Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Yonghui Xiao	Date

Protecting Locations of Individual Movement under Temporal Correlations

by

Yonghui Xiao
Doctor of Philosophy

Computer Science and Informatics

Li Xiong, Ph.D.
Advisor

Vaidy Sunderam, Ph.D.
Committee member

Michelangelo Grigni, Ph.D.
Committee member

Ashwin Machanavajjhala, Ph.D.
Committee member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Protecting Locations of Individual Movement under Temporal Correlations

by

Yonghui Xiao
B.S., Xi'an Jiaotong Unversity, 2005
M.S., Tsinghua University, 2011

Advisor: Li Xiong, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Abstract

Protecting Locations of Individual Movement under Temporal Correlations
By Yonghui Xiao

Concerns on location privacy frequently arise with the rapid development of GPS enabled devices and location-based applications. In this dissertation, we study how to protect the locations of individual movement under temporal correlations. First, we propose two types of privacy notions, location privacy and customizable privacy. Location privacy is used to protect the true location of a user at each timestamp; Customizable privacy means the user can configure personalized privacy notions depending on different demand. Second, we investigate how to preserve these privacy notions. We show that the traditional ℓ_1-norm sensitivity in differential privacy exaggerates the real sensitivity, and thus leads to too much noise in the released data. Hence we study the real sensitivity, called sensitivity hull, for the data release mechanism. Then we design the optimal location release mechanism for location privacy. We show that the data release mechanism has to be dynamically updated for the customizable privacy to guarantee the privacy is protectable, which is measured by a notion of degree of protection. Third, we implement these algorithms on real-world datasets to demonstrate the efficiency and effectiveness.

Protecting Locations of Individual Movement under Temporal Correlations

by

Yonghui Xiao
B.S., Xi'an Jiaotong Unversity, 2005
M.S., Tsinghua University, 2011

Advisor: Li Xiong, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Acknowledgments

First of all, I would like to thank my advisor Prof. Li Xiong. I have been working with Li for over six years since I was a master student in China. It was a pleasant and productive process during my work with Li, and it is also a treasure in my life which inspired me, encouraged me and changed me. Li is a special advisor who fully supports the crazy ideas of me. For example, in 2015, when I thought I can start a company based on the research, Li helped me got the NSF fund of 50 thousand dollars and participated the startup launch training in the NSF I-Corps program. I always feel that I am very lucky to have such a wonderful advisor.

I would also like to thank my thesis committee members, Prof. Vaidy Sunderam and Prof. Michelangelo Grigni, for their suggestions and comments on my dissertation. Thanks to Prof. Ashwin Machanavajjhala for coming from Duke University in North Carolina for my dissertation defense.

Thanks to Prof. James Nagy for teaching the useful course of Numerical Analysis, which is heavily used in my research. Thanks to Prof. Fusheng Wang for letting me take the qualifier exam. Thanks to Prof. Michele Benzi and Prof. Lars Ruthotto for their excellent courses.

Thanks to the members of our research group, Jinfei Liu, Xiaofeng Xu, Liyue Fan, Luca Bonomi, Yousef Elmehdwi, Yang Cao, Layla Pournajaf, Daniel Garcia ulloa, Farnaz Tahmasebian, Si Zhang, Michael Solomon, Slawomir Goryczka and James Gardner. Thanks to my friends, Zhengyao Wu, Huanhuan Yang, Yikai Wang, Yunyi Hu and Skanda Vivek, who are willing to discuss technical problems with me.

Finally, let me thank my family, and especially my girlfriend (and wife now) Xiao Fu, for their kind support and company.

To my family

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

With the technology advances in smartphones with localization capabilities, location based applications have been tremendously popular in people's lives. Location-based services (LBS) [13, 33] range from searching points of interest to location-based games and location-based commerce. Location-based social networks allow users to share locations with friends, to find friends, and to provide recommendations about points of interest based on their locations. Geospatial crowdsourcing [1, 34] allows individual users equipped with mobile devices collect and contribute location-dependent data about phenomena of common interest.

One major concern of location based applications is the location exposure [4]. In a survey [22] about location privacy, 78% smartphone users among 180 participants believe that apps accessing their location pose privacy threats. On the other hand, with the escalated fear of privacy exposure, there is not much users can do to protect themselves except turning off the location based

applications thoroughly. The reason is that to use these applications, users have to provide their locations to the respective service providers or other third parties. Moreover, since users' whereabouts contain sensitive information, e.g. the occupation, religion and even the relationship between users, it may cause serious troubles for users, like unwanted location based spams/scams, blackmails or even physical danger.

To tackle the privacy concerns and facilitate the location based applications, we investigate how to protect users' movement with state-of-the-art privacy notion, differential privacy (DP) [14]. DP was originally proposed to protect aggregated statistics of a dataset by bounding the knowledge gain of an adversary whether a user opts in or out of a dataset. Applying DP for location protection is still at an early stage. In particular, several works (e.g. [10, 20, 31, 53]) have applied DP on location or trajectory data but in a *data publishing* or *data aggregation* setting. In this setting, a trusted data publisher with access to a set of location snapshots or user trajectories publishes an *aggregate* or synthetic view of the original data while guaranteeing user-level DP, i.e. protecting the presence of a user's location or entire trajectory in the aggregated data. In contrast, in our setting, the protection needs to be enforced on the fly for *a single user*.

Next we describe several challenges on personal movement. First, a rigorous notion is needed for location privacy while accounting for temporal correlations. Second, users may have varying privacy preferences and utility requirements. Thus a flexible privacy notion is necessary for such scenario. In the following, we introduce two privacy notions on location in details, including location privacy and customizable privacy.

1.1.1 Location Privacy

We first consider the straightforward privacy, location privacy, which means to protect the exact location of a user at a single timestamp. An intuitive method, called location obfuscation, is to transform the exact location of a user to an area or a perturbed location (location perturbation) (e.g. [2, 23]). For example, in Figure 1.1, the user is at the "$\star$" position. Then a circle containing the "$\star$" can be used as the noisy location of the user. However, there are several shortcomings about this approach. We use the example in Figure 1.1 to explain this.

- Suppose a user moved from school to the cafeteria (where "$\star$" is) in Figure 1.1 (left). Three perturbed locations were released by selecting a point probabilistically in each of the three circles (by some spatial cloaking methods). Even though the individual locations were seemingly protected at each timestamp, considering them together with road constraints or the user's moving pattern will enable an adversary to accurately figure out the user is in the cafeteria, resulting in privacy breach.

- Suppose a user's location "$\star$" is protected in a circle as shown in Figure 1.1 (right). If by estimation based on previous locations the user can only be in the five places at current timestamp as shown in the figure, then the obfuscated location actually exposes the true location. Thus technically, the radius of the circle (in location obfuscation) should be subject to temporal correlations.

We can see that to protect the location privacy, temporal correlations have to be considered, in both the privacy notion and the location release mechanism. While such temporal correlations can be commonly modeled by Markov

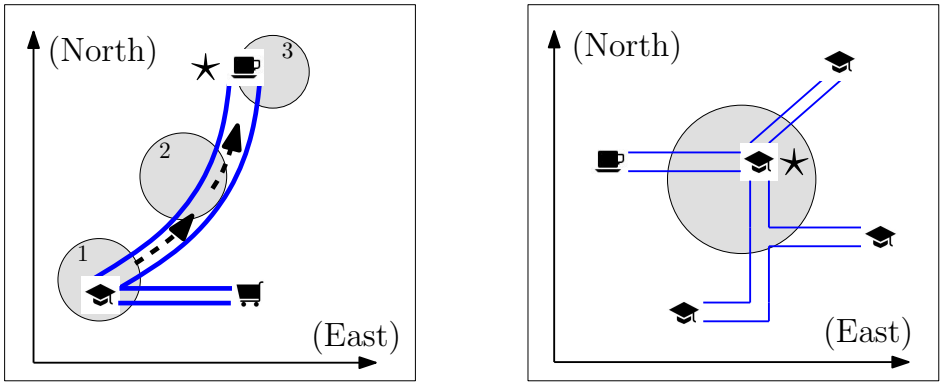


Figure 1.1: Examples of privacy breach caused by temporal correlations of user locations

chain [27, 43, 57], and few works have considered such Markov models [27, 57], it remains a challenge to provide rigorous privacy protection under temporal correlations for continual location sharing.

1.1.2 Customizable Privacy

Next we consider how to customize the location privacy for personal privacy demands by extending the Blowfish privacy [32]. Specifically, we treat every state in Markov model as a node, and construct a graph, in which edges represent "indistinguishability" between the connecting nodes, to represent the privacy policy. We use an example to explain the customization of privacy. Figure 4.3a shows the Markov model of a moving user with 6 states, denoted by $\{\mathbf{s}_1, \cdots, \mathbf{s}_6\}$. If the user prefers to hide her state in 3 categories, i.e., cafeteria, school and grocery (the octagon, circle and square in Figure 4.3a), the privacy customization can be achieved by the graph in Figure 4.3b. Then if the user is at state $\mathbf{s}_5$, the graph ensures that $\{\mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ are indistinguishable.

The customized privacy can improve both privacy and utility. First, by customization, users can achieve their own privacy requirements, which can

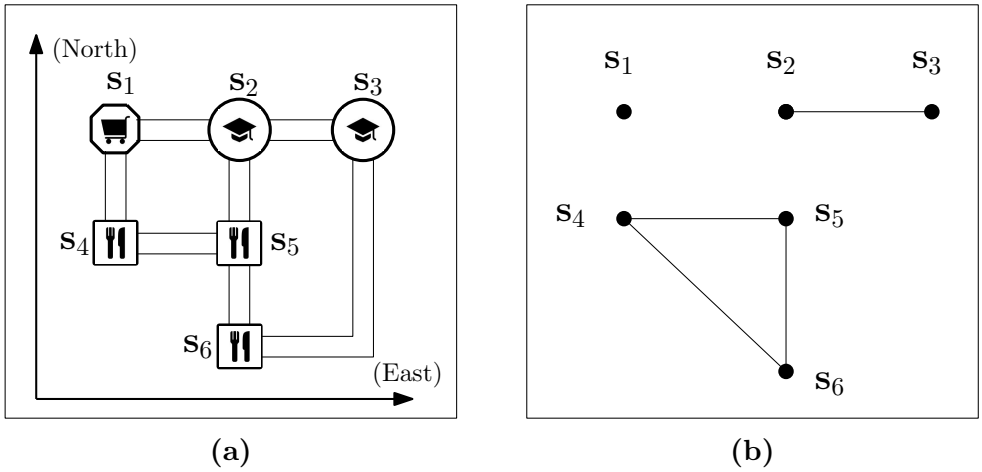


Figure 1.2: (a): protecting a state in its category (the octagon, circles or squares); (b): the policy graph connecting all nodes in a category.

vary dramatically. For example, a patient may want to prevent the exposure that she went to a hospital; while a doctor who works in the hospital may want to share that she is working at the hospital. Such customization can be achieve by our customizable privacy notion. Second, the utility of released location can also be boosted. In existing location release mechanisms, a location is usually perturbed as a circle. It is obvious that the circle can be either too small or too big for the user. For example, if a user only needs to hide her location in a category "restaurants", then other places in the circle are redundant for the preference. Through customization, we only hide the location in necessary candidates, which improves the utility.

1.2 Overview

we describe the setting of our method as follows. First, we define the privacy notions with differential privacy. Then we design algorithms to protect the privacy and release the perturbed locations. The utility of release location will be formally proven to be optimal under the privacy notions. At last, we

consider the impact of temporal correlations to quantify the overall protection.

Setting As shown in Figure 1.3, we consider a moving user with sensitive location stream who needs to share her locations to an untrusted location-based application host or other parties. A user's true locations are only known by the user. The "sanitized" locations released by the privacy mechanisms are observable to the service providers, as well as adversaries. To enable private location sharing, we address (and take advantage of) the temporal correlations, which can not be concealed from adversaries and hence are assumed to be public. Our goal is to develop the privacy mechanism to preserve the privacy.

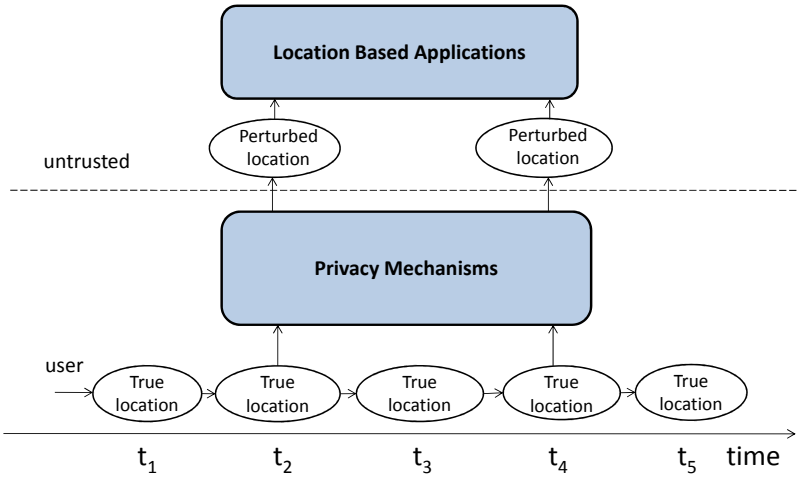


Figure 1.3: Problem setting

Location Release Mechanism To release a private location, a differentially private mechanism is used to perturbed the true location. Most existing works use Laplace mechanism, which injects independent Laplace noise to the true data. However, we prove that Laplace mechanism is just a special case of K-norm mechanism [29], and provides no better utility than K-norm mechanism.

Based on K-norm mechanism, we study the optimal mechanism to achieve the lower bound of differential privacy.

1.2.1 Contributions

Our contributions are summarized as follows.

Location Privacy. For location privacy at single timestamp, we first propose a "δ-location set" based differential privacy to protect the true location. In our problem, location changes between two consecutive timestamps are determined by temporal correlations modeled through a Markov chain [27,57]. Accordingly the "δ-location set" is intended to include all probable locations (where the user might appear). Intuitively, to protect the true location, we only need to "hide" it in the δ-location set in which any pairs of locations are not distinguishable. Second, we show that the well known ℓ_1-norm sensitivity in standard differential privacy fails to capture the geometric sensitivity in multidimensional space. Thus we propose a new notion, sensitivity hull, to capture the geometric meaning of sensitivity. We also prove that the lower bound of error is determined by the sensitivity hull. Third, we present an efficient location perturbation mechanism, called planar isotropic mechanism (PIM), to achieve δ-location set based differential privacy. To our knowledge, PIM is the first optimal mechanism that can achieve the lower bound of differential privacy.We also implement PIM on real-world datasets, showing that it preserves location utility for location based queries and significantly outperforms the baseline Laplace mechanism (LM).

Customizable Privacy. First, we propose a rigorous and customizable DPHMM notion by extending the Blowfish privacy [32]. Specifically, we treat every state in Markov model as a node, and construct a graph, in which edges represent

"indistinguishability" between the connecting nodes, to represent the privacy policy. In this way, the DPHMM notion guarantees that the true state is always protected in its connecting "neighbors". Second, we formally analyze the privacy risk under the constraint of temporal correlations. We show that the original graph may be reduced to a subgraph under the constraint, possibly with disconnected nodes. To detect the information leakage of the disconnected nodes, we define degree of protection (DoP) based on the graph to capture the protectability of a graph (if a graph is not protectable, then the disconnected nodes will be exposed). We also quantify the overall protection of Blowfish privacy in terms of differential privacy. Third, we develop a data release mechanism to achieve DPHMM. To tackle the detected information leakage, we study how to re-connect the disconnected nodes and find the optimal protectable graph based on the existing graph. We formulate the problem of building a minimum protectable graph with lowest error bound. Then we show that the problem is #P-hard and propose a fast greedy algorithm to solve it. We also implement and evaluate the data release mechanism on real-world datasets, showing that privacy and utility can be better tuned with customized policy graph. Finally, we thoroughly study the privacy guarantee of DPHMM framework. Besides comparing DPHMM with other privacy notions, we present the privacy composition results when multiple queries were answered over multiple timestamps. We also prove that the adversarial knowledge is bounded for adversaries with different prior knowledge.

1.3 Publications from this dissertation

The content of this dissertation is based on the following publications:

- Chapter 3 is based on the results presented in the paper **Protecting Locations with Differential Privacy under Temporal Correlations [62].** Yonghui Xiao, Li Xiong. CCS, 2015 and **LocLok: Location Cloaking with Differential Privacy via Hidden Markov Model**. Yonghui Xiao, Li Xiong, Si Zhang, Yang Cao. (submitted)

- Chapter 4 is based on the results presented in the paper **DPHMM: Customizable Data Release with Differential Privacy via Hidden Markov Model. [61]** (submitted).

Chapter 2

Related Works

2.1 Location Privacy

There is a rich set of literature related to location privacy. A few recent books and surveys [24,39] provide an up-to-date review of Location Privacy Preserving Mechanisms (LPPMs).

LPPMs can be roughly categorized into PIR-based methods and spatial transformation methods, as well as hybrid methods that combine the two approaches. PIR based methods (e.g. [25, 49, 52, 60]) guarantee cryptographic privacy by allowing data retrieval from a database without revealing any information to the database server about the retrieved item. However, such techniques tend to be expensive and impractical, especially for applications with large sets of points of interest and for dynamic geospatial applications that require continuous and dynamic location sharing. Spatial transformation methods generally use obfuscation methods, such as spatial cloaking, cell merging, location precision reduction or dummy cells, to achieve anonymity based privacy or uncertainty based privacy. However, anonymity or ad hoc

uncertainty based techniques do not always provide sufficient privacy protection [38, 57]. Most spatial transformation techniques proposed so far rely on syntactic privacy models such as k-anonymity, or ad-hoc uncertainty models, and do not provide rigorous privacy. Many of them only consider static scenarios or perturb the location at single timestamps without considering the temporal correlations of a moving user's locations, and hence are vulnerable to various inference attacks. The recent work [2] proposed a notion of geo-indistinguishability which extends differential privacy. However, a drawback of the privacy notion is that neighboring pairs are defined based on a radius and it does not consider the temporal correlations of multiple locations.

Several works use Markov models for modeling users' mobility and inferring user locations or trajectories [43, 54]. [27] proposed an insightful technique with a provable privacy guarantee to filter a user context stream even if the adversaries are powerful and have knowledge about the temporal correlations but it used suppression instead of perturbation. [57] investigated the question of how to formally quantify the privacy of existing LPPMs and assumed that an adversary can model users' mobility using a Markov chain learned from a population.

2.2 Inferences on Location

Various inference attacks can be carried out based on location information and external information such as moving patterns. These include: re-identification of a user, location disclosure of a user, probabilistic privacy breach of certain types of information such as presence in an area, trajectory during certain period, and so on. Several work showed (re)identification attacks based on

location information. [5] evaluated the risk of re-identifying a user even if the user identity is not explicitly released because the geo-localized history of user-requests can act as a quasi-identifier. [12] showed how to identify a user from the records of cellular network based on previous movements. [38] conducted an inference attack against obfuscation techniques and showed that a small fraction of users can be identified and home addresses of a big fraction of users were exposed.

Other works attempted to infer a user's actual location through modeling user's mobility and to define how to quantify location privacy (not just the identity). [57] investigated the question of how to formally quantify the privacy of existing LPPMs. An adversary can model users' mobility using a Markov process learned from a population. It then clarified three metrics related to location inference including accuracy, certainty and correctness, and argued the last one should be used as a privacy metric. [43] used a hierarchical Markov model to learn and infer a user's trajectory based on the places and temporal patterns they visited. [54] used the Continuous Time Bayesian Networks to predict uncertain trajectories of moving objects. [63] proposed a Sub-Trajectory Synthesis algorithm to learn the transition matrix of Markov chain and to predict a user's destination. Although these work were not in the context of privacy protection, the inspiration to us is that we should also assume adversaries have the ability to learn user's moving pattern and make the inference. Complicated inferences, like spatio-temporal queries, can also be derived with matrix computations. With the Markov model and uncertain locations of moving objects, [18] proposed a modified matrix computation method to efficiently compute the probability of a user appearing in certain region during certain time period.

In conclusion, location privacy has received a lot of attention. Although many LPPMs and related inference attacks have been proposed, they either did not protect a reliable privacy or proposed other adversary models which are not so trustworthy. To our knowledge, none of existing work has provided differential privacy in the presence of the strong adversary who can model a user's mobility through a Markov process. Because temporal correlations are the key characteristic of geospatial data, a privacy notion has to incorporate it for rigorous protection of individual movement.

2.3 Differential Privacy

While differential privacy [15] has been accepted as a standard notion for privacy protection, most works used Laplace mechanism [16] to release differentially private data. Based on Laplace mechanism, Li et al. proposed Matrix mechanism [40] to answer a batch of queries by factorizing a query matrix to generate a better "strategy" matrix that can replace the original query matrix. Other mechanisms, such as Exponential mechanism [47] and K-Norm mechanism [29], were also proposed to guarantee differential privacy. We refer readers to [30] for a comparative study of the mechanisms. A variety of differentially private applications [9, 19, 35] can also be found in literature. Several recent works have applied differential privacy to publish *aggregate* information from a large volume of location, trajectory or spatiotemporal data (e.g. [10,20,42,53]).

Because the concept of standard differential privacy is not generally applicable, several variants or generalizations of differential privacy, such as induced neighbors privacy [36], and δ-neighborhood privacy [21], have been proposed. Among these variants, Blowfish privacy [32] is the first generic framework with

customizable privacy policy. It defines sensitive information as secrets and known knowledge about the data as constraints. By constructing a policy graph, which should also be consistent with all constraints, Blowfish privacy can be formally defined.

Optimal query answering under differential privacy has been studied recently. Hardt and Talwar [29] studied the theoretical lower bound for any differentially private mechanisms and proposed K-norm mechanism. Bhaskara et al [6] studied another K-norm based method to project the sensitivity hull onto orthogonal subspaces. Nikolov et al [50] also improved the efficiency of K-norm mechanism by finding the minimal enclosing ellipsoid to release multivariate Gaussian noises. So far the best utility of existing mechanisms can be $log(d)$ approximately optimal.

Chapter 3

Differential Privacy on Location Set

In this chapter, we first define δ-location set based on temporal correlations of a user's moving pattern. Then we study how to guarantee the differential privacy on the δ-location set by proposing the sensitivity hull and the planar isotropic mechanism. Our algorithm is then evaluated in the experiment section.

3.1 Preliminaries

We denote scalar variables by normal letters, vectors by bold lowercase letters, and matrices by bold capital letters. We use $||\cdot||_p$ to denote the ℓ_p norm, $\mathbf{x}[i]$ to denote the ith element of $\mathbf{x}$, $\mathbb{E}()$ to denote the expectation, $\mathbf{x}^\intercal$ to denote the transpose of vector $\mathbf{x}$. Table 4.1 summarizes some important symbols for convenience.

$\mathbf{s}_i$	a cell in a partitioned map, $i = 1, 2, \cdots, m$
$\mathbf{u}$, $\mathbf{x}$	location in state and map coordinates
$\mathbf{u}^*$, $\mathbf{x}^*$	true location of the user
$\mathbf{z}$	the released location in map coordinate
$\mathbf{p}_t^-$	prior probability (vector) at timestamp t
$\mathbf{p}_t^+$	posterior probability (vector) at timestamp t
$\Delta\mathbf{X}$	δ-location set
K	sensitivity hull

Table 3.1: Notations in Chapter 3

3.1.1 Two Coordinate Systems

We use two coordinate systems, state coordinate and map coordinate, to represent a location for the Markov model and map model respectively. Denote $\mathcal{S}$ the domain of space. If we partition $\mathcal{S}$ into the finest granularity, denoted by "cell", then $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_m\}$ where each $\mathbf{s}_i$ is a unit vector with the ith element being 1 and other $m-1$ elements being 0. Each cell can represent a state (location) of a user. On the other hand, If we view the space as a map with longitude and latitude, then a 2×1 vector can be used to represent a user's location $\mathbf{x}$ with two components $\mathbf{x}[1]$ and $\mathbf{x}[2]$. Figure 3.1 shows an example using these two coordinate systems. If a user is in $\mathbf{s}_7$, the state coordinate and map coordinate are shown as follows. Note that the two coordinate systems can be transformed to each other. We skip how to transform them and treat $\mathbf{u}$ and $\mathbf{x}$ interchangeable.

$$\mathbf{u} = \mathbf{s}_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}$$

$$\mathbf{x} = [2, 4]^\intercal \; with \; \mathbf{x}[1] = 2 \; and \; \mathbf{x}[2] = 4$$

As time evolves, the trace of a user can be represented by a series of locations, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t$ in map coordinate or $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_t$ in state coordinate.

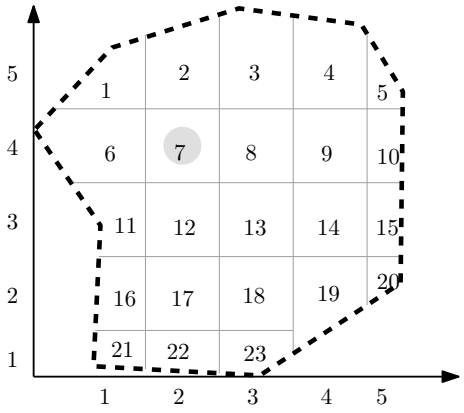


Figure 3.1: Two coordinate systems

3.1.2 Mobility and Inference Model

Our approach uses Markov chain [27,43,57] to model the temporal correlations between user's locations. Other constraints, such as road network, can also be captured by it. However, we note that Markov model, as well as any mobility models, may have limits in terms of predicability [58]. And we will discuss our solution to address these limits later.

In our problem setting, a user's true locations are unobservable, i.e. only known by the user. The "sanitized" locations released by the perturbation mechanism are observable to the service provider, as well as adversaries. Thus from an adversarial point of view, this process is a Hidden Markov Model (HMM).

At timestamp t, we use a vector $\mathbf{p}_t$ to denote the probability distribution of a user's location (in each cell). Formally,

$$\mathbf{p}_t[i] = Pr(\mathbf{u}_t^* = \mathbf{s}_i) = Pr(\mathbf{x}_t^* = \textit{the coordinate of } \mathbf{s}_i)$$

where $\mathbf{p}_t[i]$ is the ith element in $\mathbf{p}_t$ and $\mathbf{s}_i \in \mathcal{S}$. In the example of Figure 3.1, if the user is located in cells $\{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_7, \mathbf{s}_8\}$ with a uniform distribution, the

probability vector can be expressed as follows.

$$\mathbf{p} = \begin{bmatrix} 0 & 0.25 & 0.25 & 0 & 0 & 0 & 0.25 & 0.25 & 0 & \cdots & 0 \end{bmatrix}$$

Transition Probability. We use a matrix $\mathbf{M}$ to denote the probabilities that a user moves from one location to another. Let m_{ij} be the element in $\mathbf{M}$ at ith row and jth column. Then m_{ij} represents the probability that a user moves from cell i to cell j. Given probability vector $\mathbf{p}_{t-1}$, the probability at timestamp t becomes $\mathbf{p}_t = \mathbf{p}_{t-1}\mathbf{M}$. We assume the transition matrix $\mathbf{M}$ is given in our framework.

Emission Probability. If given a true location $\mathbf{u}_t^*$, a mechanism releases a perturbed location $\mathbf{z}_t$, then the probability $Pr(\mathbf{z}_t|\mathbf{u}_t^* = \mathbf{s}_i)$ is called "emission probability" in HMM. This probability is determined by the release mechanism and should be transparent to adversaries.

Inference and Evolution. At timestamp t, we use $\mathbf{p}_t^-$ and $\mathbf{p}_t^+$ to denote the prior and posterior probabilities of a user's location before and after observing the released $\mathbf{z}_t$ respectively. The prior probability can be derived by the posterior probability at previous timestamp $t-1$ and the Markov transition matrix as $\mathbf{p}_t^- = \mathbf{p}_{t-1}^+\mathbf{M}$. Given $\mathbf{z}_t$, the posterior probability can be computed using Bayesian inference as follows. For each cell $\mathbf{s}_i$:

$$\mathbf{p}_t^+[i] = Pr(\mathbf{u}_t^* = \mathbf{s}_i|\mathbf{z}_t) = \frac{Pr(\mathbf{z}_t|\mathbf{u}_t^* = \mathbf{s}_i)\mathbf{p}_t^-[i]}{\sum_j Pr(\mathbf{z}_t|\mathbf{u}_t^* = \mathbf{s}_j)\mathbf{p}_t^-[j]} \tag{3.1}$$

The inference at each timestamp can be efficiently computed by forward-backward algorithm in HMM, which will be incorporated in our framework.

3.1.3 Differential Privacy and Laplace Mechanism

Definition 3.1 (Differential Privacy)**.** *A randomized mechanism $\mathcal{A}()$ satisfies ϵ-differential privacy if for any output $\boldsymbol{z}$, $\frac{Pr(\mathcal{A}(\boldsymbol{x}_1)=\boldsymbol{z})}{Pr(\mathcal{A}(\boldsymbol{x}_2)=\boldsymbol{z})} \leqslant e^{\epsilon}$ where neighboring databases $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ satisfies*

- *(Unbounded DP) $\boldsymbol{x}_2$ can be obtained from $\boldsymbol{x}_1$ by adding or removing a tuple.*
- *(Bounded DP) $\boldsymbol{x}_2$ can be obtained from $\boldsymbol{x}_1$ by replacing a tuple.*

Laplace mechanism [16] is commonly used in the literature to achieve differential privacy. It is built on the ℓ_1-norm sensitivity, defined as follows.

Definition 3.2 (ℓ_1-norm Sensitivity)**.** *For any query $f(\boldsymbol{x})$: $\boldsymbol{x} \rightarrow \mathbb{R}^d$, ℓ_1-norm sensitivity is the maximum ℓ_1 norm of $f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)$ where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are any two instances in neighboring databases.*

$$S_f = \max_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \text{ neighboring databases}} ||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_1$$

where $||\cdot||_1$ denotes ℓ_1 norm.

A query can be answered by $f(\mathbf{x}) + Lap(S_f/\epsilon)$ to achieve ϵ-differential privacy, where $Lap() \in \mathbb{R}^d$ are i.i.d. random noises drawn from Laplace distribution.

3.1.4 Utility Metrics

To measure the utility of the perturbed locations, we follow the analysis of metrics in [57] and adopt the expected distance (called "correctness" in [57])

between the true location $\mathbf{x}^*$ and the released location $\mathbf{z}$ as our utility metric.

$$\textsc{Error} = \sqrt{\mathbb{E}||\mathbf{z} - \mathbf{x}^*||_2^2} \tag{3.2}$$

In addition, we also study the utility of released locations in the context of location based queries such as finding nearest k Points of Interest (POI). We will use precision and recall as our utility metrics in this context which we will explain later in the experiment section.

3.1.5 Convex Hull

Our proposed sensitivity hull is based on the well studied notion of convex hull in computational geometry. We briefly provide the definition here.

Definition 3.3 (Convex Hull)**.** *Given a set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, the convex hull of $\boldsymbol{X}$ is the smallest convex set that contains $\boldsymbol{X}$.*

Note that a convex hull in two-dimensional space is a polygon (also called "convex polygon" or "bounding polygon"). Because it is well-studied and implementations are also available [51], we skip the details and only use $Conv(\mathbf{X})$ to denote the function of finding the convex hull of $\mathbf{X}$.

3.2 Privacy Definition

To apply differential privacy in the new setting of continual location sharing, we conduct a rigorous privacy analysis and propose δ-location set based differential privacy in this section.

3.2.1 δ-Location Set

The nature of differential privacy is to "hide" a true database in "neighboring databases" when releasing a noisy answer from the database. In standard differential privacy, neighboring databases are obtained by either adding or removing a record (or a user) in a database. However, this is not applicable in our problem. Thus we propose a new notion, δ-location set, to hide the true location at every timestamp.

Motivations. We first discuss the intuitions that motivates our definition.

First, because the Markov model is assumed to be public, adversaries can make inference using previously released locations. Thus we, as data custodians in a privacy mechanism, can also track the temporal inference at every timestamp. At any timestamp, say t, a prior probability of the user's current location can be derived, denoted by $\mathbf{p}_t^-$ as follows.

$$\mathbf{p}_t^-[i] = Pr(\mathbf{u}_t^* = \mathbf{s}_i | \mathbf{z}_{t-1}, \cdots, \mathbf{z}_1)$$

Similar to hiding a database in its neighboring databases, we can hide the user's true location in possible locations (where $\mathbf{p}_t^-[i] > 0$). On the other hand, hiding the true location in any impossible locations (where $\mathbf{p}_t^-[i] = 0$) is a lost cause because the adversary already knows the user cannot be there.

Second, a potential shortcoming of Markov model is that the probability distribution may converge to a stationary distribution after a long time (e.g. an ergodic Markov chain). Intuitively, a user's possible locations can eventually cover the entire map given enough time. Hiding a location in a large area may yield a significantly perturbed location that is not useful at all.

According to [26], moving patterns of human have a "high degree" of tem-

poral and spatial regularity. Hence if people tend to go to a number of highly frequented locations, our privacy notion should also emphasize protecting the more probable locations in Markov model.

δ-Location Set. With above motivations, we define δ-location set at any timestamp t, denoted as $\Delta\mathbf{X}_t$. Essentially, δ-location set reflects a set of probable locations the user might appear (by leaving out the locations of small probabilities).

Definition 3.4 (δ-Location Set)**.** *Let $\boldsymbol{p}_t^-$ be the prior probability of a user's location at timestamp t. δ-location set is a set containing minimum number of locations that have prior probability sum no less than $1-\delta$.*

$$\Delta \boldsymbol{X}_t = min\{\boldsymbol{s}_i | \sum_{\boldsymbol{s}_i} \boldsymbol{p}_t^-[i] \geqslant 1-\delta\}$$

For example, if $\mathbf{p}_t^- = [0.3, 0.4, 0.05, 0.2, 0.03, 0.02]$ corresponding to $[\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6]$, then $\Delta\mathbf{X} = \{\mathbf{s}_2, \mathbf{s}_1, \mathbf{s}_4\}$ when $\delta = 0.1$; $\Delta\mathbf{X} = \{\mathbf{s}_2, \mathbf{s}_1, \mathbf{s}_4, \mathbf{s}_3\}$ when $\delta = 0.05$.

Note that if $\delta = 0$ the location set contains all possible locations. Thus 0-location set preserves the strongest privacy.

Drift. Because δ-location set represents the most probable locations, a drawback is that the true location may be filtered out with a small probability (technically, $Pr(\mathbf{x}^* \notin \Delta\mathbf{X}) = \delta$). Same situation may also occur if the Markov model is not accurate enough in practice due to its limit in predicability, as we mentioned earlier. Therefore, we denote this phenomenon as "drift" and handle it with the following surrogate approach.

Surrogate. When a drift happens, we use a surrogate location in $\Delta\mathbf{X}$ as if it

is the "true" location in the release mechanism.

Definition 3.5 (Surrogate)**.** *A surrogate $\tilde{\boldsymbol{x}}$ is the cell in $\Delta \boldsymbol{X}$ with the shortest distance to the true location $\boldsymbol{x}^*$.*

$$\tilde{\boldsymbol{x}} = \underset{\boldsymbol{s} \in \Delta \boldsymbol{X}}{argmin}\, dist(\boldsymbol{s}, \boldsymbol{x}^*)$$

where function $dist()$ denotes the distance between two cells.

Note that the surrogate approach does not leak any information of the true location, explained as follows. If $\mathbf{x}^* \in \Delta\mathbf{X}$, then $\mathbf{x}^*$ is protected in $\Delta\mathbf{X}$; if not, $\tilde{\mathbf{x}}$ is protected in $\Delta\mathbf{X}$. Using surrogate does not reveal whether $\mathbf{x}^*$ is in $\Delta\mathbf{X}$ or not. Because in any location release mechanisms $\mathbf{x}^*$ is treated as a black box (oblivious to adversaries), replacing $\mathbf{x}^*$ with $\tilde{\mathbf{x}}$ is also a black box. We formally prove the privacy guarantee in Theorem 3.6.

In some cases, a surrogate may be far from the true location. Then the released location may not be useful. Therefore, we also measure the distance between released location and true location in our experiment to reflect the long-term effect of surrogate.

3.2.2 Differential Privacy on δ-Location Set

We define differential privacy based on δ-location set, with the intuition that the released location $\mathbf{z}_t$ will not help an adversary to differentiate any instances in the δ-location set.

Definition 3.6 (Differential Privacy)**.** *At any timestamp t, a randomized mechanism $\mathcal{A}$ satisfies ϵ-differential privacy on δ-location set $\Delta \boldsymbol{X}_t$ if, for any output*

$\boldsymbol{z}_t$ and any two locations $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in $\Delta \boldsymbol{X}_t$, the following holds:

$$\frac{Pr(\mathcal{A}(\boldsymbol{x}_1) = \boldsymbol{z}_t)}{Pr(\mathcal{A}(\boldsymbol{x}_2) = \boldsymbol{z}_t)} \leqslant e^{\epsilon} \tag{3.3}$$

Above definition guarantees the true location is always protected in δ-location set at every timestamp. In another word, the released location $\mathbf{z}_t$ is differentially private at timestamp t for continual location sharing under temporal correlations. For other application settings, like protecting the trace or trajectory of a user, we defer the investigation to future works.

3.2.3 Adversarial Knowledge

In reality, there may be a variety of adversaries with all kinds of prior knowledge. Accordingly, we prove that for the problem of continual location sharing differential privacy is equivalent to adversarial privacy, first studied in [55].

Definition 3.7 (Adversarial Privacy)**.** *A mechanism is ϵ-adversarially private if for any location $\boldsymbol{s}_i \in \mathcal{S}$, any output $\boldsymbol{z}$ and any adversaries knowing the true location is in $\Delta \boldsymbol{X}$, the following holds:*

$$\frac{Pr(\boldsymbol{u}_t^* = \boldsymbol{s}_i | \boldsymbol{z}_t)}{Pr(\boldsymbol{u}_t^* = \boldsymbol{s}_i)} \leqslant e^{\epsilon} \tag{3.4}$$

where $Pr(\boldsymbol{u}_t^* = \boldsymbol{s}_i)$ and $Pr(\boldsymbol{u}_t^* = \boldsymbol{s}_i | \boldsymbol{z}_t)$ are the prior and posterior probabilities of any adversaries.

We can show Definition 3.6 is equivalent to adversarial privacy for continual location sharing, which can be derived from the PTLM property [55].

Lemma 3.1. *For the problem of continual location sharing, the following properties hold:*

I There exists only one true location at a timestamp:

$$Pr(\boldsymbol{u}^* = \boldsymbol{s} \cap \boldsymbol{u}^* = \boldsymbol{s}') = 0, \text{ for any locations } \boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$$

II Let $\boldsymbol{S} \subseteq \mathcal{S}$ be an area and $Pr(\boldsymbol{S})$ be the probability that the user is in $\boldsymbol{S}$. For any two areas $\boldsymbol{S}$ and $\boldsymbol{S}'$,

$$Pr(\boldsymbol{S})Pr(\boldsymbol{S}') \geqslant Pr(\boldsymbol{S} \cap \boldsymbol{S}')Pr(\boldsymbol{S} \cup \boldsymbol{S}')$$

Theorem 3.1. *For the problem of continual location sharing, Definition 3.6 is equivalent to Definition 3.7.*

Definition 3.7 limits the information gain for adversaries knowing the condition $\mathbf{x}_t^* \in \Delta\mathbf{X}$. If $\mathbf{x}_t^* \notin \Delta\mathbf{X}$, our framework reveals no extra information due to the surrogate approach. Thus adversarial knowledge can be bounded, discussed as follows.

Standard Adversary. For adversaries who have exactly the same Markov model and keep tracking all the released locations, their knowledge is also the same as our model (with location inference in Section 3.4.3). In this case, differential privacy and adversarial privacy are guaranteed, and we know exactly the adversarial knowledge, which in fact can be controlled by adjusting ϵ.

Weak Adversary. For adversaries who have little knowledge about the user, the released locations may help them obtain more information. With enough time to evolve, they may converge to standard adversaries eventually. But their adversarial knowledge will not exceed standard adversaries.

Strong Adversary. For adversaries who have additional information, the

released location from differential privacy may not be very helpful. Specifically, a strong adversary with auxiliary information may have more accurate prior knowledge. However, if the adversary cannot identify the true location so that $Pr(\mathbf{u}^* = \mathbf{s}_i) = 1$ for any $\mathbf{s}_i \in \mathcal{S}$, Definition 3.7 is always satisfied. On the other hand, if an "omnipotent" adversary already knows the true location, then no mechanism can actually protect location privacy.

3.2.4 Comparison with Other Definitions

Differential Privacy. Since the concept of neighboring databases is not generally applicable (as discussed earlier), induced neighborhood [36], metric based neighborhood [8] and δ-neighborhood [21] were proposed. The general idea is that the neighborhood can be formulated by some constraints of data or distance (metric) functions instead of adding or removing a record. However, applying these neighborhood based differential privacy is not feasible in our model because there is only one sole tuple (location) at each timestamp without any "neighbors". Hence we define δ-location set to extend the notion of "neighborhood".

Geo-indistinguishability. Another closely related definition is the Geo-indistinguishability [2], which protects a user's location within a radius (circle) with a "generalized differential privacy" guarantee. In other words, the neighborhood is defined with Euclidian distance. Nevertheless, such spatial perturbation technique may not be reasonable in reality. For example, as shown in Figure 1.1, the "generalized differential privacy" can still be breached given the road network constraint or user's moving pattern (which is represented by Markov model). Thus location privacy must be protected under temporal correlations.

Blowfish privacy. Our privacy definition shares the same insight as the unconstrained Blowfish privacy framework [32] in statistical data release context, which uses secret pairs and privacy policy to build a subset of possible database instances as "neighbors". We show that δ-location set based differential privacy can be instantiated as a special case of unconstrained Blowfish privacy at each timestamp.

Theorem 3.2. *Let $\mathcal{S}$ be the domain of all possible locations. Let G be a complete graph where each node denotes a location in $\mathcal{S}$. Let $\Delta \boldsymbol{X}$ be a condition such that $\boldsymbol{x}^* \in \Delta \boldsymbol{X}$. At each timestamp, Definition 3.6 is equivalent to $\{\epsilon, \{\mathcal{S}, G, \Delta \boldsymbol{X}\}$-Blowfish privacy.*

3.2.5 Discussion

Learning Markov Model. Existing methods such as the knowledge construction module in [57] or EM method in HMM can be used to acquire the transition matrix $\mathbf{M}$, which will not be discussed in this dissertation. However, depending on the power of adversaries, two typical $\mathbf{M}$ can be learned.

I Popular $\mathbf{M}$ can be learned from public transit data.

II Personal $\mathbf{M}$ can be derived with personal transit data[1].

No matter which $\mathbf{M}$ is adopted in our framework, the adversarial knowledge is always bounded, as discussed before. However, the usefulness of released locations may vary for different adversaries. We also compare the two models in our experiments.

[1]For example, mobile apps, like Google Now, may have a user's location history to derive the user's moving pattern.

When Markov Model is not Accurate. When the location data does not exactly follow a Markov model or the learned Markov model is not accurate enough, our framework still guarantees differential privacy, but may generate more error. For example, when a user starts to go to a new place for the first time, which has not been reflected in the learned Markov model, a drift happens. In this case, our mechanism still works because we can handle the drift case. Nevertheless, the utility may be downgraded, especially when the new place is far from the "probable" locations.

Composibility. Since we only need to release one perturbed location at a timestamp, the sequential composition [46] is not applicable. Otherwise, for multiple releases at a timestamp the composition of ϵ holds. On the other hand, given a series of perturbed locations $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_t\}$ released from timestamp 1 to t, a new problem is how to protect and measure the overall privacy guarantee of the entire trace. We defer this to future work.

3.3 Sensitivity Hull

The notion of sensitivity indicates the differences between any two query answers from two instances in neighboring databases. However, in multidimensional space, we show that ℓ_1-norm sensitivity (in Definition 3.2) fails to capture the exact sensitivity. Thus we propose a new notion, sensitivity hull. Note that sensitivity hull is an independent notion from the context of location privacy and can be plugged in any data-independent perturbation mechanisms.

3.3.1 Sensitivity Hull

To derive the meaning of sensitivity, let us consider the following example in traditional setting of differential privacy.

Example 3.1. *Assume we have an employee table T with attributes gender and income. Then we answer the following query workload f:*

$$f_1: \ Select\ count(*)\ from\ T\ where\ gender = \text{“}female\text{”}$$
$$f_2: \ Select\ count(*)\ from\ T\ where\ income > 50000$$

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be neighboring databases so that $\mathbf{x}_1$ is equal to $\mathbf{x}_2$ adding or removing *a random user*. Suppose $f(\mathbf{x}_2) = [10, 20]^\intercal$. Then the possible answers for $f(\mathbf{x}_1)$ could be one of the following columns, from which Δf can be derived.

$$f(\mathbf{x}_1) = \begin{bmatrix} 11 & 10 & 10 & 11 & 9 & 9 & 10 \\ 21 & 21 & 20 & 20 & 20 & 19 & 19 \end{bmatrix}$$
$$\Delta f = f(\mathbf{x}_1) - f(\mathbf{x}_2) = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & -1 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}$$
$$S_f = max||\Delta f||_1 = 2 \quad (\ell_1\text{-norm sensitivity})$$

In Figure 3.2, the dashed lines form the set of $||\Delta f||_1 = 2$ because the ℓ_1-norm sensitivity is 2. However, Δf only consists of all the "•" points. It is obvious that the ℓ_1-norm sensitivity exaggerates the "real sensitivity". To capture the geometric representation of Δf in multidimensional space, we define sensitivity hull (the solid lines in Figure 3.2) as follows.

Definition 3.8 (Sensitivity Hull)**.** *The sensitivity hull of a query f is the*

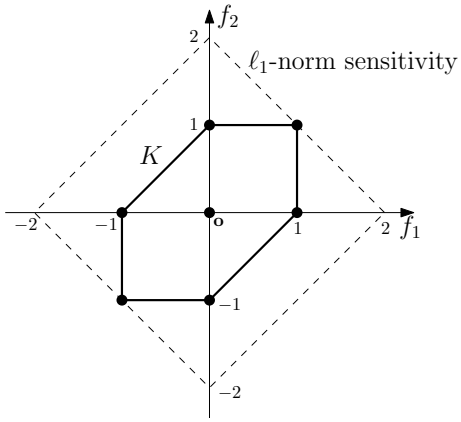


Figure 3.2: Sensitivity hull of Example 3.1. Solid lines denote the sensitivity hull K; dashed lines are the ℓ_1-norm sensitivity.

convex hull of Δf *where* Δf *is the set of* $f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)$ *for any pair* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *in* δ*-location set* $\Delta \boldsymbol{X}$.

$$K = Conv\,(\Delta f)$$

$$\Delta f = \underset{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \Delta \boldsymbol{X}}{\cup} (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2))$$

Theorem 3.3. *A sensitivity hull* K *is centrally symmetric: if* $\boldsymbol{v} \in K$ *then* $-\boldsymbol{v} \in K$.

Theorem 3.4. *If data* $\boldsymbol{x}$ *is in discrete domain, then for any* $f : \boldsymbol{x} \to \mathbb{R}^d$, *the sensitivity hull of* f *is a polytope in* $\mathbb{R}^d$.

3.3.2 Error Bound of Differential Privacy

We extend the error bound of differential privacy in database context [29] to our location setting using sensitivity hull.

Lemma 3.2. *Suppose* $\boldsymbol{F} : \mathbb{R}^N \to \mathbb{R}^d$ *is an aggregate function. When neighboring databases are obtained by changing an attribute, the sensitivity hull* K *of* $\boldsymbol{F}$ *is a polytope* $\boldsymbol{F}\boldsymbol{B}_1^N$ *where* $\boldsymbol{B}_1^N$ *is the* N*-dimensional unit* ℓ_1 *ball.*

Proof. Because the true data $\mathbf{x} \in \mathbb{R}^N$, any change (by changing an attribute) of $\mathbf{x}^N$ will result in a ± 1 operation in one of the N dimensions. Thus the convex hull for $\mathbf{x}$ changes is $\mathbf{B}_1^N$. The changes in $\mathbf{x}$ will cause variations in output domain by $\mathbf{FB}_1^N$. Because linear transformations keep the convexity, then $K = \mathbf{FB}_1^N$. □

Lemma 3.3 ([29]). [2] *To answer a linear function $\boldsymbol{F} : \mathbb{R}^N \to \mathbb{R}^d$ under Definition 3.1, every ϵ-differentially private mechanism must have*

$$\textsc{Error} \geqslant \Omega\left(\frac{d}{\epsilon}\left(\frac{\textsc{Vol}(K)}{\textsc{Vol}(B_2^d)}\right)^{1/d}\right)$$

where $\textsc{Vol}(\cdot)$ *is the Volume and* B_2^d *is the unit* ℓ_2 *ball.*

Then we can derive the lower bound of dynamic differential privacy as follows.

Theorem 3.5 (Lower Bound). *Let K be the sensitivity hull of δ-location set $\Delta \boldsymbol{X}$. To satisfy Definition 3.6, every mechanism must have*

$$\textsc{Error} \geqslant \Omega\left(\frac{1}{\epsilon}\sqrt{\textsc{Area}(K)}\right)$$

where $\textsc{Area}(K)$ *is the area of* K.

Proof. Because $\mathbf{x}$ is in discrete domain (from Markov states), K is a polygon (two-dimensional polytope). The number of vertices must be an even number $2N$ because K is symmetric. There must be a matrix $\mathbf{F} \in \mathbb{R}^{2\times N}$ that satisfies $K = \mathbf{FB}_1^N$ where $\mathbf{B}_1^N$ is the N-dimensional unit ℓ_1 ball. Therefore, answering $\mathbf{F}$ is the same problem as locating a point from K, which is our geometric

[2] To check the correctness, we refer readers to the latest version of [29] at `http://mrtz.org/papers/HT10geometry.pdf`.

location problem. In our setting, the output domain is two-dimensional. The volume also becomes area of K. Then we obtain above lower bound. □

3.4 Location Release Algorithm

3.4.1 Framework

The framework of our proposed location release algorithm is shown in Algorithm 1. At each timestamp, say t, we compute the prior probability vector $\mathbf{p}_t^-$. If the location needs to be released, we construct a δ-location set $\Delta\mathbf{X}_t$. Then if the true location $\mathbf{x}^*$ is excluded in $\Delta\mathbf{X}_t$ (a drift), we use surrogate to replace $\mathbf{x}^*$. Next a differentially private mechanism (like Algorithm 2 which will be presented next) can be adopted to release a perturbed location $\mathbf{z}_t$. In the meantime, the released $\mathbf{z}_t$ will also be used to update the posterior probability $\mathbf{p}_t^+$ (in the equation below) by Equation (3.1), which subsequently will be used to compute the prior probability for the next timestamp $t+1$. Then at timestamp $t+1$, the above process is repeated.

$$\mathbf{p}_t^+[i] = Pr(\mathbf{u}_t^* = \mathbf{s}_i | \mathbf{z}_t, \mathbf{z}_{t-1}, \cdots, \mathbf{z}_1)$$

Theorem 3.6. *At any timestamp t, Algorithm 1 is ϵ_t-differentially private on* 0*-location set.*

Proof. It is equivalent to prove adversarial privacy on 0-location set, which includes all possible locations. If $\mathbf{x}_t^* \in \Delta\mathbf{X}_t$, then $\mathbf{z}_t$ is generated by $\mathbf{x}_t^*$. By Theorem 3.8, $\mathbf{z}_t$ is ϵ_t-differentially private. So $\frac{Pr(\mathbf{u}_t^*=\mathbf{s}_i|\mathbf{z}_t)}{Pr(\mathbf{u}_t^*=\mathbf{s}_i)} \leqslant e^{\epsilon}$. When $\mathbf{x}_t^* \notin$

Algorithm 1 Location Release Framework

Require: ϵ_t, δ, $\mathbf{M}$, $\mathbf{p}_{t-1}^+$, $\mathbf{x}_t^*$

1: $\mathbf{p}_t^- \leftarrow \mathbf{p}_{t-1}^+\mathbf{M}$; ▷ Markov transition
2: **if** location needs to be released **then**
3: Construct $\Delta\mathbf{X}_t$; ▷ δ-location set
4: **if** $\mathbf{x}_t^* \notin \Delta\mathbf{X}_t$ **then** ▷ a drift
5: $\mathbf{x}_t^* \leftarrow$ surrogate;
6: **end if**
7: $\mathbf{z}_t \leftarrow$ALGORITHM 2(ϵ_t, $\Delta\mathbf{X}_t$, $\mathbf{x}_t^*$); ▷ release $\mathbf{z}_t$
8: Derive posterior probability $\mathbf{p}_t^+$ by Equation (3.1);
9: **end if**
10: **return** ALGORITHM 1(ϵ_{t+1}, δ, $\mathbf{M}$, $\mathbf{p}_t^+$, $\mathbf{x}_{t+1}^*$); ▷ go to next timestamp

$\Delta\mathbf{X}_t$, then a surrogate $\tilde{\mathbf{x}}_t$ replaces $\mathbf{x}_t^*$. Then

$$\frac{Pr(\mathbf{u}_t^* = \mathbf{s}_i|\mathbf{z}_t)}{Pr(\mathbf{u}_t^* = \mathbf{s}_i)} = \frac{\sum_k Pr(\mathbf{u}_t^* = \mathbf{s}_i|\tilde{\mathbf{x}}_t = \mathbf{s}_k)Pr(\tilde{\mathbf{x}}_t = \mathbf{s}_k|\mathbf{z}_t)}{\sum_k Pr(\mathbf{u}_t^* = \mathbf{s}_i|\tilde{\mathbf{x}}_t = \mathbf{s}_k)Pr(\tilde{\mathbf{x}}_t = \mathbf{s}_k)} \leqslant e^{\epsilon}$$

Therefore, by equivalence (Theorem 3.1) Algorithm 1 is ϵ_t-differentially private on 0-location set. □

Laplace Mechanism. With the ℓ_1-norm sensitivity in Definition 3.2, Laplace mechanism (LM) can be adopted in Line 7 of Algorithm 1. The problem of this approach is that it will over-perturb a location because ℓ_1-norm sensitivity could be much larger than the sensitivity hull, as discussed in Section 3.3. We use LM with δ-location set as a baseline in our experiment.

3.4.2 Planar Isotropic Mechanism

Because we showed (in Lemma 3.2) that the sensitivity hull of a query matrix is a polytope (polygon in our two-dimensional location setting), the state-of-art K-norm based mechanism [6, 29, 50] can be used.

Definition 3.9 (K-norm Mechanism [29])**.** *Given a linear function* $\boldsymbol{F} : \mathbb{R}^N \rightarrow$

$\mathbb{R}^d$ *and its sensitivity hull* K*, a mechanism is* K*-norm mechanism if for any output* $\boldsymbol{z}$*, the following holds:*

$$Pr(\boldsymbol{z}) = \frac{1}{\Gamma(d+1)\text{VOL}(K/\epsilon)} exp\left(-\epsilon||\boldsymbol{z} - \boldsymbol{F}\boldsymbol{x}^*||_K\right) \tag{3.5}$$

where $\boldsymbol{F}\boldsymbol{x}^*$ *is the true answer,* $||\cdot||_K$ *is the (Minkowski) norm of* K*,* $\Gamma()$ *is Gamma function and* $\text{VOL}()$ *indicates volume.*

However, standard K-norm mechanism was designed for high-dimensional structure of sensitivity hull, whereas in our problem a location is only two-dimensional. Thus we can further optimize K-norm mechanism to achieve the lower bound of differential privacy. We propose a Planar Isotropic Mechanism (PIM) based on K-norm mechanism as follows.

Rationale. The rationale of PIM is that in two-dimensional space we efficiently transform the sensitivity hull to its isotropic position[3] so that the optimality is guaranteed.

Theorem 3.7. *[29] If the sensitivity hull* K *is in* C*-appro-ximately isotropic position, then* K*-norm mechanism has error* $O(C)\text{LB}(K)$ *where* $\text{LB}(K)$ *is the lower bound of differential privacy.*

From Theorem 3.7, we know that K-norm mechanism would be the optimal solution if the sensitivity hull K is in isotropic position, denoted by K_I. Although in high-dimensional space transforming a convex body to its isotropic position is extremely expensive, it is feasible in two-dimensional space. To this end, we need the following corollary (which can be derived from [45, 56]).

[3]We refer readers to [3, 48] for a detailed study of isotropic position.

Corollary 3.1 (Isotropic Transformation)**.** *For any convex body K in $\mathbb{R}^2$, any integer $p \geqslant 1$, there is an absolute constant c such that if $l \geqslant 4cp^2$, with probability at least $1 - 2^{-p}$, $K_I = \boldsymbol{T}K$ is in isotropic position.*

$$\boldsymbol{T} = \left(\frac{1}{l} \sum_{i=1}^{l} \boldsymbol{y}_i \boldsymbol{y}_i^{\intercal} \right)^{-\frac{1}{2}} \tag{3.6}$$

where $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_l$ are independent random points uniformly distributed in K.

Therefore, the isotropic transformation of any sensitivity hull K can be fulfilled by sampling, which is a trivial task in two-dimensional space. For instance, a hit-and-run algorithm [44] only takes $O(log^3(1/\delta))$ time where δ is an error parameter. We skip the sampling details and refer readers to the survey paper of Santosh Vempala [59] for a complete study.

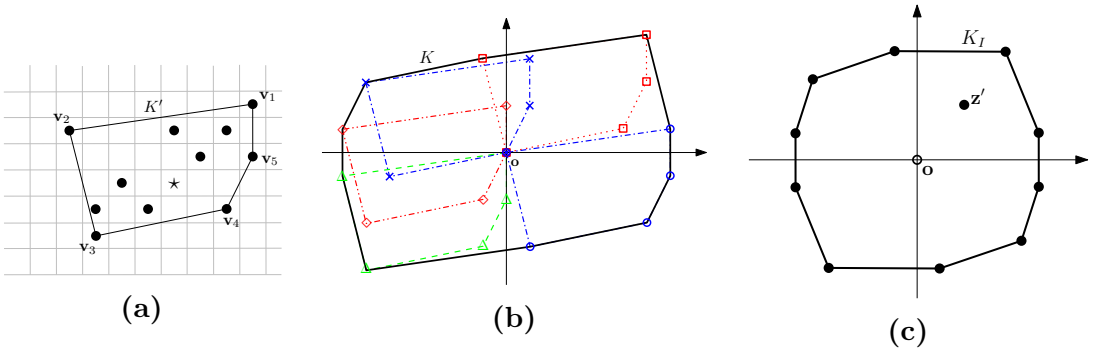


Figure 3.3: (a) Convex hull of $\Delta\mathbf{X}$. (b) Finding the sensitivity hull K. (c) Transform K to isotropic position K_I. Sample a point $\mathbf{z}'$.

Algorithm. As an overview, PIM involves the following steps:

(1) Compute sensitivity hull K from $\Delta\mathbf{X}$;

(2) Transform K to isotropic position K_I;

(3) generating a noise in the space of K_I by K-norm mechanism;

(4) Transform to the original space.

We first describe how to compute sensitivity hull K. Suppose we have a

δ-location set $\Delta\mathbf{X}$ at a timestamp. We can first derive the convex hull of $\Delta\mathbf{X}$, denoted by $K' = Conv(\Delta\mathbf{X})$. For example, in Figure 3.3a, the convex hull K' is shown by the black lines given δ-location set as "$\bullet$" and "$\star$" where "$\star$" is the true location. Denote $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_h$ the vertices of K'. Then we use a set $\Delta\mathbf{V}$ to store $\mathbf{v}_i - \mathbf{v}_j$ for any $\mathbf{v}_i$ and $\mathbf{v}_j$ from the vertices of K' as the equation below. In Figure 3.3b, for instance, the polygon "$\triangle \cdots \triangle$" denotes $\mathbf{v}_i - \mathbf{v}_1$ for all $\mathbf{v}_i$. Then $Conv(\Delta\mathbf{V})$ will be the sensitivity hull K of the δ-location set, as shown by the polygon with solid lines in Figure 3.3b.

$$K = Conv(\Delta\mathbf{V})$$

$$\Delta\mathbf{V} = \bigcup_{\mathbf{v}_1,\mathbf{v}_2 \in \text{ vertices of } K'} (\mathbf{v}_1 - \mathbf{v}_2)$$

Next we transform K to its isotropic position K_I. We sample $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_l$ uniformly from K. Then a matrix $\mathbf{T}$ can be derived by Equation (3.6). To verify if $\mathbf{T}$ is stable, we can derive another $\mathbf{T}'$. If the Frobenius norm $||\mathbf{T}' - \mathbf{T}||_F$ is small enough (e.g. $< 10^{-3}$), then we accept $\mathbf{T}$. Otherwise we repeat above process with larger l. In the end, $K_I = \mathbf{T}K$ is the isotropic position of K, as shown in Figure 3.3c.

Next a point $\mathbf{z}'$ can be uniformly sampled from K_I. We generate a random variable r from Gamma distribution $\Gamma(3, \epsilon^{-1})$. Let $\mathbf{z}' = r\mathbf{z}'$. Then we transform the point $\mathbf{z}'$ to the original space by $\mathbf{z}' = \mathbf{T}^{-1}\mathbf{z}'$. The released location is $\mathbf{z} = \mathbf{x}^* + \mathbf{z}'$.

Algorithm 2 summarizes the process of PIM. Lines 5∼6 can be iterated until $\mathbf{T}$ is stable, whereas the computational complexity is not affected by the iterations because the number of samples is bounded by a constant (by Corollary 3.1).

Algorithm 2 Planar Isotropic Mechanism

Require: ϵ, $\Delta\mathbf{X}$, $\mathbf{x}^*$

1: $K' \leftarrow Conv(\Delta\mathbf{X})$; ▷ convex hull of ΔX

2: $\Delta\mathbf{V} \longleftarrow \bigcup_{\mathbf{v}_1,\mathbf{v}_2 \in \text{ vertices of } K'} (\mathbf{v}_1 - \mathbf{v}_2)$;

3: $K \leftarrow Conv(\Delta\mathbf{V})$; ▷ sensitivity hull

4: Repeat lines 5,6 with larger l if $\mathbf{T}$ is not stable:

5: Sample $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_l$ uniformly from K;

6: $\mathbf{T} \leftarrow \left(\frac{1}{l}\sum_{i=1}^{l} \mathbf{y}_i \mathbf{y}_i^\intercal\right)^{-\frac{1}{2}}$;

7: $K_I = \mathbf{T}K$; ▷ isotropic transformation

8: Uniformly sample $\mathbf{z}'$ from K_I;

9: Sample $r \sim \Gamma(3, \epsilon^{-1})$;

10: **return** $\mathbf{z} = \mathbf{x}^* + r\mathbf{T}^{-1}\mathbf{z}'$; ▷ release z

Privacy and Performance Analysis. We now present the privacy property, complexity, and the error of PIM.

Theorem 3.8. *Algorithm 2 is ϵ-differentially private on δ-location set $\Delta\boldsymbol{X}$.*

Proof. The isotropic transformation is a unique $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ mapping. In the space of K_I, it is easy to prove that the probability distribution of $\mathbf{z}'$ is equal to Equation (3.5) in which K_I is in place of K. Therefore, Algorithm 2 is ϵ-differentially private. □

Theorem 3.9. *Algorithm 2 takes $O(nlog(h) + h^2log(h))$ time where n is the size of $\Delta\boldsymbol{X}$ and h is number of vertices on $Conv(\Delta\boldsymbol{X})$.*

Proof. Line 1 takes time $O(nlog(h))$ where n is the size of $\Delta\mathbf{X}$ and h is number of vertices on $Conv(\Delta\mathbf{X})$. Line 3 takes time $O(h^2log(h))$. Because the number of samples is bounded by a constant, lines 4∼6 need $O(1)$ time. Thus the overall complexity is $O(nlog(h) + h^2log(h))$. □

Theorem 3.10. *Algorithm 2 has error $O\left(\frac{1}{\epsilon}\sqrt{\text{AREA}(K)}\right)$ at most, which means it achieves the lower bound in Theorem 3.5.*

Proof. From Equation (3.5), we can obtain the following error in the isotropic space

$$\textsc{Error}^2 = \int_{\mathbf{x}} Pr(\mathbf{x})||\mathbf{x}||_2^2 d\mathbf{x} = \frac{12det(\mathbf{T}^{-1})}{\epsilon^2 \textsc{Area}(K_I)} \mathop{\mathbb{E}}_{\mathbf{x}\in K_I} ||\mathbf{x}||_2^2$$

where $det(\mathbf{T}^{-1})$ is the determinant of $\mathbf{T}^{-1}$. By the isotropic property,

$$\textsc{Error}^2 = \frac{12L_{K_I}^2}{\epsilon^2} det(\mathbf{T}^{-1})\textsc{Area}(K_I)$$

where L_{K_I} is the isotropic constant of K_I. After transforming back to the original space, it is easy to show that $det(\mathbf{T}^{-1})\textsc{Area}(K_I)$ becomes $\textsc{Area}(K)$. Therefore, Algorithm 2 has error at most $O(\frac{1}{\epsilon}\sqrt{\textsc{Area}(K)})$, which is the lower bound in Theorem 3.5. Hence Algorithm 2 is optimal. □

3.4.3 Location Inference

The inference of line 8 in Algorithm 1 is a general statement because inference methods depend on specific release algorithms. To implement the inference for PIM, we need to transform the location $\mathbf{s}_i$ and the released location $\mathbf{z}_t$ to the isotropic space of K_I. Then in Equation (3.1), the probability $Pr(\mathbf{z}_t|\mathbf{u}_t^* = \mathbf{s}_i)$ can be computed as follows. This completes the whole algorithm.

$$Pr(\mathbf{z}_t|\mathbf{u}_t^* = \mathbf{s}_i) = \frac{\epsilon^2}{2\textsc{Area}(K_I)} exp(-\epsilon||\mathbf{z}_t' - \mathbf{s}_i'||_{K_I})$$

$$\mathbf{z}_t' = \mathbf{T}\mathbf{z};\ \mathbf{s}_i' = \mathbf{T}\mathbf{s}_i$$

3.5 Experimental Evaluation

In this section we present experimental evaluation of our method. All algorithms were implemented in Matlab on a PC with 2.9 GHz Intel i7 CPU and 8 GB Memory.

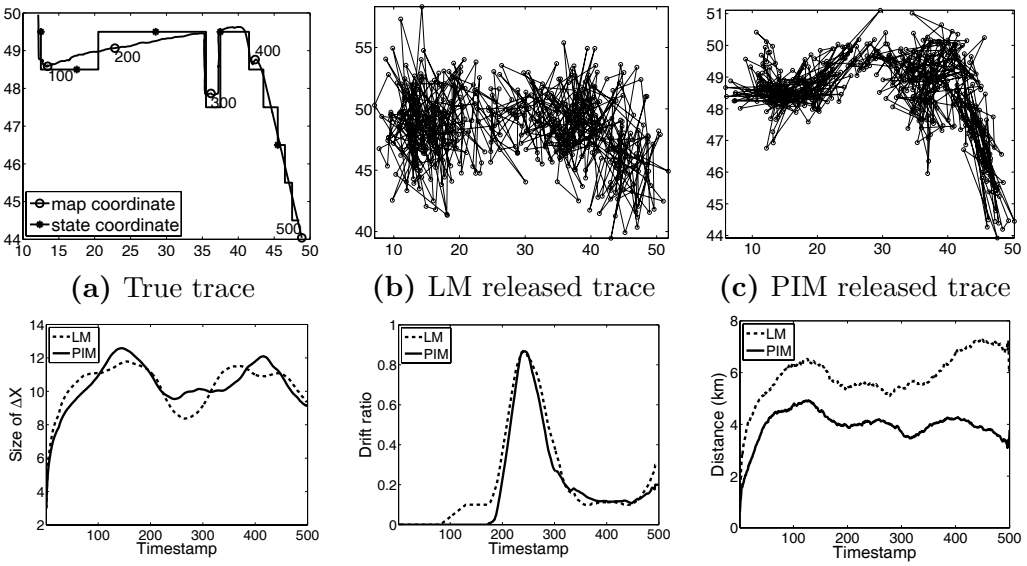
Datasets. We used two real-world datasets.

I Geolife data. Geolife data [64] was collected from 182 users in a period of over three years. It recorded a wide range of users' outdoor movements, represented by a series of tuples containing latitude, longitude and timestamp. The trajectories were updated in a high frequency, e.g. every $1 \sim 60$ seconds. We extracted all the trajectories within the 3rd ring of Beijing to train the Markov model, with the map partitioned into cells of $0.34 \times 0.34\ km^2$.

II Gowalla data. Gowalla data [11] contains $6,442,890$ check-in locations of $196,586$ users over the period of Feb. 2009 to Oct. 2010. We extracted all the check-ins in Los Angeles to train the Markov model, with the map partitioned into cells of $0.89 \times 0.89\ km^2$. Because check-ins were logged in a relatively low frequency, e.g. every $1 \sim 50$ minutes, we can examine the difference of the results from Gowalla and Geolife.

Metrics. We used the following metrics in our experiment, including two internal metrics: size of $\Delta\mathbf{X}$, drift ratio, and two sets of utility metrics: distance, precision and recall. We skip the runtime report because most locations were released within 0.3 second by PIM.

I Since our privacy definition is based on δ-location set $\Delta\mathbf{X}$, we evaluated the size of $\Delta\mathbf{X}$ to understand how $\Delta\mathbf{X}$ grows or changes.



(a) True trace **(b)** LM released trace **(c)** PIM released trace

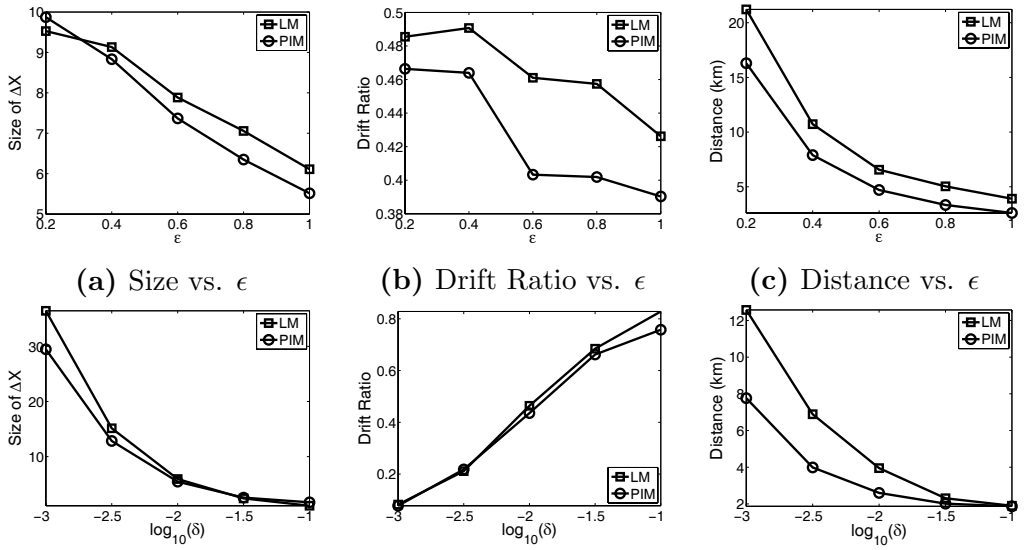
(d) Size of $\Delta\mathbf{X}$ **(e)** Drift ratio **(f)** Distance

Figure 3.4: Performance over time: (a) The true (original) trace; (b)(c) Released traces; (d) Size of $\Delta\mathbf{X}$ over time; (e) Drift ratio over time; (f) Distance over time.

II The definition of $\Delta\mathbf{X}$ and the potential limit of Markov model may cause the true location to fall outside $\Delta\mathbf{X}$ (drift). Thus we measured the drift ratio computed as the number of timestamps the true location is excluded in $\Delta\mathbf{X}$ over total number of timestamps.

III We measured the distance between the released location and the true location, which can be considered as a general utility metric independent of specific location based applications.

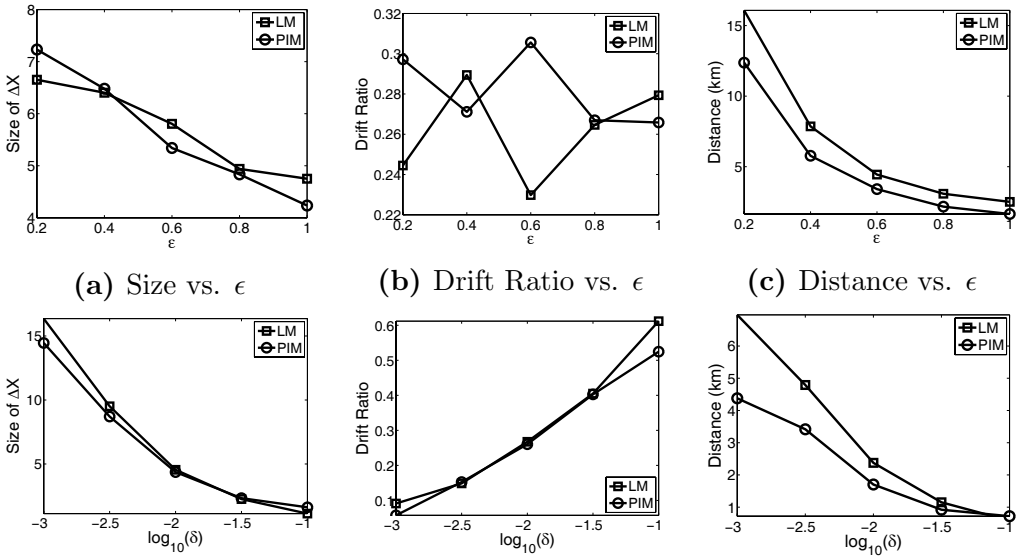
IV We also run k nearest neighbor (kNN) queries using the released locations and report its precision and recall compared to the true kNN set using the original location. Suppose the true kNN set is R, the returned k'NN set (we set $k' \geqslant k$) is R', precision is defined as $|R \cap R'|/k'$, and recall is defined as $|R \cap R'|/k$.



(a) Size vs. ϵ **(b)** Drift Ratio vs. ϵ **(c)** Distance vs. ϵ

(d) Size vs. δ **(e)** Drift Ratio vs. δ **(f)** Distance vs. δ

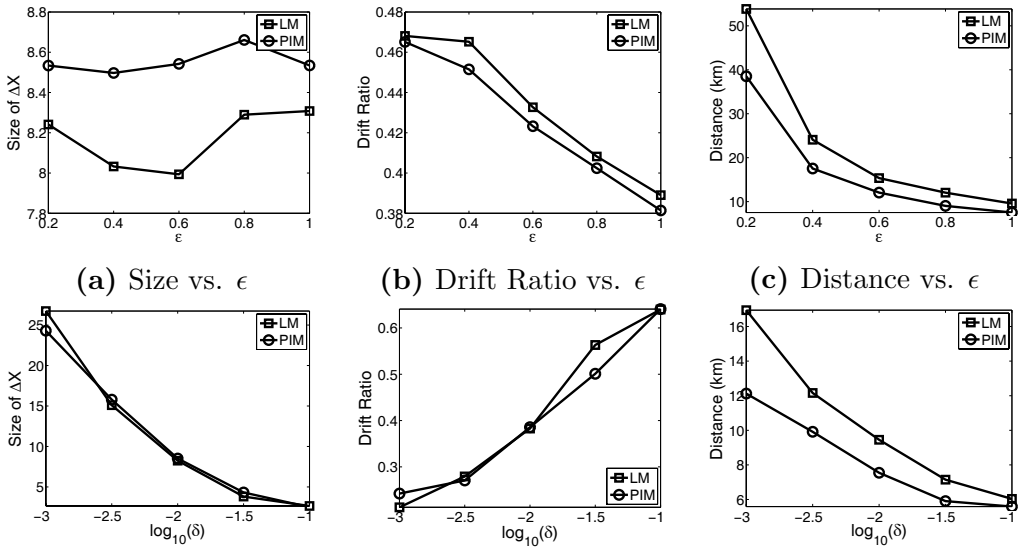
Figure 3.5: Impact of parameters on GeoLife data with popular $\mathbf{M}$: (a)(d) Impact of ϵ and δ on size of $\Delta\mathbf{X}$; (b)(e) Impact of ϵ and δ on drift ratio; (c)(f) Impact of ϵ and δ on distance.



(a) Size vs. ϵ **(b)** Drift Ratio vs. ϵ **(c)** Distance vs. ϵ

(d) Size vs. δ **(e)** Drift Ratio vs. δ **(f)** Distance vs. δ

Figure 3.6: Impact of parameters on GeoLife data with personal $\mathbf{M}$: (a)(d) Impact of ϵ and δ on size of $\Delta\mathbf{X}$; (b)(e) Impact of ϵ and δ on drift ratio; (c)(f) Impact of ϵ and δ on distance.



(a) Size vs. ϵ **(b)** Drift Ratio vs. ϵ **(c)** Distance vs. ϵ

(d) Size vs. δ **(e)** Drift Ratio vs. ϵ **(f)** Distance vs. δ

Figure 3.7: Impact of parameters on Gowalla data with popular $\mathbf{M}$: (a)(d) Impact of ϵ and δ on size of $\Delta\mathbf{X}$; (b)(e) Impact of ϵ and δ on drift ratio; (c)(f) Impact of ϵ and δ on distance.

3.5.1 Performance Over Time

In order to show the performance of a release mechanism as a user moves over time, including how $\Delta\mathbf{X}$ changes, how often drift happens and how accurate is the perturbed location, we first run a set of experiments for a single test trajectory with popular $\mathbf{M}$ learned from all users. We selected a random test trajectory from Geolife dataset consisting of 500 timestamps. We tested both PIM and LM at each timestamp with $\epsilon = 1$ and $\delta = 0.01$. Each method was run 20 times and the average is reported. Figure 3.4a shows the original trajectory in map and state (grid) coordinates; Figures 3.4b and 3.4c show the released (perturbed) locations at each timestamp. We can see that the released locations of PIM is closer to the true location, compared with LM.

Size of $\Delta\mathbf{X}$. From Figure 3.4d we see that the size of $\Delta\mathbf{X}$ does not increase dramatically, instead it maintains at stable level after a few timestamps. The

reason is that by selecting the δ-location set the inference mechanism only boost probabilities of locations in $\Delta\mathbf{X}$. Then the probabilities of other locations decay gradually. Thus a stable δ-location set can be maintained.

Drift Ratio. In Figure 3.4e, the peak of drift ratio happened in timestamp $200 \sim 300$. This can be explained by the fact that the true trajectory has a turning corner as in Figure 3.4a, and the transition probability of making this right turn is relatively small in the Markov model.

When a drift happens, we use surrogate for release mechanisms. Because the surrogate is the nearest cell to the true location in $\Delta\mathbf{X}$ and the release mechanism is based on the surrogate, the posterior probability of the surrogate will be boosted. Consequently, in the next timestamp the probability that $\Delta\mathbf{X}$ includes the previous true location rises. This "lagged catch-up" can be verified by Figures 3.4f, 3.4b and 3.4c.

Distance. The distance is reported in Figure 3.4f. We can see that PIM provided more accurate locations than LM for two reasons. First, because PIM is optimal, the posterior probability distribution is more accurate than LM. Second, with such distribution a better (Bayesian) inference can be obtained, making $\Delta\mathbf{X}$ more accurate for the coming timestamp.

3.5.2 Impact of Parameters

Since the performance may vary for different trajectories, we chose 100 trajectories from 100 users, each of which has 500 timestamps, to evaluate the overall performance and the impact of parameters. The default values are $\epsilon = 1$ and $\delta = 0.01$ if not mentioned. The average performances for both datasets are reported in Figures 3.5 (on GeoLife data with popular $\mathbf{M}$), Figure 3.6 (on Ge-

oLife data with personal $\mathbf{M}$) and Figure 3.7 (on Gowalla data with popular $\mathbf{M}$).

Size of $\Delta\mathbf{X}$ vs. ϵ. In Figures 3.5a and 3.6a (Geolife data), size of $\Delta\mathbf{X}$ shrinks with larger ϵ because the inference result is enhanced by big ϵ. On the other hand, impact of ϵ would be negligible in Gowalla data because one-step transition in Markov model has limited predictability (check-ins are not frequent), as in Figure 3.7a.

Size of $\Delta\mathbf{X}$ vs. δ. Size of $\Delta\mathbf{X}$ is mainly determined by δ as shown in Figures 3.5d, 3.6d and 3.7d. Note that LM and PIM have similar size of $\Delta\mathbf{X}$, meaning the true location is hidden in the similar size of candidates. When δ grows, size of $\Delta\mathbf{X}$ reduces dramatically because more improbable locations are truncated. However, δ cannot be too large because it preserves nearly no privacy if size of $\Delta\mathbf{X}$ is close to 1. Thus we use $\delta = 0.01$ by default, which guarantees the sizes of $\Delta\mathbf{X}$ are larger than 4 in the three settings.

Drift Ratio vs. ϵ. Figures 3.5b and 3.7b show that drift ratio declines with larger ϵ, which is easy to understand because larger ϵ provides more accurate release. However, the impact of ϵ is not obvious in Figure 3.6b. The reason is that the size of $\Delta\mathbf{X}$ is already small as in Figure 3.6d, hence the increase of ϵ does not help much in improving the accuracy of the inference.

Drift Ratio vs. δ. Figures 3.5e, 3.6e and 3.7e show that drift ratio rises when δ increases due to reduced $\Delta\mathbf{X}$, and PIM is slightly better than LM. However, due to the phenomenon of "lagged catch-up", we will see next that the accuracy of the released locations was still improved with increasing δ.

Distance vs. ϵ. Figures 3.5c, 3.6c and 3.7c show the distance with varying ϵ. We can see that PIM performed better than LM. In Gowalla data, because

check-in locations are far away from each other, the distance is larger than Geolife.

Distance vs. δ. Because bigger δ will result in less candidates in δ-location set, the distance declines when δ increases. Figures 3.5f, 3.6f and 3.5f show that PIM achieves better accuracy than LM. However, from $10^{-1.5}$ to 10^{-1}, the improvement on distance is very small while privacy guarantee drops significantly as in Figures 3.5d, 3.6d and 3.7d. Especially in Figure 3.6d, size of $\Delta\mathbf{X}$ is 1 when $\delta = 0.1$. Therefore, choosing a high value of δ (like $\delta > 0.03$) does not provide the best trade-off of privacy and utility.

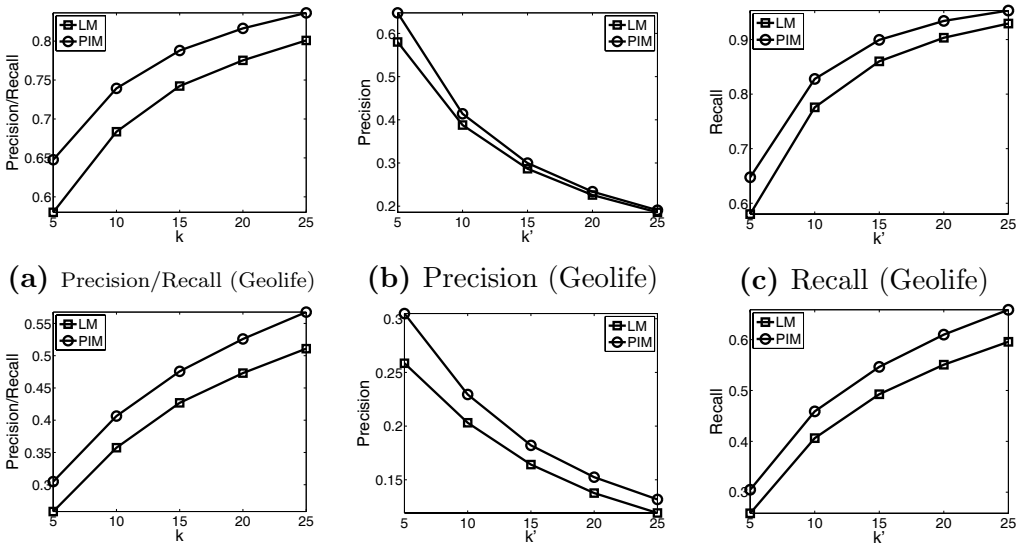
Impact of Markov model. Comparing Figures 3.5 on popular $\mathbf{M}$ and Figure 3.6 on personal $\mathbf{M}$, we can see the impact of different Markov model. With more accurate (personal) model, better utility can be achieved, including smaller size of $\Delta\mathbf{X}$, lower drift ratio and less distance. However, the same privacy level (ϵ-differential privacy) is maintained (on different $\Delta\mathbf{X}$) regardless of $\mathbf{M}$.

3.5.3 Utility for Location Based Queries

To demonstrate the utility of released locations, we also measured the precision and recall of kNN queries at each of the 500 timestamps in the 100 trajectories with popular $\mathbf{M}$. The average results of kNN from original locations and k'NN from released locations are reported in Figure 3.8 with $\epsilon = 1$ and $\delta = 0.01$.

In Figures 3.8a and 3.8d, we show the precision and recall with $k = k'$. Note that in this case precision is equal to recall. We can see that when k grows precision and recall also increase because the nearest neighbors have to be found in larger areas. PIM is consistently better than LM.

Next we fixed $k = 5$ and varied k'. Figures 3.8b and 3.8e show the precision



(a) Precision/Recall (Geolife) **(b)** Precision (Geolife) **(c)** Recall (Geolife)

(d) Precision/Recall (Gowalla) **(e)** Precision (Gowalla) **(f)** Recall (Gowalla)

Figure 3.8: kNN results: (a)(d) precision and recall under $k = k'$; (b)(e) precision vs. k'; (c)(f) recall vs. k'.

drops when k' rises because of a larger returned set. On the other hand, Figures 3.8c and 3.8f indicate recall increases with large k'. Overall, PIM has better precision and recall than LM.

3.6 Conclusion Remarks

In this chapter we proposed δ-location set based differential privacy to protect a user's true location at every timestamp under temporal correlations. We generalized the notion of "neighboring databases" to δ-location set for the new setting and extended the well known ℓ_1-norm sensitivity to sensitivity hull in order to capture the geometric meaning of sensitivity. Then with sensitivity hull we derived the lower bound of δ-location set based differential privacy. To achieve the lower bound, we designed the optimal planar isotropic mechanism to release differentially private locations with significantly high efficiency and utility.

Chapter 4

Customizable Privacy in Hidden Markov Model

Although location privacy can be preserved by Chapter 3, another important factor is that privacy requirements are highly individualized for personal location protection. For example, if a user is at a restaurant, she may prefer to hide her true location but reveal that she is at one of the many restaurants [1]. Such privacy preference cannot be achieved by existing privacy notions. In static aggregate data release context, Kifer et al. [37] proposed a Pufferfish framework to protect secrets of databases among indistinguishable pairs. He et al. [32] further developed a Blowfish framework by generating a policy graph where a node represents a secret to be protected, and an edge represents indistinguishability between the two connected nodes. While the Blowfish framework works well in the static aggregate data release setting with its deterministic constraints, it is not clear how the model can be extended for dynamic data streams where the temporal correlations described by Markov model can be

[1] This is different from k-anonymity because the number of restaurants can be much larger than k, and the released data can still be differentially private.

$\mathcal{S}$	domain of states in Markov model
$\mathbf{s}_i, \mathbf{s}_j, \mathbf{s}_k$	a state in Markov model
$\mathbf{s}^*$	the true state
$\mathcal{C}$	constraint (set)

Table 4.1: Notations in Chapter 4

considered as dynamic and probabilistic constraints.

In this chapter, we first extend the Blowfish notion to a DPHMM notion under temporal correlations. Then we show that the policy graphs in the customizable privacy can be reduced to smaller graphs. To ensure the privacy guarantee, we investigate whether the new graphs also preserves the privacy, measured by a notion of DoP (degree of protection). We study how to generate the optimal graph based on the new graphs with minimum error. The algorithm is then evaluated in the experiment section.

4.1 Preliminaries

We use operators $\cup$ and $\cap$ to denote union and intersection of sets; $|\cdot|$ to denote the number of elements in a set; $||\cdot||_p$ denotes ℓ_p norm; $\overline{\mathbf{ab}}$ denotes a line connecting points $\mathbf{a}$ and $\mathbf{b}$. Table 4.1 summarizes some important symbols for convenience.

4.1.1 Blowfish Privacy

Unlike differential privacy which protects all neighboring databases together, Blowfish privacy only protects the connected secrets in its policy graph.

Definition 4.1 (Blowfish Neighbors [28])**.** *Given a graph G and a set of constraints $\mathcal{C}$, two databases D_1 and D_2 are neighbors if they satisfy the constraint $\mathcal{C}$, and*

- *(Unbounded Blowfish) D_2 can be obtained by adding a tuple to or removing a tuple from D_1 if the tuple (secret) is connected to a "null" node in G.*
- *(Bounded Blowfish) D_1 and D_2 only differ one tuple, whose values in D_1 and D_2 are connected in G.*

A randomized mechanism $\mathcal{A}()$ satisfies ϵ-Blowfish privacy if for any output $\boldsymbol{z}$, $\frac{Pr(\mathcal{A}(D_1)=\boldsymbol{z})}{Pr(\mathcal{A}(D_2)=\boldsymbol{z})} \leqslant e^{\epsilon}$.

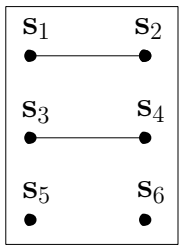
For example, Figure 4.1a is a patients' table where each row is a secret indicating the patient's disease e.g., secrets $\mathbf{s}_3$ and $\mathbf{s}_4$ are "Bob has cancer" and "Bob has diabetes" respectively. For bounded Blowfish privacy, it uses a policy graph to enforce the indistinguishability between the secrets, which can be regarded as *edge protection* in the graph. For instance, Figure 4.1b ensures adversaries cannot distinguish whether Bob has cancer or diabetes by connecting $\mathbf{s}_3$ and $\mathbf{s}_4$. For unbounded Blowfish privacy, it uses a policy graph to disguise *the existence of secrets*. For example, Figure 4.1c connects all secrets to a "null" node, which represents the non-existence of these nodes. Thus the adversaries cannot know whether a secret is real or not. Furthermore, the bounded and unbounded Blowfish privacies can also be combined in one graph by adding the null node into the graph of bounded Blowfish.

4.1.2 Hidden Markov Model

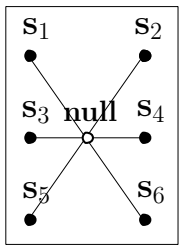
We denote the domain of states by $\mathcal{S}$, $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N\}$ where each $\mathbf{s}_i$ is a unit vector with the ith element being 1 and other $N-1$ elements being 0. We denote $\mathbf{s}^*$ the true state at each timestamp. For privacy protection, $\mathbf{s}^*$ is unobservable to (hidden from) any adversaries. Thus it is an HMM. At timestamp t, we use a vector $\mathbf{p}_t \in [0,1]^{1\times N}$ to denote the probability distribution of

$\mathbf{s}_1$	Alice	cancer
$\mathbf{s}_2$	Alice	asthma
$\mathbf{s}_3$	Bob	cancer
$\mathbf{s}_4$	Bob	diabetes
$\mathbf{s}_5$	Chad	cancer
$\mathbf{s}_6$	Chad	diabetes

(a)



(b)



(c)

Figure 4.1: (a): a table showing patients' diseases with each row being a secret; (b): a policy graph of bounded Blowfish; (c): a policy graph of unbounded Blowfish.

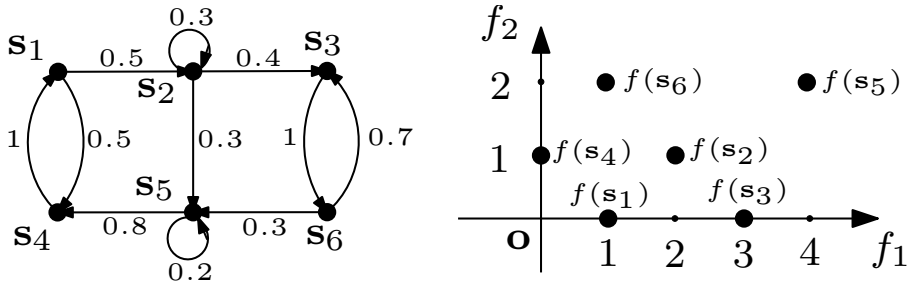


Figure 4.2: (left) a Markov model with transition probabilities; (right) its measurement query in Example 4.2.

true state. Formally, $\mathbf{p}_t[i] = Pr(\mathbf{s}_t^* = \mathbf{s}_i)$ where $\mathbf{p}_t[i]$ is the ith element in $\mathbf{p}_t$ and $\mathbf{s}_i \in \mathcal{S}$.

Example 4.1 (Running Example)**.** *The example in Figure 4.3a is described by a random-walk Markov model in Figure 4.2 (left) where each state denotes a location on the map, if the true state at timestamp t is* $\boldsymbol{s}_1$*, then* $\boldsymbol{s}_t^* = \boldsymbol{s}_1$ *=[1 0 0 0 0 0],* $\boldsymbol{p}_t$ *=[1 0 0 0 0 0].*

Measurement Query. At each timestamp, a measurement query $f : \mathcal{S} \to \mathbb{R}^d$ about current state is evaluated. We denote the space containing all possible outputs of f by measurement space.

Example 4.2 (Measurement Query)**.** *Let* $f : \mathcal{S} \to \mathbb{R}^2$ *be two quantities about*

the true state in Figure 4.2:

$$f_1: \textit{ temperatue of current state}$$

$$f_2: \textit{ noise level of current state}$$

Then f *can be expressed as* $f(\boldsymbol{s}) = \begin{bmatrix} 1 & 2 & 3 & 0 & 4 & 1 \\ 0 & 1 & 0 & 1 & 2 & 2 \end{bmatrix} \boldsymbol{s}^\intercal$ *where each column corresponds the answer of a state, e.g.* $f(\boldsymbol{s}_1) = [1, 0]^\intercal, f(\boldsymbol{s}_2) = [2, 1]^\intercal$. *Above answer can be denoted in measurement space, as in Figure 4.2 (right).*

4.2 Problem Statement

We describe the problem and discuss the challenges when applying the Blowfish framework in the new setting of hidden Markov model.

Relationship of policies. Although the Blowfish policy works well individually, the relationship of different policies has not been studied so far. Consider the following examples.

- Given the policy graph in Figure 4.1b which protects $\mathbf{s}_1 \sim \mathbf{s}_4$, are secrets $\mathbf{s}_5$ and $\mathbf{s}_6$ also protected? Since they are disconnected in the graph, shall we assume they are disclosed? If so, is it necessary to connect them to a null node, or connect them to other nodes (and which)?

- If $\{\epsilon, G\}$-Blowfish privacy is preserved where G is the graph in Figure 4.1b, what is the privacy guarantee for all the secrets $\{\mathbf{s}_1 \sim \mathbf{s}_6\}$, i.e., what is the relationship between Blowfish privacy and differential privacy?

We will answer above questions in Example 4.4, followed with theoretical result.

4.2.1 Probabilistic Constraint

Next we discuss how the above problems emerge in our setting of hidden Markov model. We can first try to apply Blowfish privacy in Markov model potentially. For example, Figure 4.3a shows the Markov model of a moving user with 6 states, denoted by $\{\mathbf{s}_1, \cdots, \mathbf{s}_6\}$. If the user prefers to hide her state in 3 categories, i.e., cafeteria, school and grocery (the octagon, circle and square in Figure 4.3a), the privacy customization can be achieved by the graph in Figure 4.3b. Then if the user is at state $\mathbf{s}_5$, the graph ensures that $\{\mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ are indistinguishable.

A main difference between Blowfish framework and our framework is the constraint type. In Blowfish framework, the constraints are deterministic, which leads to the NP-hard complexity [32]. Whereas the constraints in Markov model are probabilistic. For example, in Blowfish framework, Bob can have cancer and diabetes at the same time, or no disease at all. However, in Markov model, there has to and can only exist ONE state, which means the existence of one state excludes all other states. Such rigid constraints pose higher privacy risk than in Blowfish framework.

At any timestamp t, the prior probability $\mathbf{p}_t^-$ can be derived as

$$\mathbf{p}_t^-[i] = Pr(\mathbf{s}_t^* = \mathbf{s}_i | \mathbf{z}_{t-1}, \cdots, \mathbf{z}_1)$$

Clearly, with $\mathbf{p}_t^-$ the states can be divided into two sets: $\mathbf{p}_t^- = 0$ and $\mathbf{p}_t^- > 0$, leading to the following consequences: (1) For the non-existing states ($\mathbf{p}_t^-[i] = 0$), the unbounded Blowfish privacy is meaningless. (2) For the possible states ($\mathbf{p}_t^-[i] > 0$), unbounded Blowfish becomes bounded Blowfish privacy automatically. (3) Bounded Blowfish privacy only holds for the possible states

because all edges connecting the non-existing states disappear in the policy graph. Therefore, we focus on the bounded Blowfish, which means a state is mixed with other states in the graph.

Without ambiguity, we define the constraint of Markov model as the set of states with $\mathbf{p}_t^- > 0$.

Definition 4.2 (Constraint)**.** *Let $\boldsymbol{p}_t^-$ be the prior probability at timestamp t. Constraint $\mathcal{C}_t$ consists of all states satisfying the constraint $\boldsymbol{p}_t^-[i] > 0$.*

$$\mathcal{C}_t := \{\boldsymbol{s}_i | \boldsymbol{p}_t^-[i] > 0, \forall \boldsymbol{s}_i \in \mathcal{S}\}$$

Under the constraint $\mathcal{C}_t$, the policy graph may be reduced. In Figure 4.3, assume the user moved from $\mathbf{s}_1$ to $\mathbf{s}_5$. If an adversary infers by temporal correlations that the true state can only be $\{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_5\}$, the shaded area in Figure 4.3c, is the graph in Figure 4.3b still applicable? In this case, although $\mathbf{s}_5$ is connected to $\mathbf{s}_4$ and $\mathbf{s}_6$, the adversary can eliminate $\mathbf{s}_4$ and $\mathbf{s}_6$ with the knowledge (constraint). In consequence, the original edges $\overline{\mathbf{s}_4\mathbf{s}_5}$ and $\overline{\mathbf{s}_5\mathbf{s}_6}$ disappear, as shown in Figure 4.3d. Then the following questions arise: is $\mathbf{s}_5$ still protected? If not, how to re-generate a new graph to protect $\mathbf{s}_5$ based on the current graph? We will answer these questions in Examples 4.6 and 4.7.

4.2.2 Problem Statement

We summarize our problem as follows. Given an initial state (or probability), a Markov model and a customizable policy graph, how to answer a measurement query $f : \mathcal{S} \to \mathbb{R}^d$ at each timestamp under the HMM assumptions? First, the Markov model can be known to any adversaries. Second, all the previously released answers (observable) can be accessed by adversaries to make inference

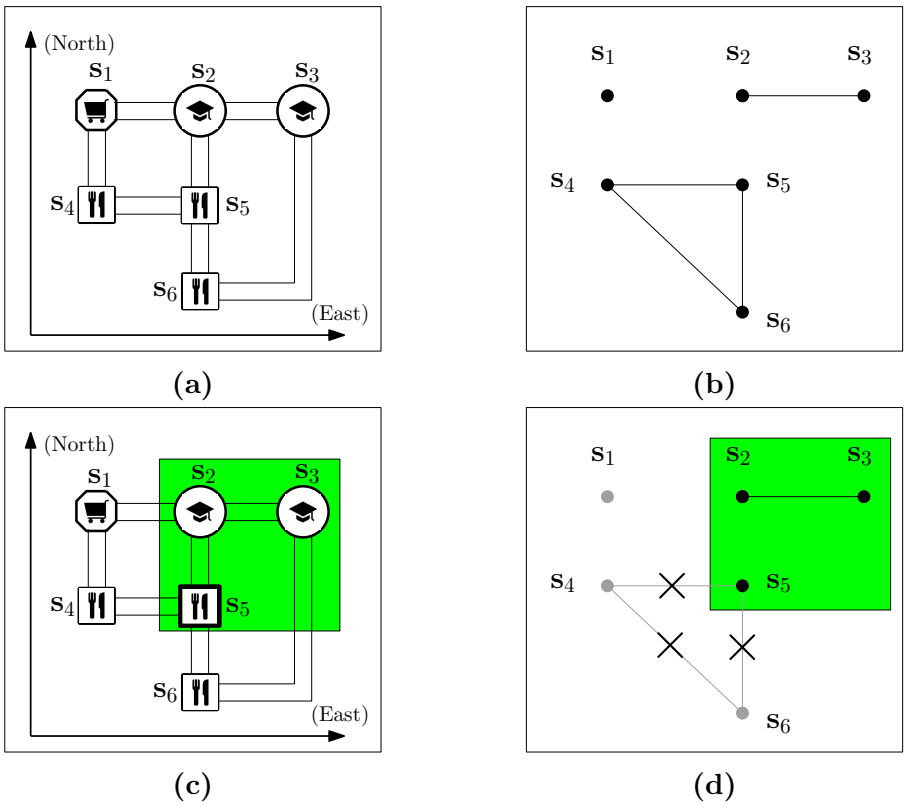


Figure 4.3: Running example. (a): protecting a state in its category (the octagon, circles or squares); (b): the policy graph connecting all nodes in a category; (c): an adversary estimated that the true location can only be $\{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_5\}$; (d): the reduced graph from (b) with the constraint in (c).

about the true state. Third, the data release mechanism is transparent to adversaries. The released answer $\mathbf{z}_t$ should have the following properties: (1) it guarantees a privacy notion to protect the true state; (2) it minimizes the error, measured by the ℓ_2 distance $\text{ERROR} = \sqrt{\mathbb{E}||\mathbf{z}_t - f(\mathbf{s}_t^*)||_2^2}$ between the released answer $\mathbf{z}_t$ and the true answer $f(\mathbf{s}^*)$.

Learning the Markov Model. A Markov model can be learned from publicly available data or perturbed personal data using standard methods, such as EM algorithm. Even if an adversary can obtain such a model, we still need to protect the true state. In the DPHMM, we assume the Markov model has been learned, and is also known to any adversaries.

Incomplete Model. Depending on the power of adversaries, an incomplete (inaccurate) Markov model can be used by adversaries. In this case, the privacy is still guaranteed while the inference result may be downgraded for the adversary (Section 4.5.3).

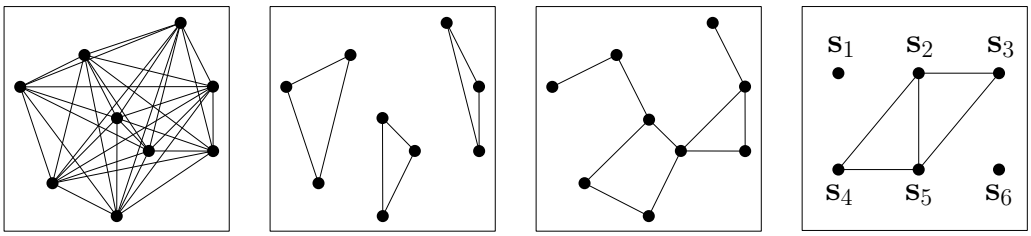
4.3 Privacy Definition

To derive the meaning of DPHMM, we extend Blowfish privacy from [28, 32]. Related privacy notions are also discussed in this section.

4.3.1 Policy Graph

Policy Graph without Constraint. We first study the problem in the whole domain $\mathcal{S}$ without any constraint. Given the true state $\mathbf{s}^*$ at a timestamp, a user may prefer to hide $\mathbf{s}^*$ in a group of candidate states, denoted by $\mathcal{N}(\mathbf{s}^*)$ as neighbors of $\mathbf{s}^*$ where $\mathcal{N}(\mathbf{s}^*) \subseteq \mathcal{S}$. Intuitively, the more neighbors a state has, the more privately it is protected. For simplicity, we assume $\mathbf{s}_i \in \mathcal{N}(\mathbf{s}_i)$ for all states $\mathbf{s}_i$ because it is straightforward that $\mathbf{s}_i$ is hidden in its neighbor set $\mathcal{N}(\mathbf{s}_i)$.

We can represent the privacy policy by a undirected graph where a node represents a state and an edge connects an indistinguishable pair of states. The graph has to be undirected for the following reason. When the true state is $\mathbf{s}_i$, we can draw edges from $\mathbf{s}_i$ to all $\mathbf{s}_j \in \mathcal{N}(\mathbf{s}_i)$. The edge $\overline{\mathbf{s}_i\mathbf{s}_j}$ means $\mathbf{s}_i$ and $\mathbf{s}_j$ are indistinguishable given true state $\mathbf{s}_i$. On the other hand, when true state is $\mathbf{s}_j$, the indistinguishability $\overline{\mathbf{s}_j\mathbf{s}_i}$ also has to hold. Otherwise an adversary may infer that $\mathbf{s}_i$ and $\mathbf{s}_j$ are only connected when $\mathbf{s}_i$ is the true state. Therefore, the edge $\overline{\mathbf{s}_i\mathbf{s}_j}$ is undirected, meaning it holds for both $\mathbf{s}_i$ and $\mathbf{s}_j$.



(a) G_{cplt} **(b)** G_{categ} **(c)** G_{util} **(d)** G_{trs}

Figure 4.4: Examples of policy graphs without constraint. (a): all states are connected; (b): states in a category are connected; (c): nearby states are connected; (d): transition protection for Example 4.1. To protect a one-step transition $\mathbf{s}_i \rightarrow \mathbf{s}_j$, we can require all pairs of states $\mathbf{s}_j$ and $\mathbf{s}_k$ to be indistinguishable if they can be transited from the same previous state $\mathbf{s}_i$. $G_{trs} := \{G|\overline{\mathbf{s}_j\mathbf{s}_k} \in \mathcal{E} \text{ iff } m_{ij} > 0 \text{ and } m_{ik} > 0, \forall i, j, k\}$. Note that with G_{trs} even if $\mathbf{s}_t^*$ were exposed, $\mathbf{s}_{t+1}^*$ would still be protected.

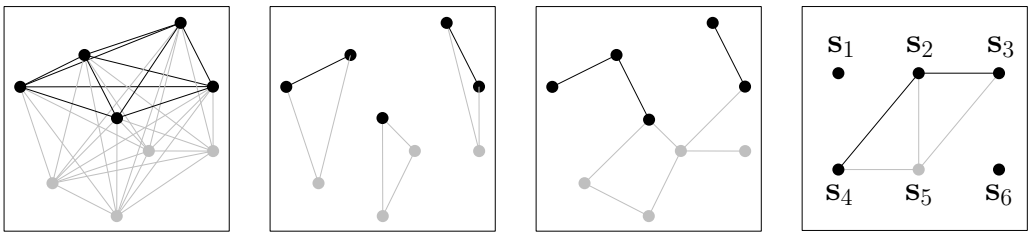
Definition 4.3 (Policy)**.** *A policy is an undirected graph $G = (\mathcal{S}, \mathcal{E})$ where $\mathcal{S}$ denotes all states (nodes) and $\mathcal{E}$ represents indistinguishability (edges) between states.*

Definition 4.4 (Neighbors)**.** *Let $\boldsymbol{s}$ be a state in $\mathcal{S}$. The neighbors of $\boldsymbol{s}$, denoted by $\mathcal{N}(\boldsymbol{s})$, is the set of nodes connected with $\boldsymbol{s}$ by an edge, including $\boldsymbol{s}$ itself.*

$$\mathcal{N}(\boldsymbol{s}) := \{\boldsymbol{s}\} \cup \{\boldsymbol{s}'|\overline{\boldsymbol{s}\boldsymbol{s}'} \in \mathcal{E}, \boldsymbol{s}' \in \mathcal{S}\}$$

To better adjust utility and privacy for any particular applications, how to design policy graph is not a trivial task. We briefly discuss some policy graphs in Figure 4.4. In DPHMM, we assume a policy graph is given.

Policy Graph with Constraint. With the constraint $\mathcal{C}_t$ (Definition 4.2), the policy graph G has to be built on $\mathcal{C}_t$ at each timestamp t. Then the policy graph becomes a subgraph with the nodes in $\mathcal{C}_t$ and the residual edges in G, denoted by constrained policy graph $G \cap \mathcal{C}_t$. It is intuitive that with different $\mathcal{C}_t$ graphs may be different over time.



(a) $G_{cplt} \cap \mathcal{C}_1$ **(b)** $G_{categ} \cap \mathcal{C}_1$ **(c)** $G_{util} \cap \mathcal{C}_1$ **(d)** $G_{trs} \cap \mathcal{C}_2$

Figure 4.5: Policy graphs of Figure 4.4 with constraints $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted by the black points in the graphs.

Example 4.3 (Constrained Policy Graph)**.** *Figure 4.5 shows the policy graphs of Figure 4.4 with the constraint sets. The black points indicate the constraint sets. The gray points and their edges are removed from the original graph.*

4.3.2 DPHMM

With policy graph G and any constraint $\mathcal{C}_t$, $\{\epsilon, G, \mathcal{C}_t\}$-DPHMM can be defined as follows with the intuition that at any timestamp the true state cannot be distinguished from its remaining "neighbors" under the constraint.

Definition 4.5 ($\{\epsilon, G, \mathcal{C}_t\}$-DPHMM)**.** *Let G be the policy graph, $\mathcal{C}_t$ be the constraint at timestamp t. An $\{\epsilon, G, \mathcal{C}_t\}$-DPHMM algorithm $\mathcal{A}()$ generates an output $\boldsymbol{z}_t$ such that for any $\boldsymbol{z}_t$ and any state $\boldsymbol{s}_j \in \mathcal{C}_t$, the following condition is satisfied:*

$$\begin{cases} \forall \boldsymbol{s}_k \in \mathcal{N}(\boldsymbol{s}_j) \cap \mathcal{C}_t, & \text{if } \boldsymbol{s}_j \text{ is connected to } \boldsymbol{s}_k \text{ in } G \cap \mathcal{C}_t; \\ \exists \boldsymbol{s}_k \in \mathcal{C}_t, & \text{if } \boldsymbol{s}_j \text{ is disconnected in } G \cap \mathcal{C}_t; \end{cases}$$

$$e^{-\epsilon} \leqslant \frac{Pr(\mathcal{A}(\boldsymbol{s}_j) = \boldsymbol{z}_t)}{Pr(\mathcal{A}(\boldsymbol{s}_k) = \boldsymbol{z}_t)} \leqslant e^{\epsilon} \tag{4.1}$$

In above definition, if $\mathbf{s}_j$ is connected with any $\mathbf{s}_k$ in $\mathcal{C}_t$, then $\mathbf{s}_j$ and $\mathbf{s}_k$ are

indistinguishable by Equation (4.1); However, if $\mathbf{s}_j$ is disconnected, $\mathbf{s}_j$ may be exposed [2]. To protect $\mathbf{s}_j$ in this case, we have to connect $\mathbf{s}_j$ to another node $\mathbf{s}_k$ in $\mathcal{C}_t$ (such new graph is called protectable graph in Section 4.4.2) to form a new edge of indistinguishability between $\mathbf{s}_j$ and $\mathbf{s}_k$. The user has the choice to specify which $\mathbf{s}_k$ to use to protect $\mathbf{s}_j$ [3]. We also discuss how to find the optimal $\mathbf{s}_k$ in Section 4.5.1.

4.3.3 Comparison with Other Definitions

Among the variant definitions of differential privacy [32, 36, 37, 62] we briefly compare some closely related definitions as follows.

δ-Location Set based Differential Privacy. Chapter 3 defined differential privacy on a subset of possible states (locations) derived from Markov model. The indistinguishability is ensured among any two locations in the δ-location set, which can be viewed as a new constraint. Thus it is a special case of DPHMM with complete graph.

Theorem 4.1. *δ-location set based ϵ-differential privacy is equivalent to $\{\epsilon, G_{cplt}, \mathcal{C}'_t\}$-DPHMM where G_{cplt} is a complete graph and $\mathcal{C}'_t = min\{\boldsymbol{s}_i | \sum_{\boldsymbol{s}_i} \boldsymbol{p}_t^-[i] \geqslant 1-\delta\}$.*

Blowfish Framework. There are three differences between DPHMM and Blowfish framework. (1) The constraints in Blowfish are deterministic; while constraints in Markov model are probabilistic. (2) The graph in Blowfish is static; while in Markov model the graph can be reduced. When there are disconnected nodes in the reduced graph, privacy risk needs to be tackled. (3)

[2] The exposure consists of two scenarios: (1) an adversary knows $\mathbf{s}_j$ is the true state. (2) an adversary knows $\mathbf{s}_j$ is not the true state.

[3] We do not hide the policy information (e.g. $\mathbf{s}_j$ and $\mathbf{s}_k$ are connected). DPHMM ensures that an adversary cannot distinguish whether $\mathbf{s}_j$ or $\mathbf{s}_k$ is the true state.

We quantify the privacy guarantee of Blowfish in terms of differential privacy (Section 4.4.1).

Theorem 4.2. *$\{\epsilon, G, \mathcal{C}_t\}$-DPHMM is equivalent to $\{\epsilon, \{\mathcal{S}, \mathbb{G}_t, \mathcal{C}_t\}\}$-Blowfish privacy where $\mathcal{S}$ is the domain of states in a Markov model, $\mathbb{G}_t$ is the set of graphs satisfying the condition in DPHMM and $\mathcal{C}_t$ is the constraint.*

4.4 Privacy Risk

Given a constrained policy graph $G \cap \mathcal{C}_t$, when a node $\mathbf{s}_i$ is disconnected (without neighbors), one may conclude that $\mathbf{s}_i$ will be disclosed. However, we show that this may not be the case. We first quantify the privacy risk given the static Blowfish policies. Then we formalize the privacy risk and define degree of protection (DoP) to measure the risk.

4.4.1 Blowfish Analysis

Let us continue with the example in in Figure 4.1a.

Example 4.4 (Laplace Exposure)**.** *Given the table $\mathcal{T}$ in Figure 4.1a and the graph in Figure 4.1b, let f be a two-dimensional query:*

$$f_1: \textit{ select count(*) from } \mathcal{T} \textit{ where disease=“cancer”}$$
$$f_2: \textit{ select count(*) from } \mathcal{T} \textit{ where disease=“diabetes”}$$

W.l.o.g, for a database D we assume the answer to the query is $f(D) = [10, 20]^\intercal$. Then $f(D \cup \boldsymbol{s}_1) = [11, 20]^\intercal$, $f(D \cup \boldsymbol{s}_2) = [10, 20]^\intercal$, $f(D \cup \boldsymbol{s}_3) = [11, 20]^\intercal$, $f(D \cup \boldsymbol{s}_4) = [10, 21]^\intercal$. $\Delta f = \pm\,[f(D \cup \boldsymbol{s}_1) - f(D \cup \boldsymbol{s}_2), f(D \cup \boldsymbol{s}_3) - f(D \cup \boldsymbol{s}_4)] =$

$\begin{bmatrix} 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}^\intercal$ *where each column is a data point of the differences between connected nodes. Then* ℓ_1*-norm sensitivity* $S_f = 2$*. Assume* $\{\epsilon, G\}$*-Blowfish privacy is preserved where* G *is the graph in Figure 4.1b. For* $\boldsymbol{s}_5$*,* $f(D \cup \boldsymbol{s}_5) = [11, 20]^\intercal$*.* $f(D \cup \boldsymbol{s}_5) - f(D) = [1, 0]^\intercal$*. Because* $||[1, 0]^\intercal||_1 = 1 < S_f$*, it is protected by Laplace mechanism. Similarly for* $\boldsymbol{s}_6$*,* $f(D \cup \boldsymbol{s}_6) - f(D) = [0, 1]^\intercal$*. Thus* $\boldsymbol{s}_6$ *is also protected by Laplace mechanism since* $||[0, 1]^\intercal||_1 = 1 < 2$*.*

Based on above example, can we claim $\mathbf{s}_5$ and $\mathbf{s}_6$ are always protected by Blowfish policy in Figure 4.1b? The answer is no because the real protection is provided by data release mechanisms. Next we analyze the risk in above example by K-norm mechanism.

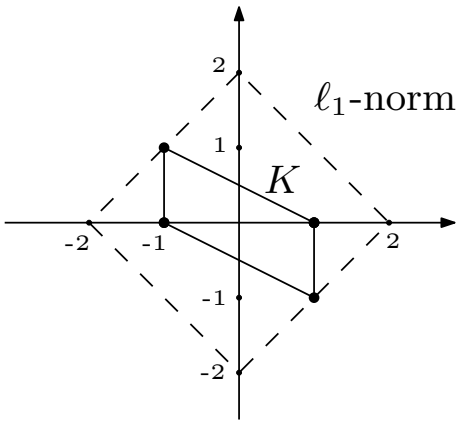


Figure 4.6: Sensitivity hull K in Example 4.5.

Example 4.5 (K-norm Exposure)**.** *We can derive the sensitivity hull* $K = Conv(\Delta f)$ *in Figure 4.6. Then secret* $\boldsymbol{s}_5$ *is protected with both Laplace mechanism and* K*-norm mechanism because* $f(D \cup \boldsymbol{s}_5) - f(D) = [1, 0]^\intercal \in K$*; while secret* $\boldsymbol{s}_6$ *is not by* K*-norm mechanism because* $f(D \cup \boldsymbol{s}_6) - f(D) = [0, 1]^\intercal \notin K$*. It can be proven that with* K*-norm mechanism the unbounded differential privacies for the existences of* $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{s}_5, \boldsymbol{s}_6\}$ *are* $\{\epsilon, 0, \epsilon, 2\epsilon, \epsilon, 2\epsilon\}$ *respectively*

Mechanisms	Risk Factor	Risk Level
Gaussian Mechanism [41]	ℓ_2-norm	small
Laplace Mechanism [16]	ℓ_1-norm	medium
K-norm Mechanism [29]	K-norm	large

Table 4.2: Privacy risk when using different Mechanisms

(e.g. $\frac{Pr(\mathcal{A}(D\cup \boldsymbol{s}_6)=z)}{Pr(\mathcal{A}(D)=z)} \leqslant e^{2\epsilon}$*). Thus it is* 2ϵ*-unbounded-DP in total. This also illustrates why we need to re-design a new optimal graph with less privacy loss in Section 4.5.1.*

Note that (1) the original bounded Blowfish privacy (i.e. the graph in Figure 4.1b) does not protect $\boldsymbol{s}_6$ *in the first place. Hence the original Blowfish privacy still holds; (2) although* $\boldsymbol{s}_4$ *is connected with* $\boldsymbol{s}_3$*, the existence of* $\boldsymbol{s}_4$ *is also at risk (*2ϵ*-DP); (3) although* $\boldsymbol{s}_6$ *is not connected with* $\boldsymbol{s}_4$*, it is indistinguishable with* $\boldsymbol{s}_4$ *by default.*

In summary, the privacy risk of Blowfish policies depends on the data release mechanisms, which have different sensitivity functions of Δf. It can be proven that with smaller sensitivity comes better utility but more risk with Blowfish policies. In Table 4.2 we summarize the privacy risks for common mechanisms. For Gaussian Mechanism [41], Laplace Mechanism [16] and K-norm Mechanism [29], the sensitivities are the ℓ_2-norm, ℓ_1-norm and K-norm of Δf. Because ℓ_2-norm $> \ell_1$-norm $\geqslant K$-norm, their privacy risks are small, medium and large relatively. Because of the higher risk, we use K-norm mechanism to quantify the privacy risk in the rest of this chapter.

Quantifying Blowfish. We quantify the overall protection of Blowfish privacy. First, it is intuitive that bounded Blowfish is weaker than (or equal to) bounded DP, and unbounded Blowfish is also weaker than (or equal to) unbounded DP. For lack of space, below we quantify the protection of bounded

Blowfish in terms of unbounded DP. It can be easily extended to other cases.

Definition 4.6 ($\{\epsilon, G, \mathcal{C}\}$-*Constrained*DP)**.** *Let $G = (\mathcal{S}, \mathcal{E})$ be the policy graph in Blowfish privacy, and $\mathcal{C}$ be the instances satisfying the constraint in either Markov model or database context. A randomized mechanism $\mathcal{A}()$ satisfies $\{\epsilon, G, \mathcal{C}\}$-constrained differential privacy if for any output $\boldsymbol{z}$, one of the following condition holds:*

(1). in Markov model, $\frac{Pr(\mathcal{A}(\boldsymbol{s}_j)=\boldsymbol{z})}{Pr(\mathcal{A}(\boldsymbol{s}_k)=\boldsymbol{z})} \leqslant e^{\epsilon}, \forall \boldsymbol{s}_j, \boldsymbol{s}_k \in \mathcal{C}$;
(2). in database context, $\frac{Pr(\mathcal{A}(D_1)=\boldsymbol{z})}{Pr(\mathcal{A}(D_2)=\boldsymbol{z})} \leqslant e^{\epsilon}, \forall D_1, D_2, \in \mathcal{C}, D_1$ can be obtained by adding $\boldsymbol{s}_i$ to or removing $\boldsymbol{s}_i$ from D_2, $\boldsymbol{s}_i \in \mathcal{S}$.

Note that in Markov model, above definition is actually bounded DP because unbound DP becomes bounded DP by nature (Section 4.2.1).

Theorem 4.3 (Blowfish Protection)**.** *Let G be the policy graph, and $\mathcal{C}$ be the instances satisfying the constraint in Blowfish privacy. With K-norm mechanism, if $\{\epsilon, \{\mathcal{S}, G, \mathcal{C}\}\}$-Blowfish privacy holds, then it satisfies*
(1). $\left\{\left(\max_{\forall \boldsymbol{s}_j, \boldsymbol{s}_k \in \mathcal{C}} ||f(\boldsymbol{s}_j) - f(\boldsymbol{s}_k)||_K\right)\epsilon, G, \mathcal{C}\right\}$-constrainedDP in Markov model;
(2). $\left\{\left(\max_{\forall \boldsymbol{s}_i \in \mathcal{C}} ||f(\boldsymbol{s}_i)||_K\right)\epsilon, G, \mathcal{C}\right\}$-constrainedDP in database context, where K is the sensitivity hull of query f.

Discussion. As shown in Chapter 3, ℓ_1-norm sensitivity is bigger than sensitivity hull. Following this, we can further prove that Laplace mechanism is a special case of K-norm mechanism and provides no better utility than K-norm mechanism. Thus we use K-norm mechanism as a unifying mechanism in the following analysis of this paper.

For standard Laplace mechanism, the answer for query workload $\mathbf{F} \in \mathbb{R}^{d \times N}$

[16] is

$$\mathbf{z} = \mathbf{F}\mathbf{x}^* + \frac{\Delta\mathbf{F}}{\epsilon}\tilde{\mathbf{n}}$$

where $\Delta\mathbf{F}$ is the ℓ_1 norm sensitivity of $\mathbf{F}$ and $\tilde{\mathbf{n}} \in \mathbb{R}^d$ are i.i.d variables from standard Laplace distribution with mean 0 and variance 1.

Lemma 4.1. *Let $f_{\tilde{\boldsymbol{n}}}(\tilde{\boldsymbol{n}})$ be the joint distribution of $\tilde{\boldsymbol{n}} \in \mathbb{R}^d$ where $\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_d$ are from i.i.d standard Laplace distribution. Then*

$$f_{\tilde{\boldsymbol{n}}}(\tilde{\boldsymbol{n}}) = \frac{1}{2^d} exp\left(-||\tilde{\boldsymbol{n}}||_1\right)$$

Proof. For a scalar variable $\tilde{n}$ from standard Laplace distribution, $f_{\tilde{n}}(\tilde{n}) = \frac{1}{2}exp(-|\tilde{n}|)$. Then for $\tilde{\mathbf{n}} = [\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_d]^T$,

$$\begin{aligned}
&f_{\tilde{\mathbf{n}}}(\tilde{\mathbf{n}} = [\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_d]^T) \\
&= \frac{1}{2}exp(-|\tilde{n}_1|) \cdot \frac{1}{2}exp(-|\tilde{n}_2|) \cdot \cdots \cdot \frac{1}{2}exp(-|\tilde{n}_d|) \\
&= \frac{1}{2^d} exp\left(-||\tilde{\mathbf{n}}||_1\right)
\end{aligned}$$

□

Theorem 4.4. *Let $f_{\boldsymbol{z}}(\boldsymbol{z})$ be the probability distribution of $\boldsymbol{z}$ from standard Laplace mechanism. Then*

$$f_{\boldsymbol{z}}(\boldsymbol{z}) = \frac{\epsilon^d}{2^d \Delta \boldsymbol{F}^d} exp\left(-\frac{\epsilon}{\Delta \boldsymbol{F}} ||\boldsymbol{z} - \boldsymbol{F}\boldsymbol{x}^*||_1\right)$$

Theorem 4.5. *Laplace mechanism is a special case of K-norm mechanism when $K = K_{S_f}^{\Diamond}$ where $K_{S_f}^{\Diamond}$ is the cross polytope $\{\boldsymbol{x} \in \mathbb{R}^d : ||\boldsymbol{x}||_1 \leqslant S_f\}$ and S_f is the ℓ_1-norm sensitivity of f.*

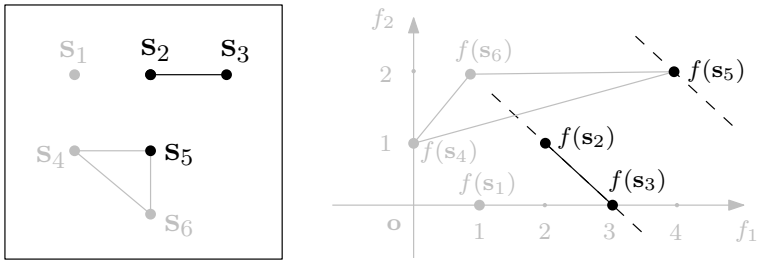


Figure 4.7: (left) constrained policy graph; (right) the query results in measurement space.

Proof. In Equation (3.5), let $K = K^{\Diamond}_{\Delta\mathbf{F}}$. Then $\textsc{Vol}(K^{\Diamond}_{\Delta\mathbf{F}}) = \frac{2^d}{\Gamma(d+1)}\Delta\mathbf{F}^d$. The $K^{\Diamond}_{\Delta\mathbf{F}}$-norm of any $\mathbf{z} \in \mathbb{R}^d$ is $\frac{||\mathbf{z}||_1}{\Delta\mathbf{F}}$. Then we can obtain

$$Pr(\mathbf{z}) = \frac{\epsilon^d}{2^d\Delta\mathbf{F}^d} exp(-\frac{\epsilon}{\Delta\mathbf{F}}||\mathbf{z} - \mathbf{F}\mathbf{x}^*||_1)$$

□

Corollary 4.1. *Laplace mechanism provides no better utility than K-norm mechanism because $K^{\Diamond}_{S_f}$ always contains sensitivity hull K.*

4.4.2 Degree of Protection

Under constraint, connectivity of policy graph is destructed. Thus the old graph evolves to the new reduced graph. In the following example, we show that with the new graph existing data release methods leads to exposure of disconnected nodes.

Example 4.6 (Information Exposure)**.** *Given the query in Example 4.2 and the graph in Figure 4.3b, assume at a timestamp t, the constraint set $\mathcal{C}_t = \{\boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_5\}$. Then the constrained graph is shown in Figure 4.7 (left). We use K-norm mechanism to answer the query in Example 4.2. Sensitivity hull of the query f is $Conv(f(\boldsymbol{s}_2) - f(\boldsymbol{s}_3), f(\boldsymbol{s}_3) - f(\boldsymbol{s}_2))$. For a K-norm based*

mechanism $\mathcal{A}()$, it means $\mathcal{A}(\boldsymbol{s}_2)$ and $\mathcal{A}(\boldsymbol{s}_3)$ are on the line of $\overline{f(\boldsymbol{s}_2)f(\boldsymbol{s}_3)}$ (dashed line through $f(\boldsymbol{s}_2)$ and $f(\boldsymbol{s}_3)$ in Figure 4.7 (right)); $\mathcal{A}(\boldsymbol{s}_5)$ is on the dashed line through $f(\boldsymbol{s}_5)$ in Figure 4.7 (right). We assume the released result $\boldsymbol{z} = f(\boldsymbol{s}_5)$. Then $Pr(\mathcal{A}(\boldsymbol{s}_3) = f(\boldsymbol{s}_5)) = 0$. Clearly, any $\boldsymbol{z}$ not on the line of $\overline{f(\boldsymbol{s}_2)f(\boldsymbol{s}_3)}$ leads to complete exposure of $\boldsymbol{s}_5$.

Intuitively, given a disconnected node $\mathbf{s}_i$ in the constrained set $\mathcal{C}_t$, if there exists another node $\mathbf{s}_j \in \mathcal{C}_t$ such that the difference $f(\mathbf{s}_i) - f(\mathbf{s}_j)$ is contained in the sensitivity hull K, then $\mathbf{s}_i$ is protected by $\mathbf{s}_j$. Otherwise, if no such node $\mathbf{s}_j$ exists, then $\mathbf{s}_i$ is exposed. Therefore, privacy risk can be measured by sensitivity hull K as follows. If there is no $\mathbf{s}_j$ such that $f(\mathbf{s}_j) \in f(\mathbf{s}_i) + K$, then $\mathbf{s}_i$ is exposed, meaning that Equation (4.1) will not hold. To capture such geometric meaning (i.e., $f(\mathbf{s}_j) - f(\mathbf{s}_i) \in K$), we define degree of protection.

Definition 4.7 (DoP)**.** *At any timestamp t, the degree of protection (*DoP*) of a state $\boldsymbol{s}_i$ is the number of states contained in $f(\boldsymbol{s}_i) + K_t$ where K_t is the sensitivity hull.*

$$\text{DoP}(\boldsymbol{s}_i, K_t) = |\{\boldsymbol{s}_j | f(\boldsymbol{s}_j) \in f(\boldsymbol{s}_i) + K_t, \boldsymbol{s}_j \in \mathcal{C}_t\}|$$

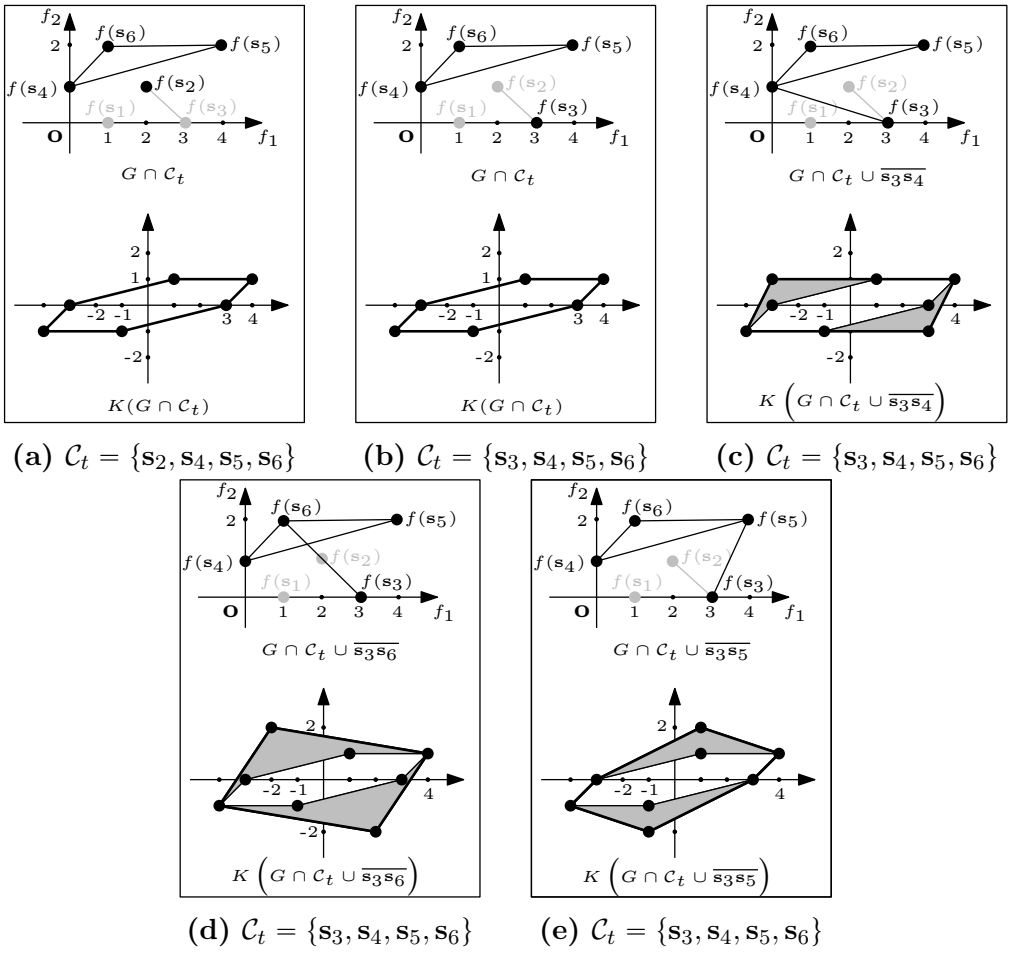
Because $f(\mathbf{s}_i)$ is always in $f(\mathbf{s}_i) + K$, $\text{DoP}(\mathbf{s}_i, K) \geqslant 1$ for all $\mathbf{s}_i \in \mathcal{C}_t$. Note that not all disconnected nodes are exposed. For example, in Figure 4.8a, $\mathbf{s}_2$ is disconnected under constraint $\mathcal{C}_t = \{\mathbf{s}_2, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$. However, $\text{DoP}(\mathbf{s}_2) = 3$ since $f(\mathbf{s}_2) + K$ contains $f(\mathbf{s}_4)$ and $f(\mathbf{s}_5)$.

If all the nodes of a graph have $\text{DoP} > 1$, we say it is protectable. Note that a complete graph is always protectable because every two nodes are connected.

Definition 4.8 (Protectable Graph)**.** *A graph $\mathcal{G}$ is protectable if all its nodes have* $\text{DoP} > 1$.

Theorem 4.6 (Exposure Condition)**.** *With Laplace mechanism or K-norm based mechanisms, a graph $\mathcal{G}$ cannot satisfy the DPHMM condition (Definition 4.5) iff $\mathcal{G}$ is not protectable.*

Computation. The computation of protectability (i.e. DoP) is to check the number of $f(\mathbf{s}_j)$ inside a convex body $f(\mathbf{s}_i) + K$ for all $\mathbf{s}_j \in \mathcal{C}_t$. Because the problem of checking whether a point is a convex body has been well studied in computational geometry, we skip the discussion of details.



(a) $\mathcal{C}_t = \{\mathbf{s}_2, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ **(b)** $\mathcal{C}_t = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ **(c)** $\mathcal{C}_t = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$

(d) $\mathcal{C}_t = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$ **(e)** $\mathcal{C}_t = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$

Figure 4.8: (a): if $\mathcal{C}_t = \{\mathbf{s}_2, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$, then $\mathbf{s}_2$ is also protected because $f(\mathbf{s}_4) \in f(\mathbf{s}_2) + K$ and $f(\mathbf{s}_5) \in f(\mathbf{s}_2) + K$; (b): if $\mathcal{C}_t = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$, then $\mathbf{s}_3$ is exposed; (c): adding $\overline{\mathbf{s}_3\mathbf{s}_4}$ to graph; (d): adding $\overline{\mathbf{s}_3\mathbf{s}_6}$ to graph; (e): adding $\overline{\mathbf{s}_3\mathbf{s}_5}$ to graph;

4.5 Data Release Mechanism

If privacy risk is detected, we build a protectable graph as a supergraph of existing graph [4]. In this section, we first formulate the problem of building a minimum protectable graph with lowest error bound. Next we show that this problem is #P-hard, and propose a fast greedy algorithm. Then we present the data release mechanism.

4.5.1 Minimum Protectable Graph

It is clear that a protectable graph satisfies the DPHMM condition in Definition 4.5. Therefore, when information is exposed, we need to build a protectable graph by re-connecting the disconnected nodes so that they have DoP > 1. Next we formulate the problem of building a minimum protectable graph and investigate its computational complexity, then propose a greedy algorithm to this end.

Minimum Protectable Graph. Because the error bound of differential privacy is determined by the volume of sensitivity hull $K(\mathcal{G}_t)$ [29] where $\mathcal{G}_t$ is the graph under constraint at timestamp t, the optimal graph should have the minimum volume of $K(\mathcal{G}_t)$ for best utility. We define the optimal graph as follows.

Given the policy graph G and the constraint set $\mathcal{C}_t$ at timestamp t, the optimal graph $\widehat{G}_t$ is a graph containing $G \cap \mathcal{C}_t$ with minimum volume $K(\widehat{G}_t)$

[4] On the other hand, since the policy graph is customizable to users, the protectable graph can also be created by users. In this case, it is not necessary to derive another minimum protectable graph again.

under the DPHMM condition (Definition 4.5):

$$\widehat{G}_t = \underset{\mathcal{G}}{argmin}\, \textsc{Vol}(K(\mathcal{G})) \tag{4.2}$$

$$\text{subject to: } (G \cap \mathcal{C}_t) \subseteq \widehat{G}_t, \text{and } \widehat{G}_t \text{ satisfies the DPHMM condition}$$

Example 4.7 (Minimum Protectable Graph)**.** *Given the query in Example 4.2 and the graph in Figure 4.3b, Figure 4.8b shows the graph under constraint* $\mathcal{C}_t = \{\boldsymbol{s}_3, \boldsymbol{s}_4, \boldsymbol{s}_5, \boldsymbol{s}_6\}$*. Then* $\boldsymbol{s}_3$ *is exposed because* $f(\boldsymbol{s}_3) + K$ *contains no other node. To satisfy the DPHMM condition, we need to connect* $\boldsymbol{s}_3$ *to another node in* $\mathcal{C}_t$*, i.e.* $\boldsymbol{s}_4$*,* $\boldsymbol{s}_5$ *or* $\boldsymbol{s}_6$*.*

If $\boldsymbol{s}_3$ *is connected to* $\boldsymbol{s}_4$*, then Figure 4.8c shows the new graph and its sensitivity hull. By adding two new edges* $\{f(\boldsymbol{s}_3) - f(\boldsymbol{s}_4), f(\boldsymbol{s}_4) - f(\boldsymbol{s}_3)\}$ *to* Δf*, the shaded areas are attached to the sensitivity hull. Similarly, Figures 4.8d and 4.8e show the new sensitivity hulls when* $\boldsymbol{s}_3$ *is connected to* $\boldsymbol{s}_6$ *and* $\boldsymbol{s}_5$ *respectively. Because the smallest* $\textsc{Area}(K)$ *is in Figure 4.8c, the optimal graph* $\widehat{G}_t$ *is* $G \cap \mathcal{C}_t \cup \overline{\boldsymbol{s}_3\boldsymbol{s}_4}$*.*

Complexity. We can see that to derive the optimal graph, minimum volume $\textsc{Vol}(K)$ should be computed. For any query $f : \mathcal{S} \to \mathbb{R}^d$, K is a polytope in $\mathbb{R}^d$. However, the volume computation of polytope is #P-hard [17]. Thus it follows that the computation of minimum volume is no easier than #P-hard.

Theorem 4.7. *The problem of minimum protectable graph in Equation (4.2) is #P-hard.*

Greedy Algorithm. Due to the computational complexity, we propose a greedy algorithm similar to minimum spanning tree. The idea is to connect each disconnected node to its nearest (in measurement space) node. For other

theoretical algorithms of volume computation with polynomial time bound, please see [59]. Algorithm 3 shows the greedy algorithm, which takes $O(N^2)$ time where $N = |\mathcal{V}|$ is the number of nodes.

Algorithm 3 Protectable Graph

Require: G, $\mathcal{C}_t$, f

1: $\mathcal{G}_t \leftarrow G \cap \mathcal{C}_t$;
2: **for all** exposed node $\mathbf{s}_i \in \mathcal{C}_t$ **do**
3: $\quad \mathbf{s}_j \leftarrow \underset{\mathbf{s} \in \mathcal{C}_t}{argmin} ||f(\mathbf{s}) - f(\mathbf{s}_i)||_2$;
4: $\quad \mathcal{G}_t \leftarrow \mathcal{G}_t \cup \overline{\mathbf{s}_i \mathbf{s}_j}$; ▷ connect to nearest node
5: **end for**
6: **return** protectable graph $\mathcal{G}_t$;

Discussion. It is possible to design fast algorithms in low dimensional space to derive the minimum protectable graph. We propose a fast algorithm in 2-dimensional space.

In 2-dimensional space, it only takes $O(mlog(m))$ time to find a convex hull where $m = |\mathcal{E}|$ is the number of edges. Thus we can connect the disconnected node $\mathbf{s}_i$ to the rest (at most $2m$) nodes, generating at most $2m$ convex hulls. We use $\sum_{i=1,j=i+1}^{i=h} det(\mathbf{v}_i, \mathbf{v}_j)$ to derive the area of a convex hull with clockwise nodes $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_h$ where h is the number of vertices and $\mathbf{v}_{h+1} = \mathbf{v}_1$. By comparing the area of these convex hulls, we can find the smallest area in $O(nm^3)$ time where n is the number of exposed nodes.

Theorem 4.8. *Algorithm 4 takes $O(nm^3)$ time where $m = |\mathcal{E}|$ is the number of edges and n is the number of exposed nodes.*

4.5.2 Data Release Mechanism

The data release mechanism is shown in Algorithm 5. At each timestamp t, we compute the prior probability vector $\mathbf{p}_t^-$. Under the constraint $\mathcal{C}_t$, the

Algorithm 4 2D Minimum Protectable Graph

Require: G, $\mathcal{C}_t$, $f : \mathcal{S} \to \mathbb{R}^2$

1: $\mathcal{G}_t(\mathcal{V}, \mathcal{E}) \leftarrow G \cap \mathcal{C}_t$;
2: $K \leftarrow K(\mathcal{G}_t)$;
3: **for all** exposed node $\mathbf{s}_i \in \mathcal{V}$ **do**
4: $\quad \mathbf{s}_k \leftarrow \varnothing$;
5: $\quad min\textsc{Area} \leftarrow \infty$;
6: $\quad$ **for all** other node $\mathbf{s}_j \in \mathcal{V}$ **do**
7: $\quad\quad K \leftarrow K(\mathcal{G}_t \cup \overline{\mathbf{s}_i\mathbf{s}_j})$; ▷ $O(m^2)$
8: $\quad\quad \textsc{Area} = \sum_{i=1,j=i+1}^{i=h} det(\mathbf{v}_i, \mathbf{v}_j)$ where $\mathbf{v}_{h+1} = \mathbf{v}_1$;
9: $\quad\quad$ **if** $\textsc{Area} < min\textsc{Area}$ **then**
10: $\quad\quad\quad \mathbf{s}_k \leftarrow \mathbf{s}_j$;
11: $\quad\quad\quad min\textsc{Area} = \textsc{Area}$; ▷ `find minimum area`
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad \mathcal{G}_t \leftarrow \mathcal{G}_t \cup \overline{\mathbf{s}_i\mathbf{s}_k}$
15: $\quad K \leftarrow K(\mathcal{G}_t)$;
16: **end for**
17: **return** graph $\mathcal{G}_t(\mathcal{V}, \mathcal{E})$;

graph G becomes a subgraph $G \cap \mathcal{C}_t$. To satisfy the DPHMM condition, we derive a protectable graph $\mathcal{G}_t$ by Algorithm 3. Next a differentially private mechanism can be adopted to release a perturbed answer $\mathbf{z}_t$. Then the released $\mathbf{z}_t$ will also be used to update the posterior probability $\mathbf{p}_t^+$ (in the equation below) by Equation (3.1), which subsequently will be used to compute the prior probability for the next timestamp $t + 1$.

$$\mathbf{p}_t^+[i] = Pr(\mathbf{s}_t^* = \mathbf{s}_i | \mathbf{z}_t, \mathbf{z}_{t-1}, \cdots, \mathbf{z}_1)$$

Note that in line 4 $\mathbf{z}_t$ can be released by any differentially private mechanisms.

Theorem 4.9. *Given policy graph G and query f, Algorithm 5 satisfies $\{\epsilon, G, \mathcal{C}_t\}$-DPHMM at any timestamp.*

Algorithm 5 Customizable Data Release Mechanism

Require: ϵ_t, G, f, $\mathbf{M}$, $\mathbf{p}_{t-1}^+$, $\mathbf{s}_t^*$

1: $\mathbf{p}_t^- \leftarrow \mathbf{p}_{t-1}^+ \mathbf{M}$; ▷ Markov transition
2: $\mathcal{C}_t \leftarrow \{\mathbf{s}_i | \mathbf{p}_t^-[i] > 0\}$; ▷ constraint
3: $\mathcal{G}_t \leftarrow$ ALGORITHM 3(G, $\mathcal{C}_t$, f); ▷ protectable graph $\mathcal{G}_t$
4: $\mathbf{z}_t \leftarrow K$-norm based mechanism($f(\mathbf{s}_t^*)$, $K(\mathcal{G}_t)$);
5: Derive $\mathbf{p}_t^+$ by Equation (3.1); ▷ inference
6: **return** ALGORITHM 5(ϵ_{t+1}, G, f, $\mathbf{M}$, $\mathbf{p}_t^+$, $\mathbf{s}_{t+1}^*$); ▷ go to next timestamp

Theorem 4.10. *Given policy graph G and query f, at any timestamp t, Algorithm 5 satisfies* $\left\{ \left(\max_{\forall \boldsymbol{s}_j, \boldsymbol{s}_k \in \mathcal{C}_t} ||f(\boldsymbol{s}_j) - f(\boldsymbol{s}_k)||_{K_t} \right) \epsilon, G, \mathcal{C}_t \right\}$ *-constrainedDP (Definition 4.6) where $\mathcal{C}_t$ is the constraint, K_t is the sensitivity hull of query f and protectable graph $\mathcal{G}_t$.*

4.5.3 Adversarial Knowledge

There might be a variety of adversaries with different prior knowledge in reality. Thus we consider the adversarial knowledge in this section. Similar to existing works [55, 62], we assume that the Markov model and the data release mechanism, including the sensitivity hull K, is transparent to any adversaries, meaning adversaries know how the query answers were released. If this assumption does not hold, then adversarial knowledge can only be worse, leading to less privacy disclosures.

We define constrained adversarial privacy as follows, with a similar adversary-constraint $\mathcal{C}_t^{\mathcal{A}}$ derived from the prior knowledge $\mathbf{p}_t^{\mathcal{A}}$ of any adversaries:

$$\mathcal{C}_t^{\mathcal{A}} := \{\mathbf{s}_i | \mathbf{p}_t^{\mathcal{A}}[i] > 0, \forall \mathbf{s}_i \in \mathcal{S}\}$$

Definition 4.9 ($\{\epsilon, \mathcal{C}_t^{\mathcal{A}}\}$-*Constrained*AP)**.** *For adversaries with knowledge $\mathcal{C}_t^{\mathcal{A}}$, a mechanism is ϵ-adversarially private if for any output $\boldsymbol{z}_t$ and any state $\boldsymbol{s}_i \in$*

$\mathcal{C}_t^{\mathcal{A}}$, $\frac{Pr(\boldsymbol{s}_i|\boldsymbol{z}_t)}{Pr(\boldsymbol{s}_i)} \leqslant e^{\epsilon}$.

We discuss various adversarial knowledge as follows.

- Case I: $\mathcal{C}_t^{\mathcal{A}} \subset \mathcal{C}_t$. When $|\mathcal{C}_t^{\mathcal{A}}| = 1$, the adversary has already known the true state. Then no privacy can be protected in this case. Otherwise, $G \cap \mathcal{C}_t^{\mathcal{A}}$ is a subgraph of $G \cap \mathcal{C}_t$. Then $\{\epsilon, G, \mathcal{C}_t^{\mathcal{A}}\}$-DPHMM still holds.

- Case II: $\mathcal{C}_t \subset \mathcal{C}_t^{\mathcal{A}}$. Similar to the analysis in Section 4.4.2, Algorithm 5 (using K-norm based mechanism) may not satisfy $\{\epsilon, G, \mathcal{C}_t^{\mathcal{A}}\}$-DPHMM if any node in $\mathcal{C}_t^{\mathcal{A}}$ has DoP = 1, derived from $K_t(\mathcal{G}_t)$ in Algorithm 5.

- $\left\{\left(\max_{\forall \mathbf{s}_i, \mathbf{s}_j \in \mathcal{C}_t^{\mathcal{A}} \cap \mathcal{C}_t} ||f(\mathbf{s}_i) - f(\mathbf{s}_j)||_{K_t}\right) \epsilon, \mathcal{C}_t^{\mathcal{A}} \cap \mathcal{C}_t\right\}$-*cons-trained*AP holds in both cases. Hence $\left\{\left(\max_{\forall \mathbf{s}_i, \mathbf{s}_j \in \mathcal{C}_t^{\mathcal{A}} \cap \mathcal{C}_t} ||f(\mathbf{s}_i) - f(\mathbf{s}_j)||_{K_t}\right) \epsilon, \mathcal{C}_t^{\mathcal{A}} \cap \mathcal{C}_t\right\}$-*constrained*DP also holds.

4.6 Privacy Composition

In some cases, multiple queries need to be answered. Thus we analyze the privacy composition for multiple data releases. Note that the parallel composition [46] is not applicable because there is only one state in Markov model. We refer readers to [7] for the detailed study of privacy quantification.

Single-Time Multiple-Queries. At one timestamp, it is possible that many queries should be answered. Then the privacy cost ϵ composes for all queries.

Theorem 4.11. *At timestamp t, an $\{\epsilon, G, \mathcal{C}_t\}$-DPHMM mechanism released multiple answers $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_n$ for queries $f_1, f_2, \cdots, f_n$ with $\epsilon_1, \epsilon_2, \cdots, \epsilon_n$, then it satisfies $\{\sum_{i=1}^{n} \epsilon_i, G, \mathcal{C}_t\}$-DPHMM and $\left\{\sum_{i=1}^{n}\left(\max_{\forall \boldsymbol{s}_j, \boldsymbol{s}_k \in \mathcal{C}_t} ||f(\boldsymbol{s}_j) - f(\boldsymbol{s}_k)||_{K_i}\right) \epsilon_i, G, \mathcal{C}_t\right\}$-constrainedDP where K_i denotes the sensitivity hull of f_i.*

Multiple-Time Single-Query. If a query was answered over multiple timestamps, then the privacy protection has to be enforced on the sequence. Under the probabilistic constraint, we define differentially private sequence with all possible sequences.

Definition 4.10. *A constraint set of sequences* $\mathcal{Q} = \{\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_n\}$ *is a set of* n *possible sequences with* $Pr(\boldsymbol{Q}_i) > 0$ *for all* $\boldsymbol{Q}_i \in \mathcal{S}^t$, $i = 1, 2, \cdots, n$.

Definition 4.11 ($\{\epsilon, \mathcal{Q}\}$-*Constrained*DPS)**.** *During timestamps* $1, 2, \cdots, t$ *in an HMM, a randomized mechanism* $\mathcal{A}()$ *generates* $\{\epsilon, \mathcal{Q}\}$-*ConstrainedDPS if for any output sequence* $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t$ *and any possible sequences* $\boldsymbol{Q}_j$ *and* $\boldsymbol{Q}_k$ *in* $\mathcal{Q}$*, the following holds*

$$\frac{Pr\left(\mathcal{A}(\boldsymbol{Q}_j) = (\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t)\right)}{Pr\left(\mathcal{A}(\boldsymbol{Q}_k) = (\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t)\right)} \leqslant e^{\epsilon}$$

Theorem 4.12. *During timestamps* $i = 1, 2, \cdots, t$ *in an* $\{\epsilon_i, G, \mathcal{C}_i\}$-*DPHMM with policy graph* G *and constraints* $\mathcal{C}_i = \{\boldsymbol{Q}_j[i] | \forall \boldsymbol{Q}_j \in \mathcal{Q}\}$*, the released sequence* $\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t$ *for a query* f *satisfies* $\left\{\sum_{i=1}^{t}\left(\max_{\forall \boldsymbol{s}_j, \boldsymbol{s}_k \in \mathcal{C}_i} ||f(\boldsymbol{s}_j) - f(\boldsymbol{s}_k)||_{K_i}\right)\epsilon_i, \mathcal{Q}\right\}$-*constrainedDPS where* K_i *denotes the sensitivity hull at timestamp* i.

Above compositions can be combined for the case of multiple-time and multiple-queries data releases. This completes our analysis of privacy composition over time.

4.7 Empirical Evaluation

We report the experimental evaluation in this section. All algorithms were implemented in Matlab on a PC with 2.4GHz CPU and 4GB memory.

Datasets. We used the two datasets, Geolife and Gowalla, with same configurations as Section 3.5. The Markov models were learned from the raw data. From each dataset, 20 sequences, each of which contains 100 timestamps, were selected for our experiment. Then the average result is reported.

Mechanisms. For better utility, we used the planar isotropic mechanism in Chapter 3 (with $\delta = 0.01$) to release the locations of users. We denote the notions by DPHMM and DPLS [5] respectively. The default value of ϵ is 1 if not mentioned.

Application. For location data, a common application is to release the location coordinates. Thus we use the measurement query $f : \mathcal{S} \to \mathbb{R}^2$ that returns a 2×1 vector of longitude and latitude.

Three policy graphs were adopted in our experiments: utility-oriented G_{util}, privacy-oriented G_{trs}, as defined in Section 4.3.1, and k-neighbors graph G_{knb}.

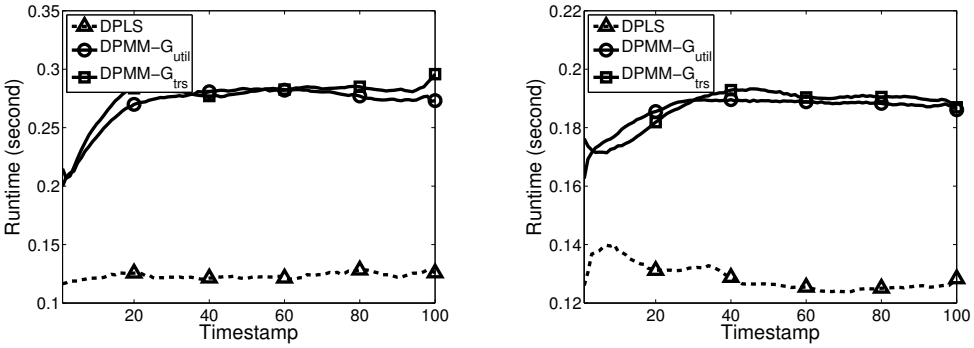
- G_{util} connects all nodes if their distances of locations are less than r;
- G_{trs} guarantees that even if the previous states were completely exposed, privacy can still be protected in the current timestamp.
- G_{knb} connects all nodes to k neighbors of nodes.

Because of different customizations of the three graphs, we can examine the different results of them. In G_{util}, the default values of r for GeoLife and Gowalla are $1(km)$ and $2(km)$ respectively. In G_{knb}, the default value of k is 9.

Metrics. We used the following metrics in our experiment.

- To measure the efficiency, the runtime of data release method was evaluated.

[5] Differential privacy on location set.



(a) Runtime on GeoLife **(b)** Runtime on Gowalla

Figure 4.9: Runtime.

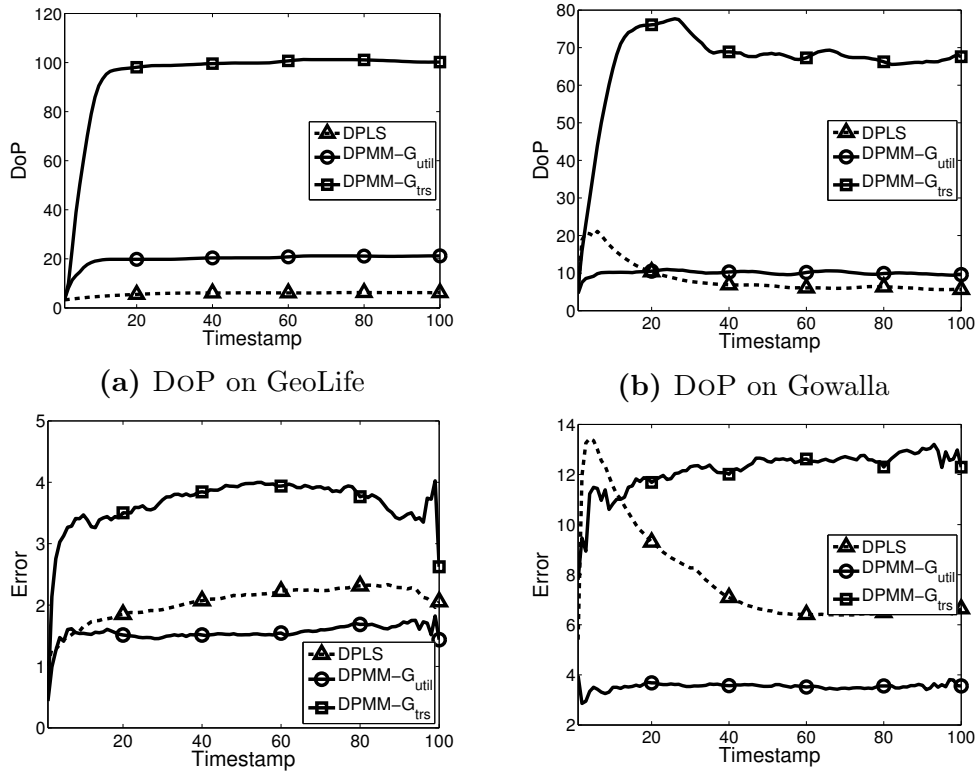
- DoP represents the number of nodes that a node is hidden in. Hence to reflect the privacy level of true states, DoP of true states was computed.
- The utility of DPHMM was measured by Error $= ||\mathbf{z}_t - f(\mathbf{s}_t^*)||_2$ where $\mathbf{z}_t$ is the released answer and $f(\mathbf{s}_t^*)$ is the true answer.

4.7.1 Runtime

Figure 4.9 shows the runtime report on the two datasets. We can see that the runtime of DPHMM is a little bit longer than DPLS. The reason is that DPLS uses a tighter constraint than $\mathcal{C}_t$, which in our setting became numerous when Markov model converged to a stationary distribution gradually. Then the computation of sensitivity hull took more time with larger graph. It is also worth noting that sensitivity hull converges with $\mathcal{C}_t$. As time evolves, the runtime also converges with Markov model to a stable level.

4.7.2 Performance over Time

At each timestamp, the (smoothed) DoP and Error are shown in Figure 4.10. As expected, G_{trs} provides the strongest protection of privacy, while G_{util} and



(a) DoP on GeoLife **(b)** DoP on Gowalla

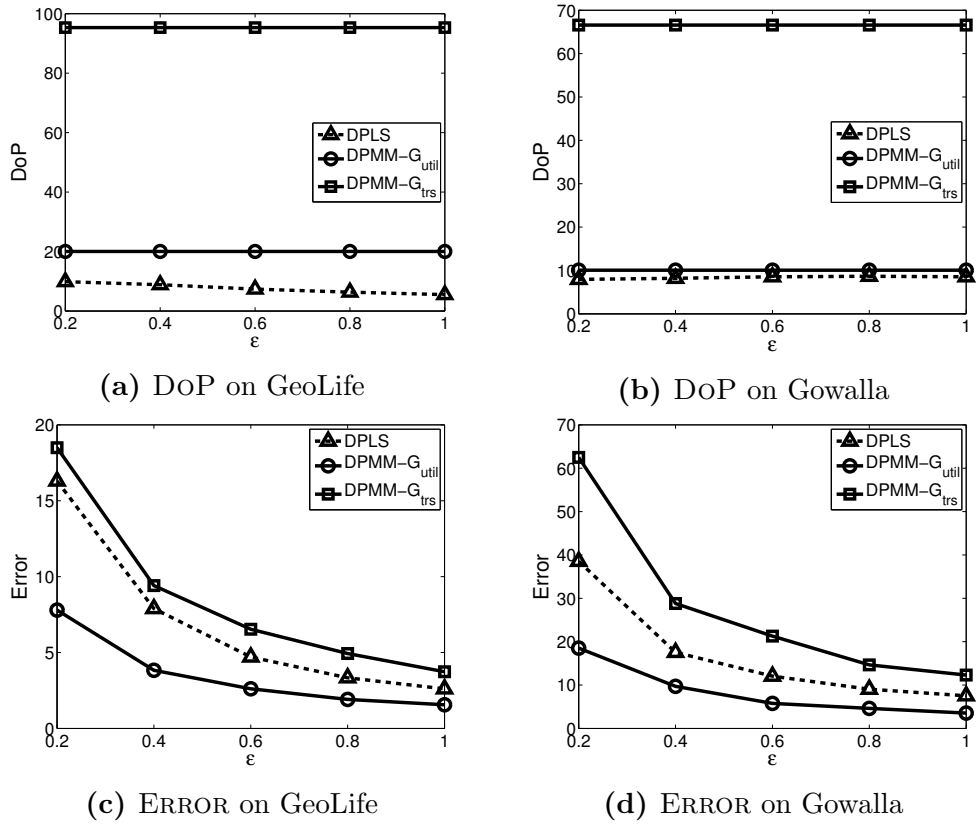
(c) Error on GeoLife **(d)** Error on Gowalla

Figure 4.10: Performance over time.

G_{knb} have the lower error on both datasets. With G_{trs}, the true state was protected in a set of 100 and 70 possible states for the two datasets. Provided such strong protection, the error also rises. With G_{util} and G_{knb}, the query error was smaller than DPLS yet the DoP was even larger than DPLS. Therefore, we can infer that customizable graph provides better trade-off between privacy and utility.

4.7.3 Impact of Privacy Budget ϵ

We also measure the average performance over the 100 timestamps with different parameter ϵ in Figure 4.11. In Figures 4.11a and 4.11b, DoP stays the same with different ϵ because the size of $\mathcal{C}_t$ does not change with ϵ. Again



(a) DoP on GeoLife **(b)** DoP on Gowalla

(c) Error on GeoLife **(d)** Error on Gowalla

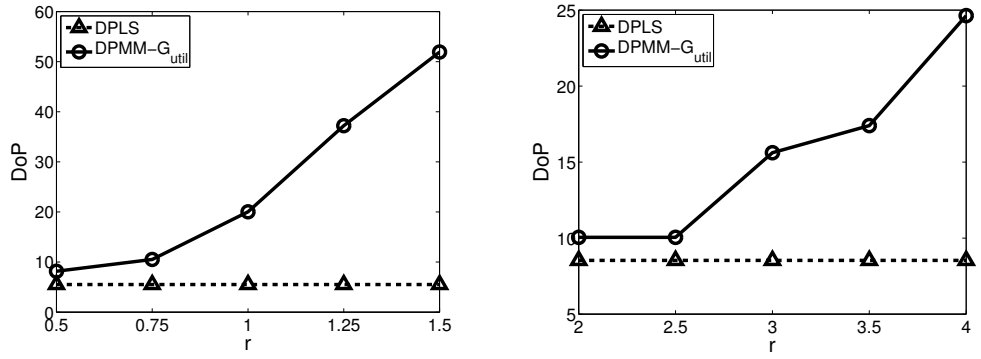
Figure 4.11: Impact of ϵ.

we see that G_{trs} provides the largest DoP with little sacrifice of utility, compared with DPLS. Figures 4.11c and 4.11d verifies that the larger ϵ, the smaller Error, which is easy to understand because ϵ determines the shape of noise distribution.

4.7.4 Tuning Privacy and Utility by Graphs

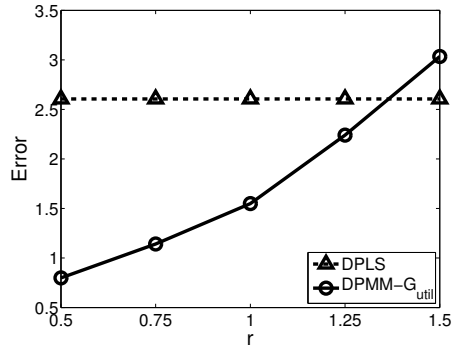
To better understand the trade-off between privacy and utility with different graphs, we also tested the performance with different $G_{util}(r)$ and $G_{knb}(k)$.

Different G_{util}. Intuitively, with larger r comes stronger protection and bigger Error, which is confirmed in Figures 4.12. However, the Error of DPHMM is still lower than DPLS in most results, although it is expected that Error

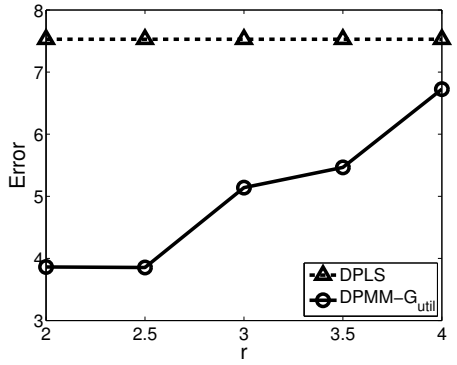


(a) DoP on GeoLife

(b) DoP on Gowalla



(c) Error on GeoLife



(d) Error on Gowalla

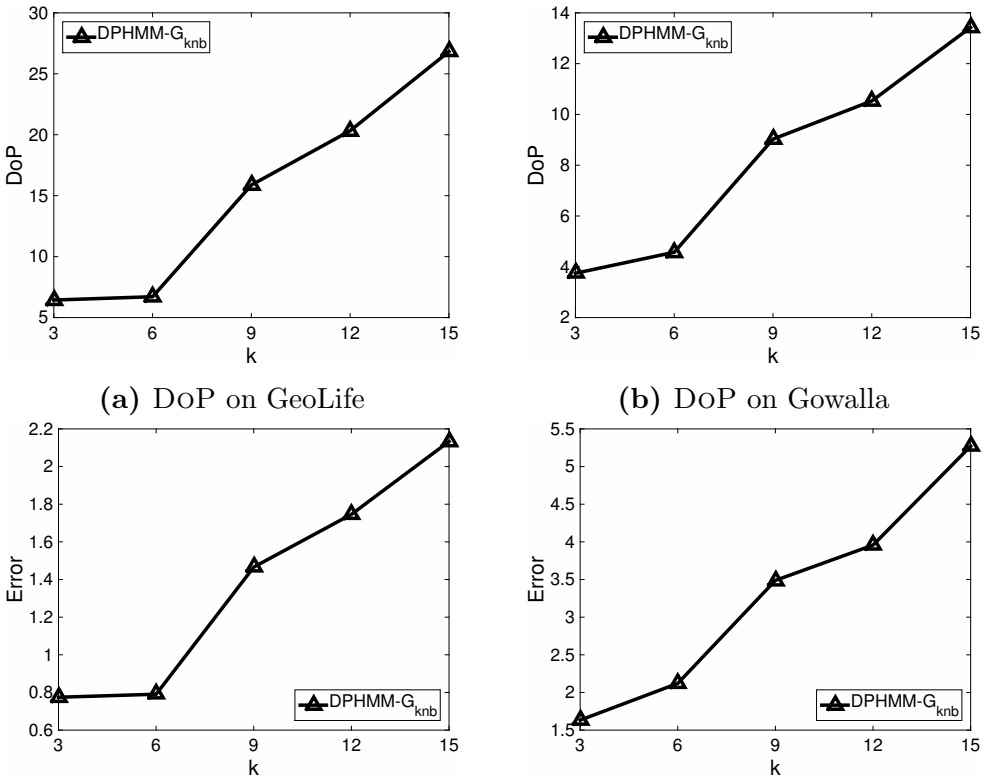
Figure 4.12: Tuning G_{util} with r.

grows with bigger r.

Different G_{knb}. Figures 4.13 show varying privacy and utility with G_{knb}. Similarly, both DoP and Error grow when k increases. Note that DoP is larger than k because some states are implicitly protected by the graph, as explained before. Therefore, we can conclude that with different policy graphs privacy and utility can be better tuned in different scenarios.

4.8 Conclusion Remarks

In this chapter we proposed DPHMM, a rigorous and customizable privacy framework to protect the true states in Markov model while allowing privacy-



(a) DoP on GeoLife **(b)** DoP on Gowalla

(c) Error on GeoLife **(d)** Error on Gowalla

Figure 4.13: Tuning G_{knb} with k.

utilty tradeoffs in the released data, by embedding a differentially private data release mechanism in hidden Markov model. DPHMM guarantees that the true state in Markov model at every timestamp is protected by a customizable policy graph. Under the temporal correlations, the graph may be reduced to subgraphs. Thus we studied the consequential privacy risk by introducing the notion of protectable graph based on the degree of protection. To prevent information exposure we studied how to build an optimal protectable graph based on the current graph. The privacy guarantee of DPHMM has also been thoroughly investigated, by comparing it with other privacy notions and studying the composition results over multiple queries and timestamps.

DPHMM can be used in a variety of applications to release private data

for purposes like data mining or social studies. Future works can also study how to efficiently design and implement the policy graph for various privacy requirements.

Chapter 5

Conclusion

In this dissertation, we investigated how to protect the locations of individual movement under temporal correlations. First, we proposed a notion of differential privacy based on δ-location set to protect the true location at each timestamp. We showed that the traditional ℓ_1-norm sensitivity in differential privacy exaggerates the real sensitivity, and defined the real sensitivity as sensitivity hull. Then we used the sensitivity hull in the PIM to release the noisy location at each timestamp. We proved that the PIM achieves the lower bound of error, thus it is an optimal data release mechanism. Second, we studied how to design the customizable privacy notion for different scenarios. We demonstrated that the privacy demand of individuals can be represented as policy graphs. However, under temporal correlations the graphs can be reduced to simpler graphs with disconnected nodes. To enforced privacy, we defined DPHMM, and proposed mechanisms to guarantee the privacy of DPHMM by re-generating the protectable graphs based on the DoP. Future works can further study how to design the policy graphs for various privacy demands.

Bibliography

[1] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22. ACM, 2010.

[2] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. CCS '13, pages 901–914. ACM, 2013.

[3] G. Apostolos. Notes on isotropic convex bodies. 2003.

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

[5] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Proceedings of the Second VDLB international conference on Secure Data Management*, SDM'05, pages 185–199, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] A. Bhaskara, D. Dadush, R. Krishnaswamy, and K. Talwar. Unconditional differentially private mechanisms for linear queries. STOC '12, New York, NY, USA, 2012.

[7] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy under temporal correlations. *ICDE*, 2017.

[8] K. Chatzikokolakis, M. Andrés, N. Bordenabe, and C. Palamidessi. Broadening the scope of differential privacy using metrics. Lecture Notes in Computer Science, pages 82–102. Springer Berlin Heidelberg, 2013.

[9] K. Chatzikokolakis, C. Palamidessi, and M. Stronati. A predictive differentially-private mechanism for mobility traces. In *PETS*, pages 21–41. Springer, 2014.

[10] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: a case study on the montreal transportation system. KDD '12, pages 213–221, New York, NY, USA, 2012. ACM.

[11] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. KDD '11, pages 1082–1090, New York, NY, USA, 2011.

[12] Y. De Mulder, G. Danezis, L. Batina, and B. Preneel. Identification via location-profiling in gsm networks. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, pages 23–32, New York, NY, USA, 2008. ACM.

[13] A. Dey, J. Hightower, E. de Lara, and N. Davies. Location-based services. *Pervasive Computing, IEEE*, 9(1):11–12, 2010.

[14] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[15] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[16] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. Theory of Cryptography Conference, 2006.

[17] M. E. Dyer and A. M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, Oct. 1988.

[18] T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Zufle. Querying uncertain spatio-temporal data. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 354–365, 2012.

[19] L. Fan, L. Bonomi, L. Xiong, and V. Sunderam. Monitoring web browsing behavior with differential privacy. WWW '14, New York, NY, USA, 2014.

[20] L. Fan, L. Xiong, and V. S. Sunderam. Differentially private multi-dimensional time series release for traffic monitoring. In *DBSec*, pages 33–48, 2013.

[21] C. Fang and E.-C. Chang. Differential privacy with δ-neighbourhood for spatial and dynamic datasets. ASIA CCS '14, pages 159–170, New York, NY, USA, 2014. ACM.

[22] K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. CCS '14, pages 239–250, New York, NY, USA, 2014. ACM.

[23] B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, 7(1):1–18, 2008.

[24] G. Ghinita. *Privacy for Location-Based Services*. Synthesis Lectures on Information Security, Privacy, and Tru. Morgan & Claypool, 2013.

[25] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132. ACM, 2008.

[26] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[27] M. Götz, S. Nath, and J. Gehrke. Maskit: Privately releasing user context streams for personalized mobile applications. SIGMOD '12, New York, NY, USA, 2012.

[28] S. Haney, A. Machanavajjhala, and B. Ding. Design of policy-aware differentially private algorithms. *Proc. VLDB Endow.*, 9(4):264–275, Dec. 2015.

[29] M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714. ACM, 2010.

[30] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. Principled evaluation of differentially private algorithms using dpbench. SIGMOD '16, pages 139–154.

[31] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava. Dpt: Differentially private trajectory synthesis using hierarchical reference systems. *Proc. VLDB Endow.*, 8(11):1154–1165, July 2015.

[32] X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: Tuning privacy-utility trade-offs using policies. SIGMOD '14, pages 1447–1458, New York, NY, USA, 2014. ACM.

[33] I. A. Junglas and R. T. Watson. Location-based services. *Communications of the ACM*, 51(3):65–69, 2008.

[34] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.

[35] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proc. VLDB Endow.*, 7(12):1155–1166, Aug. 2014.

[36] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. SIGMOD '11, New York, NY, USA, 2011.

[37] D. Kifer and A. Machanavajjhala. A rigorous and customizable framework for privacy. PODS '12, pages 77–88, New York, NY, USA, 2012. ACM.

[38] J. Krumm. Inference attacks on location tracks. PERVASIVE'07, pages 127–143, Berlin, Heidelberg, 2007. Springer-Verlag.

[39] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[40] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, New York, NY, USA, 2010.

[41] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *Proc. VLDB Endow.*, 5(6):514–525, Feb. 2012.

[42] H. Li, L. Xiong, and X. Jiang. Differentially private synthesization of multi-dimensional data using copula functions. EDBT'14, pages 475–486, 2014.

[43] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, Apr. 2007.

[44] L. Lovász and S. Vempala. Hit-and-run from a corner. STOC '04, pages 310–314, New York, NY, USA, 2004. ACM.

[45] L. Lovász and S. Vempala. Simulated annealing in convex bodies and an o*(n4) volume algorithm. *J. Comput. Syst. Sci.*, 72(2):392–417, Mar. 2006.

[46] McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD '09*, pages 19–30, New York, NY, USA, 2009. ACM.

[47] F. McSherry and K. Talwar. Mechanism design via differential privacy. FOCS '07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[48] V. Milman and A. Pajor. Isotropic position and inertia ellipsoids and zonoids of the unit ball of a normed n-dimensional space. Lecture Notes in Mathematics, pages 64–104. Springer Berlin Heidelberg, 1989.

[49] K. Mouratidis and M. L. Yiu. Shortest path computation with no information leakage. *Proceedings of the VLDB Endowment*, 5(8):692–703, 2012.

[50] A. Nikolov, K. Talwar, and L. Zhang. The geometry of differential privacy: The sparse and approximate cases. ACM STOC '13, pages 351–360, NY, USA, 2013.

[51] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

[52] S. Papadopoulos, S. Bakiras, and D. Papadias. Nearest neighbor search with strong location privacy. *Proceedings of the VLDB Endowment*, 3(1-2):619–629, 2010.

[53] W. H. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *ICDE*, pages 757–768, 2013.

[54] S. Qiao, C. Tang, H. Jin, T. Long, S. Dai, Y. Ku, and M. Chau. Putmode: prediction of uncertain trajectories in moving objects databases. *Applied Intelligence*, 33(3):370–386, Dec. 2010.

[55] V. Rastogi, M. Hay, G. Miklau, and D. Suciu. Relationship privacy: output perturbation for queries with joins. PODS '09, pages 107–116, New York, NY, USA, 2009. ACM.

[56] M. Rudelson. Random vectors in the isotropic position. *J. Funct. Anal*, pages 60–72, 1999.

[57] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. IEEE SP '11, pages 247–262, Washington, DC, USA, 2011.

[58] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[59] S. Vempala. Geometric random walks: a survey. *Combinatorial and Computational Geometry*, 2005.

[60] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 139–152. ACM, 2009.

[61] Y. Xiao, Y. Shen, J. Liu, L. Xiong, H. Jin, and X. Xu. DPHMM: customizable data release with differential privacy via hidden markov model. *CoRR*, abs/1609.09172, 2016.

[62] Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. CCS '15, pages 1298–1309. ACM, 2015.

[63] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering*, ICDE '13, 2013.

[64] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.