta_{\tilde{l}}$, are retained until a summation of the children nodes' noise counts exceed the value of the parent node, $\sum_{all\ child\ nodes} c_{i+1} \geq c_i$. Due to our incremental privacy budget allocation mechanism, the noise added to the original prefix tree has less impact on the raw dataset than the models under an even privacy budget allocation.

Spatial-temporal dimensionality reduction model: In order to satisfy ϵ-differential privacy definition, all possible timestamp and location pairs are enumerated when building a sublevel of each node in a prefix tree. Such enumeration brings significant, yet unnecessary, challenges to computational efficiency. Although some simple rules could be designed to narrow the search domain, such as the constraint of timestamps on tree edges (from parent to child) that strictly increase as the dimensions of the timestamp and location increase. The number of combinations becomes very high and inevitably slow the algorithm. In this section we propose a set of rules to reduce the spatial and temporal dimensions.

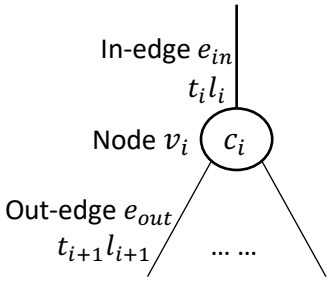

Figure 6. In- and out- edge of node.

Network geometric accessibility constraint. Suppose the in-edge of node v_i is $e_{in} \in E$, and $t_i l_i$ is the timestamp-location pair on edge e_{in}. $t_{i+1} l_{i+1}$ is the timestamp-location pair on edge e_{out}, as shown in Figure 6, $t_i l_i$ and $t_{i+1} l_{i+1}$ are adjacent points in a trajectory. In the previous work, all locations in $\mathcal{L}$ are added on the new out-edges, which means the entire location domain $\mathcal{L}$ is treated as a candidate location domain each time a node v_i needed to grow new out-edges e_{out}. However, we know that there must have been locations $l_{i+1} \in \mathcal{L}$ that are not reachable within the time interval $t_{i+1} - t_i$. An accessible and reduced location domain $\mathcal{L}_r \subseteq \mathcal{L}$ is proposed as the candidate location domain at each time new out-edges grew.

Minimum required travel time matrix KL. Matrix KL is defined to store the background knowledge of the minimal required travel time between two locations. The matrix dimension is $|\mathcal{L}| \times |\mathcal{L}|$, with $|\mathcal{L}|$ being the size of location domain. kt_{ij} in matrix KL represents the minimum arrival time from i-th location to j-th location in location domain $\mathcal{L}$.

$$\mathrm{KL}_{|\mathcal{L}|\times|\mathcal{L}|} = \begin{bmatrix} kt_{11} & \cdots & kt_{1|\mathcal{L}|} \\ \cdots & kt_{ij} & \cdots \\ kt_{|\mathcal{L}|1} & \cdots & kt_{|\mathcal{L}||\mathcal{L}|} \end{bmatrix} \tag{6}$$

As shown in Figure 6, prefix tree $\mathcal{PT}$ grows a subtree of node $v_i \in V$, after processing the out-edges that already exist in the raw trajectories, other out-edges need to be selected randomly from timestamp and location domain. 1) First, we randomly select a timestamp $t_{i+1} \in \mathcal{T}$ that satisfies $t_{i+1} > t_i$ as timestamp on a new out-edge e_{out}, then calculate a time difference $t_{i+1} - t_i$ between timestamps on edge e_{in} and e_{out}. 2) According to the difference, we traverse the row in matrix KL which represents the minimum required travel time from location l_i to all other locations in $\mathcal{L}$. 3) Only those locations with a minimum travel time $kt_{ij} \leq t_{i+1} - t_i$ are added as candidates to the location set $\mathcal{L}_r$.

As $\mathcal{L}_r \subseteq \mathcal{L}$, the candidate location domain has narrowed following the time constraint between the in- and out- edge during the growth of the prefix tree. Since matrix KL could be calculated, based on the distance between two locations and the free flow sped, the process of building a KL matrix does not consume any privacy budget, making it one of the main differences between our algorithm and that of previous works. As illustrated in Figure 5, the model is realized in a module of the HandleSubTree.

4.2. Algorithm steps

Algorithm 1, the main function, included three parts: Initialization, HandleSubTree, and TraverseOutput, as illustrated in Figure 5. The input data is trajectory dataset $\mathcal{D}$, and the output is sanitized trajectory dataset $\widehat{\mathcal{D}}$.

In Algorithm 1, raw trajectory dataset $\mathcal{D}$ is scanned once to build a trajectory prefix tree $\mathcal{PT}$, with a given height h (Algorithm 1, Line 1), and then noise is added to $\mathcal{PT}$, layer by layer, iteratively, to build a differential private prefix tree in a top-down fashion (Algorithm 1, Line 2-11). In the last step, a noisy prefix tree is traversed once and output a sanitized trajectory dataset $\widehat{\mathcal{D}}$ (Algorithm 1, Line 12-13).

In order to interpret the algorithm better, we run our algorithm on trajectories in Table 1. Suppose prefix tree height is set to $h = 3$, thresholds in level 1 to level 3 are set to $\theta_1 = 3$, $\theta_2 = 2$, $\theta_3 = 1$. Figure 7 shows a possible output prefix tree of our algorithm. As we can see, from the up down order in the tree, noise is added to the counts on each node. The nodes which have counts under the threshold are removed. Consistency is satisfied between all parent and child nodes, that is, the count on each parent node is the sum of all its child node counts.

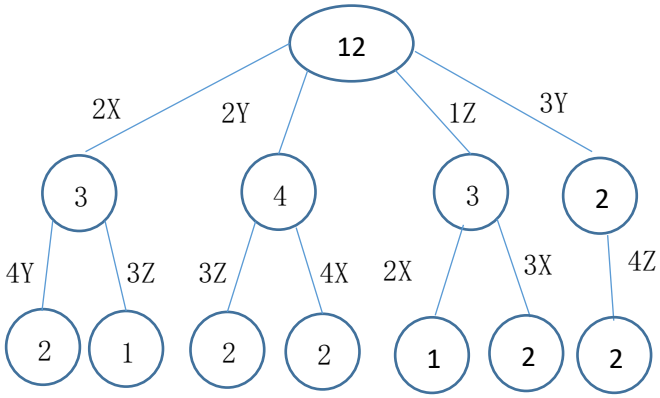

Figure 7. Noisy prefix tree of the trajectories in Table 1

Algorithm 1 MainFunc

Input: Raw trajectory dataset $\mathcal{D}$, Timestamp domain $\mathcal{T}$, Location domain $\mathcal{L}$

Input: Height of the prefix tree h

Input: Privacy budget ϵ

Input: Parameter σ, k, b

Output: Differentially-private trajectory dataset $\widehat{\mathcal{D}}$

1: Scan dataset $\mathcal{D}$ once to build a Prefix tree $\mathcal{PT}$ with height of h;
2: $i = 1$;
3: **while** $i \leq h$ do
4: $\epsilon_{\tilde{i}} = \frac{lg(i+\sigma)}{\sum_{i=1}^{h} lg(i+\sigma)} \times \epsilon$;
5: $\theta_{\tilde{i}} = k \times i^{-1} + b$;

6: **for** each node v_i in level i of $\mathcal{PT}$ **do**
7: add noise to the count value stored in node v_i;
8: HandleSubTree ($v_i, \epsilon_{\tilde{l}}, \theta_{\tilde{l}}$) ;
9: **end for**
10: i + + ;
11: **end while**
12: $\widehat{\mathcal{D}} \leftarrow$ TraverseOutput ($\mathcal{PT}$) ;
13: **return** $\widehat{\mathcal{D}}$;

One of the most important steps in Algorithm 1 is to grow a subtree of each parent node v_i by selecting out-edges of e_{out}. This is implemented in Procedure 1. When handling a sub-level, noise is added first to the count on each existing node, according to privacy budget $\epsilon_{\tilde{l}}$ (Procedure 1, Line 3). If the noise count on a node is greater than, or equal, to threshold $\theta_{\tilde{l}}$, the node is retained (Procedure 1, Line 4-7). After handling all existing nodes (Procedure 1, Line 2-11), if the summation of the noise on all existing child nodes is less than the noise count c_i on the parent node (Procedure 1, Line 12), then more timestamp and location pairs (that did not exist in the current edges) are randomly selected from a reasonable timestamp and location domain, according to the restricted location domain $\mathcal{L}_r$(Procedure 1, Line 12-27). The count value on the newly selected nodes equals 0 plus noise (Procedure 1, Line 16), and if the result is greater than, or equal to, $\theta_{\tilde{l}}$, the node is added to the child node set (Procedure 1, Line 17-19). The newly selected nodes, with an initial count of 0, are called "**empty node"**. If an empty node is selected, the noise count is added to the summation output sum (Procedure 1, Line 20-21). The summation sum is used to determine when to stop growing out-edges through the accumulation of counts on the child nodes. When the value of summation is greater than, or equal to, the count value c_i, the loop ended (Line 23-25).

Procedure 1 HandleSubTree

Input: Parent node v_i, noisy count c_i, location l_i, time t_i

Input: Privacy budget $\epsilon_{\tilde{l}}$, Threshold $\theta_{\tilde{l}}$

Output: Noisy child nodes set $\mathcal{N}$

1: $sum = 0$;
2: **for** each child node v_{i+1} of v_i **do**
3: $v_{i+1}.count = v_{i+1}.count + \mathrm{Lap}(1/\epsilon_{\tilde{l}})$;
4: **if** $v_{i+1}.count \geq \theta_l$ **then**
5: $\mathcal{N} \leftarrow v_{i+1}$;
6: $sum += v_{i+1}.count$;
7: **end if**
8: **if** $sum \geq c_i$ **then**
9: **break**;
10: **end if**
11: **end for**
12: **while** $sum < c_i$ **do**
13: Randomly select a time $t_{i+1} \in \mathcal{T}$ **and** $t_{i+1} > t_i$
14: $\mathcal{L}_r \leftarrow$ RestrictedLocDom ($l_i, t_{i+1} - t_i$);
15: **for** each location $l_{i+1} \in \mathcal{L}_r$ **do**
16: $count = \mathrm{Lap}(1/\epsilon_{\tilde{l}})$+0;
17: **if** $count \geq \theta_{\tilde{l}}$ **then**
18: add e_{out} as a new out-edge with $t_{i+1}l_{i+1}$;
19: add v_{i+1} as a new child node;

20: $v_{i+1}.count = count$;
21: $sum += count$;
22: **end if**
23: **if** $sum \geq c_i$ **then**
24: **break**;
25: **end if**
26: **end for**
27: **end while**
28: **return** $\mathcal{N}$;

A spatial-temporal dimensionality reduction model is implemented in Procedure 2. There are three input parameters: location knowledge matrix KL, location l_i, and a time interval $tval$. According to the timestamp and location pair $t_i l_i$ on the in-edge e_{in} of a parent node, a time $t_{i+1} > t_i$ is randomly selected from the time domain $\mathcal{T}$. Then, we calculate the time interval $tval = t_{i+1} - t_i$, traverse the row corresponding to location l_i in matrix KL (Procedure 2, Line 1).The locations that have a minimum arrival time kt_{ik}, that is less than, or equal to, the time interval $tval$ be added to the reachable location set $\mathcal{L}_r$ (Procedure 2, Line 2-4)). The location on the out-edge e_{out} is randomly selected from $\mathcal{L}_r$, which is a subset of location domain $\mathcal{L}$. It could be found that the set of $\mathcal{L}_r$ is determined by the parameters of location l_i and time interval $tval$. As different in-edges have different timestamp and location pairs, they have different $\mathcal{L}_r$.

Procedure 2 RestrictedLocDom

Input: Location knowledge matrix KL

Input: Location l_i, time interval $tval$ of parent node

Output: A reachable location domain $\mathcal{L}_r$

1: **for** each element kt_{ik} of the row in the matrix KL presents location l_i **do**
2: **if** $kt_{ik} \leq tval$ **then**
3: $\mathcal{L}_r \leftarrow l_k$
4: **end if**
5: **end for**
6: **Return** $\mathcal{L}_r$;

4.3. Theoretical analysis

Algorithm Improvements

We compare the proposed model with SeqPT model (Chen et al. 2012) and SafePath model (Khalil et al. 2018). Both of them allocate privacy budget evenly on each level of the prefix tree. Every node at the same level share the privacy budget, regardless of the number of nodes at each level, due to the characteristics of the prefix tree. When the tree level gets deeper, the trajectory count becomes smaller, so that the same privacy budget has a greater impact on deep nodes. For this reason, our algorithm is designed to assign adaptive, variable privacy budgets $\epsilon_{\tilde{l}}$ and threshold $\theta_{\tilde{l}}$ to different levels.

SafePath improves tree structure by introducing a taxonomy tree to build multi hierarchies. Some empty nodes are filtered out earlier, but the taxonomy tree consumes part of the privacy budget at the same time, and the privacy budget used to build the prefix tree is thus reduced. On the contrary, the proposed algorithm implements a spatial-temporal dimensionality reduction model instead of a taxonomy tree to help filter out empty nodes.

Number of descendant empty nodes

Next, we focus on the analysis of the probability that an empty node is selected. As discussed in previous research, this probability is used as the basis for theoretical analysis of the algorithm. We argue that, in the case of a fixed privacy budget, the choice of empty nodes is a must. Also, the number of empty nodes is only related to the privacy budget and, whether the probability is low or high, the number of empty nodes is relatively stable. A lower probability would neither increase the privacy level nor increase sanitized data utility, although it would increase algorithm runtime. Obviously, when the number of empty nodes needed is fixed, the lower the probability is, and the longer the entire process would take. Therefore, reducing the probability of selecting an empty node would not improve the level of differential privacy protection but, instead, will reduce the efficiency of the algorithm. Our algorithm increases the probability by reducing the threshold in order to speed up the process of picking empty nodes. This is one of the reasons why our algorithm is more efficient than those in previous works (Chen et al. , Khalil et al.) .

According to the following theoretical analysis, the probability is inversely proportional to $\epsilon_{\tilde{l}}$ and $\theta_{\tilde{l}}$. In our algorithm, the noise of the empty node is added to 0. Let $p(x) = \frac{1}{2\lambda} exp\left(\frac{-x}{\lambda}\right)$ be probability density function of the Laplace distribution. Given sensitivity $\Delta f = 1$ for a count query based on function f and privacy budget portion $\epsilon_{\tilde{l}}$, we have $\lambda = \frac{\Delta f}{\epsilon_{\tilde{l}}} = \frac{1}{\epsilon_{\tilde{l}}}$. Hence, $p(x) = \frac{\epsilon_{\tilde{l}}}{2} exp(-x\epsilon_{\tilde{l}})$. Given threshold $\theta_{\tilde{l}}$, the probability of an empty node having noise count $x + 0 \geq \theta_{\tilde{l}}$ is

$$Pr_{\theta_{\tilde{l}}} = Pr[x \geq \theta_{\tilde{l}}] = \int_{\theta_{\tilde{l}}}^{\infty} \frac{\epsilon_{\tilde{l}}}{2} exp(-x\epsilon_{\tilde{l}}) dx = \frac{1}{2} exp(-\epsilon_{\tilde{l}}\theta_{\tilde{l}}) \quad (7)$$

Since $-\epsilon_{\tilde{l}}\theta_{\tilde{l}} < 0$, we have $Pr_{\theta_{\tilde{l}}} < \frac{1}{2}$.

As $\epsilon_{\tilde{l}} = \frac{lg(\tilde{l}+\sigma)}{\sum_{\tilde{l}=1}^{h} lg(\tilde{l}+\sigma)} \times \epsilon < \epsilon$ and $\theta_{\tilde{l}} = k \times \tilde{l}^{-1} + b < b$ because $\tilde{l} \geq 1$, we have $Pr_{\theta_{\tilde{l}}} > \frac{1}{2} exp(-\epsilon \times b)$.

It can be concluded that $\frac{1}{2} exp(-\epsilon \times b) < Pr_{\theta_{\tilde{l}}} < \frac{1}{2}$.

For an empty node v_i at level i and a noise prefix tree of our algorithm with height h, the expected number of descendants of v_i that are all empty nodes can be calculated as below.

$$\mathbb{E} = \left(|\mathcal{L}_r||\mathcal{T}|Pr_{\theta_{\tilde{l}}}\right)^{h-i} < \left(\frac{1}{2}|\mathcal{L}_r||\mathcal{T}|\right)^{h-i} \quad (8)$$

Privacy guarantee

Algorithm 1 consists of three steps, namely Initialization, a HandleSubTree, and TraverseOutput. Given the total privacy budget ϵ, the first step merely converts the original trajectory dataset into the data structure of the trajectory prefix tree, so there is no privacy budget consumption in the first step.

In second step, the HandleSubTree builds a noise prefix tree by iteratively constructing one level at a time based on the output of the first step. Since all nodes on the same level contain a disjoint set of trajectories, according to the parallel composition theorem, the entire privacy budget consumed in a level is shared by all of the nodes on the same level. Each level is a dedicated privacy budget portion $\epsilon_{\tilde{l}} = \frac{lg(\tilde{l}+\sigma)}{\sum_{\tilde{l}=1}^{h} lg(\tilde{l}+\sigma)} \times \epsilon$, since the height of the noisy prefix tree is h, the HandleSubTree consumes the privacy budget in an amount that equals to $\sum_{\tilde{l}=1}^{h} \epsilon_{\tilde{l}} = \epsilon$.

The step, TraverseRelease, processes the noise prefix tree without accessing the underlying raw trajectories, so there is no privacy budget consumption in this step.

In summary, given ϵ as a user-input privacy budget, Algorithm 1 is ϵ-differentially private.

Complexity analysis

The run time of Algorithm 1 consists of three parts, which include those spent on building an initial prefix tree, adding noise and selecting an empty node to build a noise prefix tree, and traversing and releasing the sanitized trajectories. Assume that the input trajectory dataset is $\mathcal{D}$, that has $|\mathcal{D}|$ trajectories, the output sanitized dataset is $\widehat{\mathcal{D}}$, that has $|\widehat{\mathcal{D}}|$ trajectories. The height of the prefix tree is h, the size of time domain is $|\mathcal{T}|$, and the size of the location domain is $|\mathcal{L}|$.

The process of establishing a prefix tree is to scan each trajectory once, insert it into the prefix tree as a path from the root to a leaf or non-leaf node. The maximum path length is h. The time complexity of building a whole prefix tree is $O(h \cdot |\mathcal{D}|)$.

The second step of adding noise and selecting an empty node to build a noise prefix tree is the most time-consuming operation. For each selection, a timestamp t_{i+1}, that is greater than the previous location's timestamp t_i, is randomly selected from time domain $\mathcal{T}$. According to the selected timestampt_{i+1}, a set of locations $\mathcal{L}_r$ is filtered based on location back knowledge matrix KL. The time complexity is $O(|\mathcal{L}|)$. After adding Laplace noise, the timestamp and location pairs, whose noisy counts are greater than the threshold, are added to the prefix tree. Repeat the selection of $t_{i+1} \in \mathcal{T}$ and $t_{i+1} > t_i$ until the summation of the count values on all child nodes is greater than, or equal to, the noise count of the parent node. The time complexity is $|\mathcal{T}|$. The selection process is performed on all nodes, except the leaf nodes, and the number of executions has the same complexity with $|\mathcal{D}|$. Thus, the total complexity is $O(|\mathcal{D}| \cdot |\mathcal{T}| \cdot |\mathcal{L}|)$

In the last step, the computation cost of generating the private release by traversing the noisy prefix tree once is $O(h \cdot |\widehat{\mathcal{D}}|)$, which can be approximated as $O(h \cdot |\mathcal{D}|)$.

Since h is a very small constant, as compared to $|\mathcal{T}| \cdot |\mathcal{L}|$, the total complexity of Algorithm 1 is $O(|\mathcal{D}| \cdot |\mathcal{T}| \cdot |\mathcal{L}|)$.

5. Numerical experiment

This section describes a comprehensive analysis of the proposed algorithm. We evaluate the efficiency and scalability of the proposed algorithm, as well as the utility of the sanitized trajectory data used for counting queries. The real life datasets from the Shenzhen Metro smart card records, that are used, cover 2.8 million smart card users. Table 2 lists the datasets used in the experiment, in which $|\mathcal{D}|$ is the number of trajectories. Each trajectory corresponds to one user. $|\mathcal{T}|$ represents the size of time domain, we define two adjacent timestamps t, $t+1$ in $|\mathcal{T}|$. The time interval between t and $t+1$ is set to 15 minutes in our experiments. $|\mathcal{L}|$ represents the number of locations. $\max|tr|$ represents the maximum length of trajectories in the dataset, and $\text{avg}|tr|$ represents the average length of all trajectories.

By changing the size of the raw trajectory dataset, including the number of trajectories $|\mathcal{D}|$ and the size of the timestamp domain $|\mathcal{T}|$, we obtain four different trajectory datasets, as described in Table 2.

1) Dataset 1 has the smallest data size, with $|\mathcal{D}| = 393{,}552$, and the smallest $|\mathcal{T}| = 16$. Dataset 1 also has the smallest $\max|tr|$ and $\text{avg}|tr|$. We select Dataset 1 as our first experiment dataset for two reasons. The first is to test our model in a relatively small dataset, and the second is to compare it with other algorithms, including SeqPT which cannot run successfully on a larger dataset.

2) Datasets 2-4 are larger datasets from the same population. To comprehensively test our algorithm, Datasets 2-4 have $|\mathcal{D}|$ range from 772,606 to 845,727 and $|\mathcal{T}|$, which represent a significant increasement in the length of the trajectory, from 48 to 64 to 80. $\text{Max}|tr|$ ranges from 16 to 20.

3) Dataset 5 (Zheng et al. (2009) is a pedestrian trajectory dataset, which is used for verifying the developed algorithm under large location domain size $|\mathcal{L}|$. It contains GPS traces of 182 users over 5 years, majority of the data is from Beijing, China. We cut the original 14,650 trajectories into more than 734,210 trajectories.

Three performed analyses are described in this section. The first analysis focuses on the efficiency and accuracy of our algorithm with a different tree height h and privacy budget ϵ. We also verify the effectiveness of the selected threshold function by experiments. The experiment is performed on all four Metro smart card datasets (Dataset 1-4 in Table 2). The second analysis focuses on the scalability in handling datasets of different sizes and with different features and different parameters. The experiment is also conducted using all four different size datasets (listed in Table 2}. The third analysis focuses on a comparison of two similar algorithms, SeqPT and SafePath, in terms of efficiency and accuracy. Due to the limitations of SeqPT in handling a large-scale dataset, the comparison experiment with SeqPT is performed only on Dataset 1, the smallest of our datasets. The comparison experiment with SafePath is performed on all five datasets.

The proposed algorithm is implemented in Python. All experiments in this section are performed on a 64-bit personal computer with an Intel Core 2 Due 2.13 GHz CPU and 8GB RAM, running Windows 7. Table 2 lists the features of the five datasets that were used in our experiment.

Table 2: Experimental dataset statistics

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{T}\|$	$\|\mathcal{L}\|$	$\max\|tr\|$	$\text{avg}\|tr\|$
Dataset 1	393,552	16	121	6	1.84
Dataset 2	772,606	48	121	16	3.56
Dataset 3	824,957	64	121	18	3.55
Dataset 4	845,727	80	121	20	3.73
Dataset 5	734,210	300	1571	12	4.7

5.1. Utility analysis.

This section describes our examination of the utility of a sanitized dataset of our algorithm output. We follow the evaluation method from previous works by (Chen et al. 2012) and (Khalil et al. 2018) which measure the utility by generating 40,000 random count queries. We call the number of timestamp and location pairs in a count query as the query length $|q|$. For example, $q = 2\text{X}\rightarrow 3\text{Z}$ has a query length $|q| = 2$. Assuming $max|q|$ is the maximum query length $|q|$ among the 40,000 random count queries, we divid the query set into four subsets such that the query length of the i-th subset is uniformly distributed in $\left[1, \frac{i}{4} \times max|q|\right]$ and each timestamp and location pair draw values from the timestamp and location domains $|\mathcal{L}|$ and $|T|$, following an even distribution. Taking $max|q| = 8$ as an example, there are 40,000 count queries, including four subsets queries, with $|q| = 2, 4, 6, 8$, so each subset had 10,000 queries. The utility is measured by the average relative error of the count queries on the raw data set and the sanitized data set described in Eq. 3, in which the sanity bound s is set to be 0.1% of the dataset size. The privacy budget distribution at level $\tilde{l}$ could be calculated via $\epsilon_{\tilde{l}} = \frac{lg(\tilde{l}+\sigma)}{\sum_{\tilde{l}=1}^{h} lg(\tilde{l}+\sigma)} \times \epsilon$, and the threshold at level $\tilde{l}$ is $\theta_{\tilde{l}} = k \times \tilde{l}^{-1} + b$. In our experiments on utility analysis, we set $\sigma = 1.1$, $k = 1.5$, $b = 1$.

Figure 8 shows how the average relative errors vary under different h values with $\max|q| = \frac{h}{2}$. There are four subgraphs corresponding to Dataset 1-4. According to the range of $\max|tr|$ and $\text{avg}|tr|$ of the four data sets, the prefix tree height h, represented by X axis, is set to be between 3 and 14. Dataset 1 has a tree height of between 3 and 7, Dataset 2-3 is between 4 and 12, and Dataset 4 is between 4 and 14. The average relative error is shown by Y axis. Four lines in each subgraph represent $\epsilon = 0.5, 0.75, 1.0, 1.25$, respectly.

It can be observed that, in almost all cases, the relative error decreases as h increases. There are two possible explanations. The first is that the higher the tree is, the less trajectory information is lost. The other reason is that, as the tree height h increases, the query length $|q|$ also increases, resulting in a low hit rate for the query function and a smaller numerator of the error formula. Further, by comparing the four subgraphs, it could be observed that, as the dataset size increases, from Dataset1 to Dataset4, the relative error also increases. By observing the different curves in each subgraph, it could also be found that, as the privacy budget ϵ increases, the average relative error decreases slightly as well, which is consistent with our expectation.

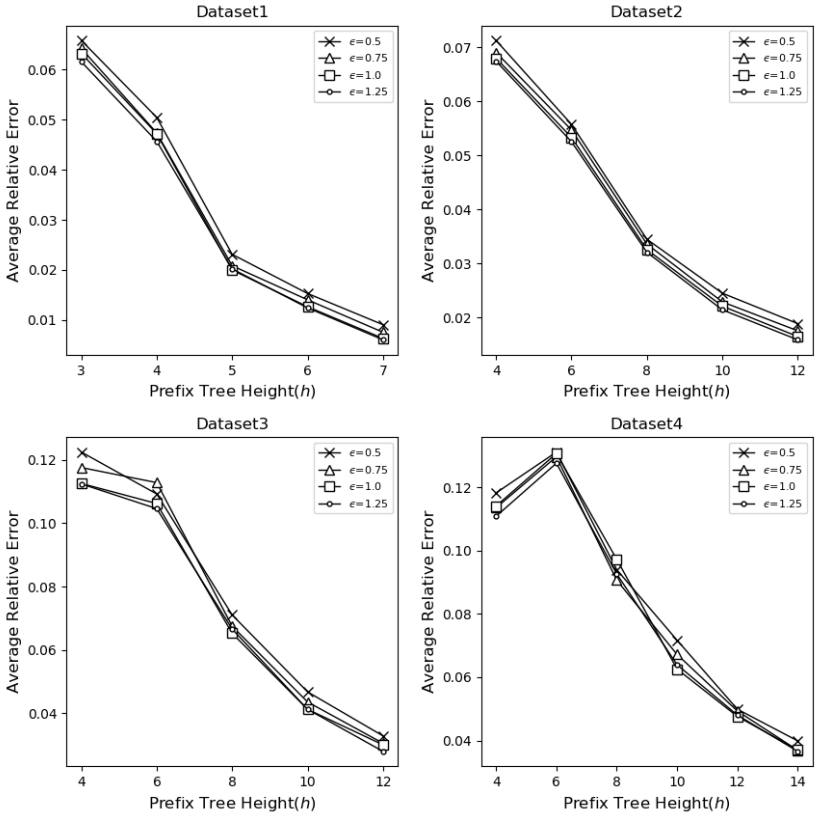

Figure 8. Average relative error vs. prefix tree height.

Figure 9 examines the average relative error under varying privacy budgets, which ranges from 0.5 to 1.5, as illustrated in X axis. The average relative error is shown in Y axis. The four lines represent four different $\max|q|$. The tree height h is set to 12 and $\max|q|$ is set to 2, 3, 4, 5 for Datasets 1; 2, 4, 6, 8; for Datasets 2 and 3, 6, 9,12; and for Dataset 3-4. It can be observed that the error rate decreases slowly when the privacy budget ϵ increases, which is consistent with the analysis above. On the other hand, it is found that $max|q|$ has a larger impact on the error rate than ϵ. When $max|q|$ is low, the relative error is relatively high. This is attributed to the fact that, when the maximum random queries length $max|q|$ becomes

longer, the hit rate in the trajectory datasets becomes lower, so there is a lower error rate under the longer $max|q|$. It is also found that, as the dataset size goes up, the average relative error also increases, which is consistent with our above findings.

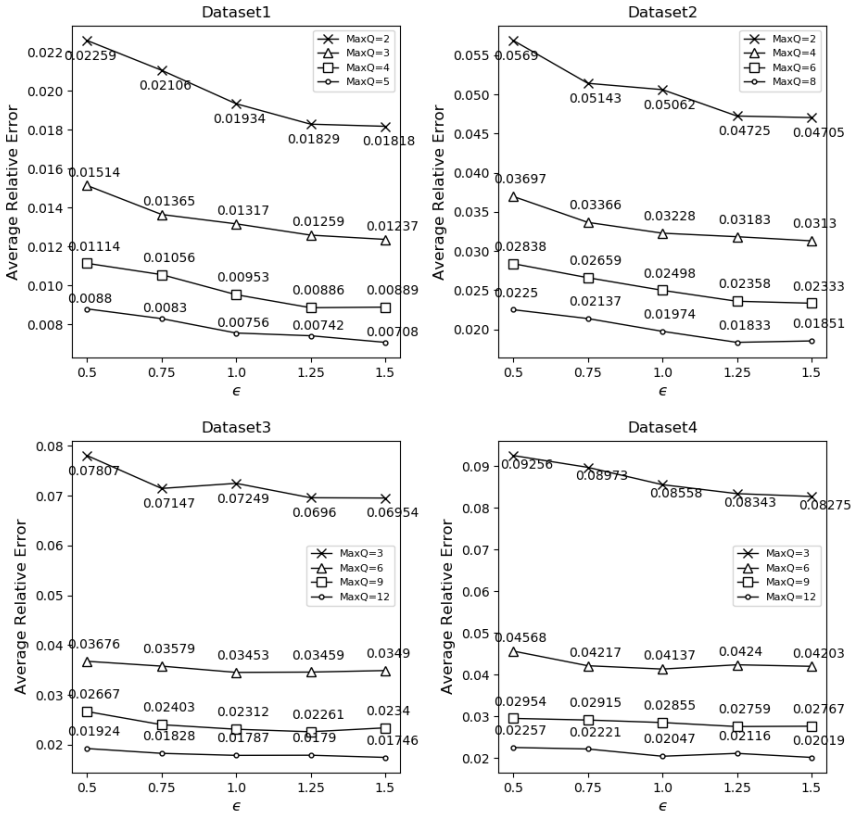

Figure 9. Average relative error vs. privacy budget.

By combining Figure 8 and 9, the proposed algorithm has an average relative error of less than 0.1, in most cases. This means that the sanitized trajectory dataset released by our algorithm has a very high utility. The worst case occurs in Datset 4 when $h = 6, \max|q| = 3$ in Figure 9, the error rate is below 0.14, and in Figure 9 $h = 12, \max|q| = 3$, the error rate is below 0.093. It is explainable because Dataset 4 is the largest dataset and has the highest dimension among the four metro smart card datasets.

We also verify the effectiveness of the selected threshold function by experiments. We compare our threshold allocation function $\theta_{\tilde{l}} = k \times \tilde{l}^{-1} + b$ with quadratic function $\theta_{\tilde{l}} = k \times \tilde{l}^{-2} + b$ and exponential function $\theta_{\tilde{l}} = k \times e^{-\tilde{l}} + b$. Figure 10 shows the average relative error under varying parameters k and b. It can be observed that the function used in our paper has the best accuracy in most cases. Four sub figures represent results on Dataset 1-4 respectively. The star line shows result of $\theta_{\tilde{l}} = k \times \tilde{l}^{-1} + b$, which is adopted by our algorithm, the other two lines represent results of $\theta_{\tilde{l}} = k \times \tilde{l}^{-2} + b$ and $\theta_{\tilde{l}} = k \times e^{-\tilde{l}} + b$. The prefix tree height $h = 12$, maximum query length $max|q| = 3$, privacy budget $\epsilon = 1$. In Dataset 1, 2 and 4, our function shows the best result in all cases. In Dataset 3, only when $k = 2.0$, $b = 1$, the error rate of exponential function is lower than ours.

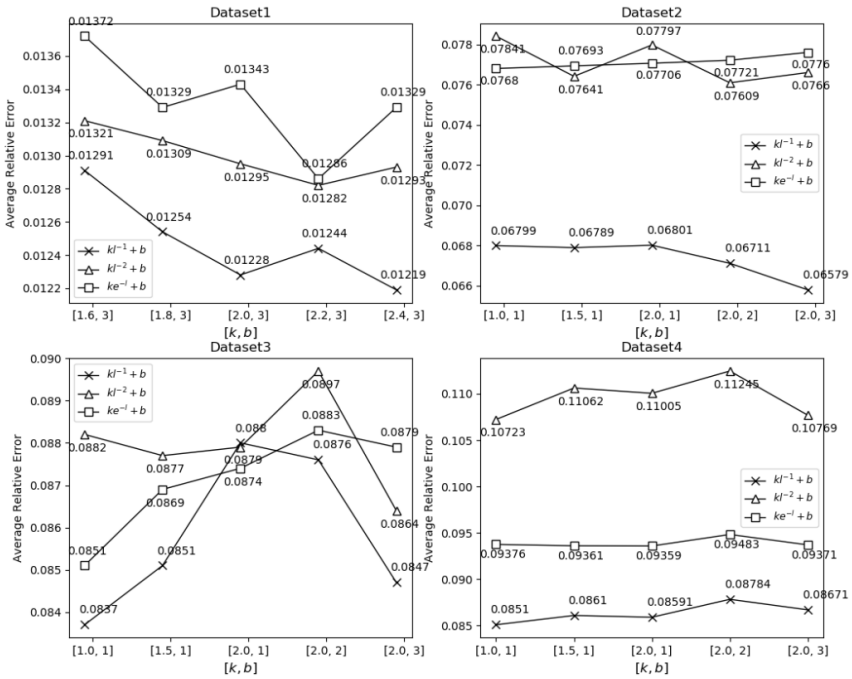

Figure 10. Average relative error under different threshold functions

5.2. Scalability analysis

We examine the scalability of our algorithm by varying the size of the raw trajectory dataset and the three parameters, k, b, and σ, that correspond to the four subgraphs in Figure 11. We set parameter h =14 and $\epsilon = 1$.

Figure 11(a) shows how runtime varied for four different datasets. X-axis represents four different datasets and there are four lines that represent time for reading, sanitization, writing, and total runtime. Reading time includes the time spent on reading a raw dataset and building the original prefix tree without noise added. Sanitization time refers to the time for building a noisy prefix tree. Writing time includes time for traversing the noisy tree and outputting sanitized trajectory data. The total run time is the summation of the three. It can be observed that reading, sanitization, and writing time all increase with the sizes of $|\mathcal{D}|$ and $|\mathcal{T}|$. Compared with the others, sanitization requires the longest run time and it also increases most significantly with the sizes of $|\mathcal{D}|$ and $|\mathcal{T}|$. Such an observation is consistent with our theoretical analysis in Section 4.3, as building a noisy prefix tree is the only process that has a complexity related with domain size.

The algorithm is found to be efficient and, even on the largest data Dataset4, the total run time is still less than 100 seconds. Figure 11(b-d) shows how parameters k, b, and σ affect run time, where X-axis represents different k, b and σ, respectively, while the other two parameters are fixed. It can be observed that, when k, b, or σ increase, downward trends are observed. In summary, the developed algorithm is relatively efficient and scalable to different dataset sizes and domain sizes with different parameters. Even in a large size dataset and domain, which is very common in real life, the proposed algorithm still enjoys a stable performance.

Combining the experiment results from Figure 10 and 11, k is recommended to be set between [1, 2], and b is recommended to be set between [1, 3].

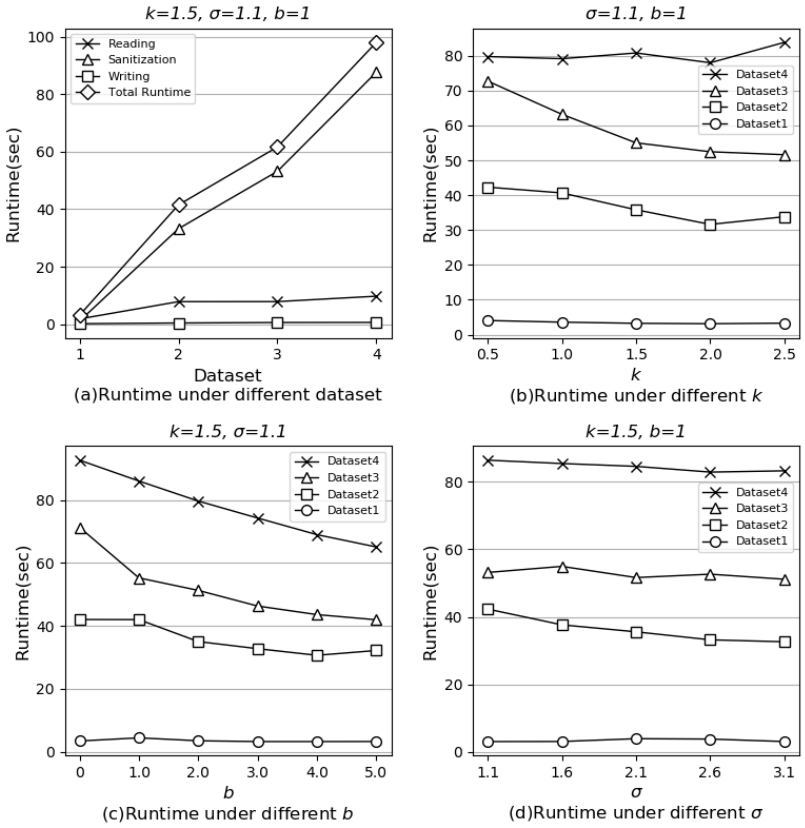

Figure 11. Scalability Analysis.

5.3. Comparisons with other models

In this section, the performance of the developed model is compared with SeqPT and SafePath, from the perspectives of average relative error and run-time efficiency.

SeqPT fails to finish and produces an out-of-memory error when prefix tree height $h = 3$ on Dataset 4, with $|\mathcal{D}| = 845{,}727, |\mathcal{T}| = 80, |\mathcal{L}| = 121, \max|tr| = 20, \text{avg}|tr| = 3.73$. When running on Dataset 1 with $|\mathcal{D}| = 393{,}552$, $|\mathcal{T}| = 16, |\mathcal{L}| = 121, \max|tr| = 6, \text{avg}|tr| = 1.84$, the run time reaches up to 20,000 seconds with $h = 8$. If h is set to be 9, the algorithm fails to finish. Due to the limitation of SeqPT with domain and dataset size, the comparison experiment with SeqPT is only performed on Dataset 1. The privacy budget is set to $\epsilon = 0.5$, and the tree height h is set between 2 and 5.

In terms of run-time efficiency, the comparison results are visualized in Figure 12, with four subgraphs showing the run-time comparison under Dataset 1-4. X-axis represents different h, Y-axis represents runtime. The proposed algorithm outperforms the other two algorithms at a prefix tree height from 2 to 5 under Dataset 1. When $h = 2$, our run-time is 1/3 of SeqPT and 1/4 of SafePath. As to $h = 5$, our run-time is 2/3 of SafePath and 1/30 of SeqPT. Although SeqPT performs pretty good when $h = 2$, it takes nearly 2 hours to run when $h = 5$. As the number of trajectories increases to $845{,}727$ with a high domain size of $|\mathcal{T}|$ and $|\mathcal{L}|$, it fails to finish and produces an out-of-memory error even if $h = 3$. So under Dataset 2-4, we only make our comparison with SafePath. In most cases, our algorithm has better performance, and the run-time efficiency advantage of our algorithm is more obvious, especially under the large dataset size of $|\mathcal{D}|$ and the high domain size of $|\mathcal{T}|$.

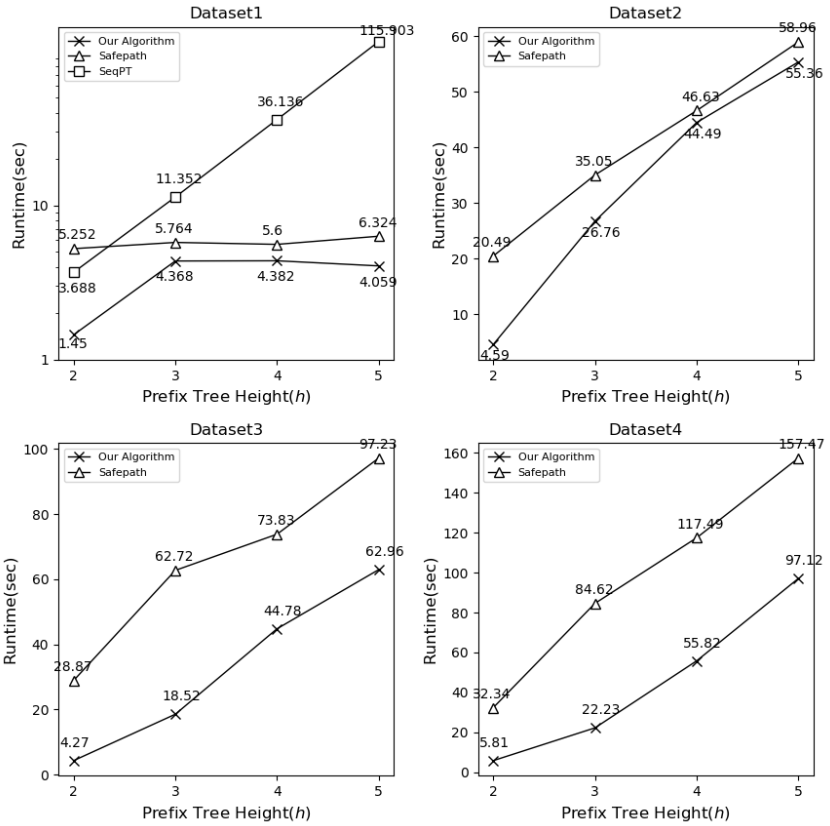

Figure 12. Run-time comparison under different tree heights.

Figure 13 shows the results of utility comparison, with four subgraphs representing results under four different datasets, $\max|q| = 2$ in this experiment. X-axis represents different h, Y-axis represents average relative error. The proposed algorithm outperforms the other two algorithms at different prefix tree heights, from 2 to 5. This especially occurs when $h = 5$ under Dataset 1, and the error rate of our algorithm is 0.034 which is about 1/3 of SafePath and 1/200 of SeqPT. Our algorithm has a better performance with all metro smart card datasets, which include both a smaller dataset with a lower domain size and a larger dataset with a higher domain size.

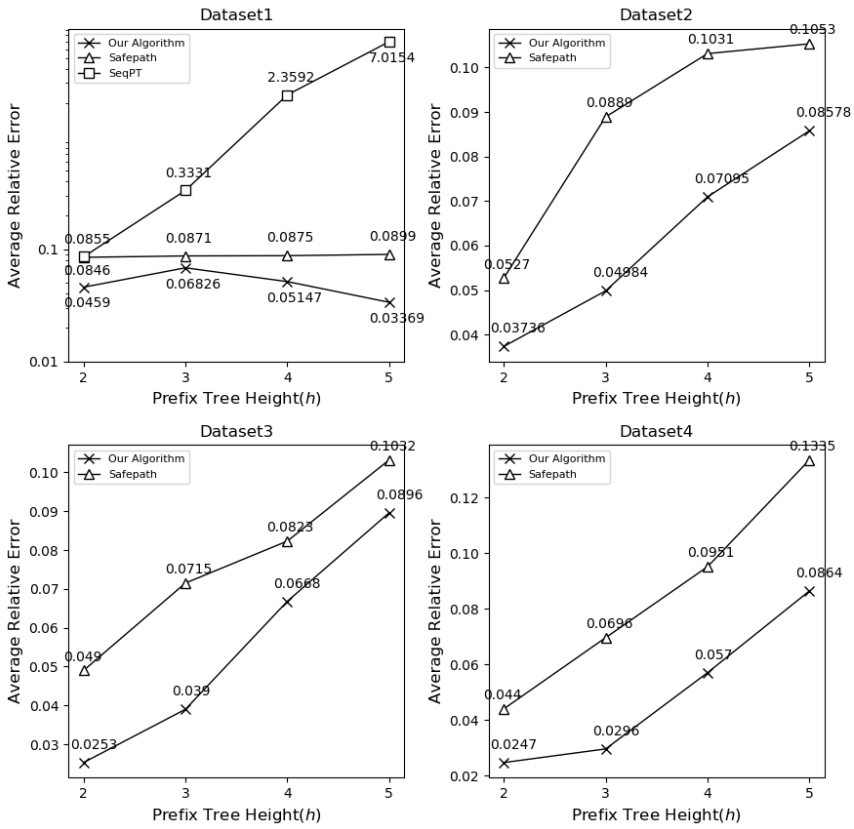

Figure 13. Average relative error comparison under different tree heights.

In order to verify the efficiency and accuracy of our algorithm on dataset with large location domain size $|\mathcal{L}|$, in the final experiment, we perform the comparison on Dataset 5. We compare the efficiency and accuracy of our algorithm with SeqPT and SafePath. Figure 14 shows how runtime and error rate vary under different values of prefix tree height. It can be found that with various prefix tree height from 2 to 5, the runtime of our algorithm increases from 113.9 seconds to 2007.3 seconds, which is lower than SafePath's 1076.9 seconds and 8976.3 seconds under $h = 2$ and $h = 3$. When h reaches to 4 and 5, SafePath fails to finish. SeqPT fails to finish under all prefix tree height values from $h = 2$ to $h = 5$. The sanitized data output by our algorithm also has a better utility compared with the other two algorithms.

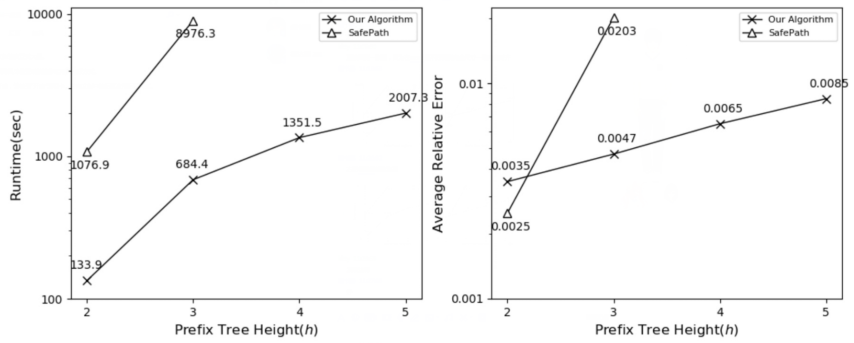

Figure 14. Algorithm comparison under large $|\mathcal{L}|$

In summary, through a comparison experiment between our algorithm, SeqPT and SafePath, with the same privacy budget ϵ and tree height h, our algorithm demonstrates better efficiency and sanitized data utility, in most cases, especially under conditions with a large dataset size and domain size.

6. Conclusion and Remarks

In this paper, we focus on the problem of a publishing sanitized trajectory dataset that satisfies the differential privacy definition. The trajectory data that we handle is spatial-temporal data with features that are large scale, high-dimensional, and sparse, which brings challenges to improving algorithm efficiency and data utility. A new prefix tree structure without taxonomy tree as sublevel, based on an algorithm with a dimensionality reduction model is proposed to improve the run-time efficiency, and an incremental privacy budget allocation model is developed to improve data utility. Through theoretical analysis and comparisons with previous works based on real-life trajectory datasets, the proposed algorithm demonstrates more efficient and scalable results. The sanitized trajectory dataset is also shown to have better utility. In addition to the transit smart card data, our method has the potential of being directly applied to other types of trajectory data, such as those from social media, navigation apps, ridesharing, and so on.

Future work could focus on the determination of optimal parameters, including tree height and parameters for the threshold and privacy budget function. Data structures, other than a prefix tree, may also be explored to further improve the performance of the proposed algorithm.

7. Acknowledgements

Research is supported by the National Natural Science Foundation of China (Grant No. 61876043, 61472089); NSFC-Guangdong Joint Found (Grant No.U1501254); Guangdong Provincial Key Laboratory of Cyber-Physical System (2016B030301008).

8. References

Abul, O., F. Bonchi and M. Nanni (2008). Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases. ICDE.

Chen, R., G. Acs and C. Castelluccia (2012). Differentially private sequential data publication via variable-length n-grams. Proceedings of the 2012 ACM conference on Computer and communications security, ACM.

Chen, R., B. Fung, B. C. Desai and N. M. Sossou (2012). Differentially private transit data publication: a case study on the montreal transportation system. Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.

Chen, R., B. C. Fung, N. Mohammed, B. C. Desai and K. Wang (2013). "Privacy-preserving trajectory data publishing by local suppression." Information Sciences 231: 83-97.

Chen, R., N. Mohammed, B. Fung, B. Desai and L. Xiong (2011). "Publishing SetValued Data via Differential Privacy." PVLDB 4: 1087-1098.

Cicek, A. E., M. E. Nergiz and Y. Saygin (2014). "Ensuring location diversity in privacy-preserving spatio-temporal data publishing." The VLDB Journal—The International Journal on Very Large Data Bases 23(4): 609-625.

Fung, B., M. Cao, B. C. Desai and H. Xu (2009). Privacy protection for RFID data. Proceedings of the 2009 ACM symposium on Applied Computing, ACM.

Fung, B. C. M., W. Ke, C. Rui and P. S. Yu (2010). "Privacy-preserving data publishing: A survey of recent developments." Acm Computing Surveys 42(4): 1-53.

Gao, J., L. Sun and M. Cai (2019). "Quantifying privacy vulnerability of individual mobility traces: A case study of license plate recognition data." Transportation Research Part C: Emerging Technologies 104: 78-94.
Ghasemzadeh, M., B. C. Fung, R. Chen and A. Awasthi (2014). "Anonymizing trajectory data for passenger flow analysis." Transportation research part C: emerging technologies 39: 63-79.
Gursoy, M. E., L. Liu, S. Truex and L. Yu (2018). "Differentially private and utility preserving publication of trajectory data." IEEE Transactions on Mobile Computing: 1-1.
He, B. Y. and J. Y. J. Chow (2019). "Optimal privacy control for transport network data sharing." Transportation Research Part C: Emerging Technologies.
He, X., G. Cormode, A. Machanavajjhala, C. M. Procopiuc and D. Srivastava (2015). "DPT: differentially private trajectory synthesis using hierarchical reference systems." Proceedings of the VLDB Endowment 8(11): 1154-1165.
Hoh, B., M. Gruteser, R. Herring, J. Ban, D. Work, J. C. Herrera, A. M. Bayen, M. Annavaram and Q. J. M. Jacobson (2008). "Virtual trip lines for distributed privacy-preserving traffic monitoring." MobiSys.
Hoh, B., T. Iwuchukwu and Q. Jacobson (2012). "Enhancing Privacy and Accuracy in Probe Vehicle-Based Traffic Monitoring via Virtual Trip Lines %J IEEE Transactions on Mobile Computing." IEEE Transactions on Mobile Computing 11(5): 849-864.
Hu, H., J. Xu, S. T. On, J. Du and J. K.-Y. Ng (2010). "Privacy-aware location data publishing." ACM Transactions on Database Systems (TODS) 35(3): 18.
Khalil, A. H., B. C. M. Fung, I. Farkhund, G. G. Dagher and E. G. Park (2018). "SafePath: Differentially-Private Publishing of Passenger Trajectories in Transportation Systems." Computer Networks 143: 126-139.
Liu, B., S. Xie, H. Wang, Y. Hong, X. Ban and M. Mohammady (2019). "VTDP: Privately Sanitizing Fine-grained Vehicle Trajectory Data with Boosted Utility." IEEE Transactions on Dependable and Secure Computing PP: 1-1.
Lu, O., Q. Zheng, L. Shaolin, H. Yuan, J. X. J. I. T. o. Dependable and S. Computing (2018). "Releasing Correlated Trajectories: Towards High Utility and Optimal Differential Privacy." IEEE Transactions on Dependable and Secure Computing: 1-1.
McSherry, F. and K. Talwar (2007). Mechanism Design via Differential Privacy. FOCS.
Mir, D. J., S. Isaacman, R. Caceres, M. Martonosi and R. N. Wright (2013). DP-WHERE: Differentially private modeling of human mobility. IEEE International Conference on Big Data.
Monreale, A., G. L. Andrienko, N. V. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo and S. Wrobel (2010). "Movement data anonymity through generalization." Trans. Data Privacy 3(2): 91-121.
Nergiz, M. E., M. Atzori and Y. Saygin (2008). Towards trajectory anonymization: a generalization-based approach. Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS. Irvine, California, ACM**:** 52-61.
Sun, Z., B. Zan, X. Ban and M. J. T. R. P. B. M. Gruteser (2013). "Privacy protection method for fine-grained urban traffic modeling using mobile sensors." Transportation Research Part B: Methodological 56: 50-69.
Sweeney, L. (2000). "Simple demographics often identify people uniquely." Health (San Francisco) 671: 1-34.
Terrovitis, M. and N. Mamoulis (2008). Privacy Preservation in the Publication of Trajectories. MDM.
Xiao, Y. and L. Xiong (2015). Protecting locations with differential privacy under temporal correlations. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM.
Yarovoy, R., F. Bonchi, L. V. Lakshmanan and W. H. Wang (2009). Anonymizing moving objects: How to hide a mob in a crowd? Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM.
Zheng, Y., L. Zhang, X. Xie and W. Y. Ma (2009). Mining Interesting Locations and Travel Sequences from GPS Trajectories. Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009.