PGLP: Customizable and Rigorous Location Privacy through Policy Graph *

Yang Cao[1], Yonghui Xiao[2] **, Shun Takagi[1], Li Xiong[2], Masatoshi Yoshikawa[1], Yilin Shen[3], Jinfei Liu[2], Hongxia Jin[3], and Xiaofeng Xu[2]

[1] Kyoto University
yang@i.kyoto-u.ac.jp, s.takagi@db.soc.i.kyoto-u.ac.jp,
yoshikawa@i.kyoto-u.ac.jp
[2] Emory University
{lxiong, jliu253}@emory.edu {yohuxiao, xuxiaofeng1989}@gmail.com
[3] Samsung Research America
{yilin.shen, hongxia.jin}@samsung.com

Abstract. Location privacy has been extensively studied in the literature. However, existing location privacy models are either not rigorous or not customizable, which limits the trade-off between privacy and utility in many real-world applications. To address this issue, we propose a new location privacy notion called PGLP, i.e., *Policy Graph based Location Privacy*, providing a rich interface to release private locations with customizable and rigorous privacy guarantee. First, we design a rigorous privacy for PGLP by extending differential privacy. Specifically, we formalize location privacy requirements using a *location policy graph*, which is expressive and customizable. Second, we investigate how to satisfy an arbitrarily given location policy graph under realistic adversarial knowledge, which can be seen as constraints or public knowledge about user's mobility pattern. We find that a policy graph may not always be viable and may suffer *location exposure* when the attacker knows the user's mobility pattern. We propose efficient methods to detect location exposure and repair the policy graph with optimal utility. Third, we design an end-to-end location trace release framework that pipelines the detection of location exposure, policy graph repair, and private location release at each timestamp with customizable and rigorous location privacy. Finally, we conduct experiments on real-world datasets to verify the effectiveness and the efficiency of the proposed algorithms.

Keywords: Spatiotemporal data · Location Privacy · Trajectory Privacy · Differential Privacy · Location-Based Services.

1 Introduction

As GPS-enabled devices such as smartphones or wearable gadgets are pervasively used and rapidly developed, location data have been continuously generated,

* This work is partially supported by JSPS KAKENHI Grant No. 17H06099, 18H04093, 19K20269, U.S. National Science Foundation (NSF) under CNS-2027783 and CNS-1618932, and Microsoft Research Asia (CORE16).

** Yang and Yonghui contributed equally to this work.

collected, and analyzed. These personal location data connecting the online and offline worlds are precious, because they could be of great value for the society to enable ride sharing, traffic management, emergency planning, and disease outbreak control as in the current covid-19 pandemic via contact tracing, disease spread modeling, traffic and social distancing monitoring [4, 19, 26, 30].

On the other hand, privacy concerns hinder the extensive use of big location data generated by users in the real world. Studies have shown that location data could reveal sensitive personal information such as home and workplace, religious and sexual inclinations [35]. According to a survey [18], 78% smartphone users among 180 participants believe that Apps accessing their location pose privacy threats. As a result, the study of *private location release* has drawn increasing research interest and many location privacy models have been proposed in the last decades (see survey [34]).

However, existing location privacy models for private location releases are either not rigorous or not customizable. Following the seminal paper [22], the early location privacy models were designed based on k-anonymity [37] and adapted to different scenarios such as mobile P2P environments [14], trajectory release [3] and personalized k-anonymity for location privacy [21]. The follow-up studies revealed that k-anonymity might not be rigorous because it syntactically defines privacy as a property of the final "anonymized" dataset [29] and thus suffers many realistic attacks when the adversary has background knowledge about the dataset [28,31]. To this end, the state-of-the-art location privacy models [1,12,38,39] were extended from differential privacy (DP) [15] to private location release since DP is considered a rigorous privacy notion which defines privacy as a property of the algorithm. Although these DP-based location privacy models are rigorously defined, they are not customizable for different scenarios with various requirements on privacy-utility trade-off. Taking an example of Geo-Indistinguishability [1], which is the first DP-based location privacy, the protection level is solely controlled by a parameter ϵ to achieve indistinguishability between any two possible locations (the indistinguishability is scaled to the Euclidean distance between any two possible locations).

This one-size-fits-all approach may not fit every application's requirement on utility-privacy trade-off. Different location-based services (LBS) may have different usage of the data and thus need different *location privacy policies* to strike the right balance between privacy and utility. For instance, a proper location privacy policy for weather apps could be "*allowing the app to access a user's city-level location but ensuring indistinguishability among locations in each city*", which guarantees both reasonable privacy and high usability for a city-level weather forecast. Similarly, for POI recommendation [2], trajectory mining [32] or crowd monitoring during the pandemic [26], a suitable location privacy policy could be "*allowing the app to access the semantic category (e.g., a restaurant or a shop) of a user's location but ensuring indistinguishability among locations with the same category*", so that the LBS provider may know the user is at a restaurant or a shop, but not sure which restaurant or which shop.

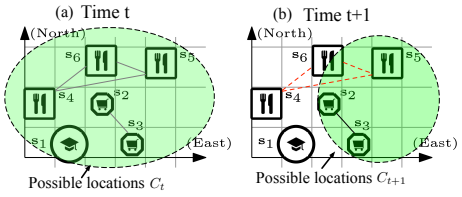

Fig. 1: An example of location policy graph and the constrained domains (i.e., possible locations) C_t and C_{t+1} at time t and $t+1$, respectively.

In this work, we study *how to release private location with customizable and rigorous privacy*. There are three significant challenges to achieve this goal. First, there is a lack of a rigorous and customizable location privacy metric and mechanisms. The closest work regarding customizable privacy is Blowfish privacy [25] for statistical data release, which uses a graph to represent a customizable privacy requirement, in which a node indicates a possible database instance to be protected, and an edge represents indistinguishability between the two possible databases. Blowfish privacy and its mechanisms are not applicable in our setting of private location release. It is because Blowfish privacy is defined on a statistical query over database with multiple users' data; whereas the input in the scenario of private location release is a single user's location.

The second challenge is how to satisfy an arbitrarily given location privacy policy under realistic adversarial knowledge, which is public knowledge about users' mobility pattern. In practice, as shown in [39,40], an adversary could take advantage of side information to rule out inadmissible locations[4] and reduce a user's possible locations into a small set, which we call *constrained domain*. We find that the location privacy policy may not be viable under a constrained domain and the user may suffer location exposure (we will elaborate how this could happen in Sec. 4.1).

The third challenge is how to release private locations continuously with high utility. We attempt to provide an end-to-end solution that takes the user's true location and a predefined location privacy policy as inputs, and outputs private location trace on the fly. We summarize the research questions below.

- How to design a rigorous location privacy metric with customizable location privacy policy? (Section 3)
- How to detect the problematic location privacy policy and repair it with high utility? (Sections 4)
- How to design an end-to-end framework to release private location continuously? (Section 5)

1.1 Contributions

In this work, we propose a new location privacy metric and mechanisms for releasing private location trace with flexible and rigorous privacy. To the best of our knowledge, this is the first DP-based location privacy notion with customizable privacy. Our contributions are summarized below.

[4] For example, it is impossible to move from Kyoto to London in a short time.

First, we formalize Policy Graph based Location Privacy (PGLP), which is a rigorous privacy metric extending differential privacy with a customizable *location policy graph*. Inspired by the statistical privacy notion of Blowfish privacy [25], we design location policy graph to represent which information needs to be protected and which does not. In a location policy graph (such as the one shown in Fig.1), the nodes are the user's possible locations, and the edges indicate the privacy requirements regarding the connected locations: an attacker should not be able to significantly distinguish which location is more probable to be the user's true location by observing the released location. PGLP is a general location privacy model compared with the prior art of DP-based location privacy notions, such as Geo-Indistinguishability [1] and Location Set Privacy [39]. We prove that they are two instances of PGLP under the specific configurations of the location policy graph. We also design mechanisms for PGLP by adapting the Laplace mechanism and Planar Isotropic Mechanism (PIM) (i.e., the optimal mechanism for Location Set Privacy [39]) w.r.t. a given location policy graph.

Second, we design algorithms that examine the feasibility of a given location policy graph under adversarial knowledge about the user's mobility pattern modeled by Markov Chain. We find that the policy graph may not always be viable. Specially, as shown in Fig.1, some nodes (locations) in a policy graph may be *excluded* (e.g., s_4 and s_6 in Fig.1 (b)) or *disconnected* (e.g., s_5 in Fig.1 (b)) due to the limited set of the possible locations. Protecting the excluded nodes is a lost cause, but it is necessary to protect the disconnected nodes since it may lead to location exposure when the user is at such a location. Surprisingly, we find that a disconnected node may *not always* result in the location exposure, which also depends on the protection strength of the mechanism. Intuitively, this happens when a mechanism "overprotects" a location policy graph by implicitly guaranteeing indistinguishability that is not enforced by the policy. We design an algorithm to detect the disconnected nodes that suffer location exposure , which are named *isolated node*. We also design a *graph repair* algorithm to ensure no isolated node in a location policy graph by adding an optimal edge between the isolated node and another node with high utility.

Third, we propose an end-to-end private location trace release framework with PGLP that takes inputs of the user's true location at each time t and outputs private location continuously satisfying a pre-defined location policy graph. The framework pipelines the calculation of constrained domains, isolated node detection, policy graph repair, and private location release mechanism. We also reason about the overall privacy guarantee in multiple releases.

Finally, we implement and evaluate the proposed algorithms on real-world datasets, showing that privacy and utility can be better tuned with customizable location policy graphs.

2 Preliminaries

2.1 Location Data Model

Similar to [39, 40], we employ two coordinate systems to represent locations for applicability for different application scenarios. A location can be represented by

an index of *grid coordinates* or by a two-dimension vector of *longitude-latitude coordinate* to indicate any location on the map. Specifically, we partition the map into a grid such that each grid cell corresponds to an area (or a point of interest); any location represented by a longitude-latitude coordinate will also have a grid number or index on the grid coordinate. We denote the location domain as $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N\}$ where each $\mathbf{s}_i$ corresponds to a grid cell on the map, $1 \leq i \leq N$. We use $\mathbf{s}_t^*$ and $\mathbf{z}_t$ to denote the user's true location and perturbed location at time t. We also use t in $\mathbf{s}^*$ and $\mathbf{z}$ to refer the locations at a single time when it is clear from the context.

Location Query For the ease of reasoning about privacy and utility, we use a location query $f : \mathcal{S} \to \mathbb{R}^2$ to represent the mapping from locations to the longitude and latitude of the center of the corresponding grid cell.

2.2 Problem Statement

Given a moving user on a map $\mathcal{S}$ in a time period $\{1, 2, \cdots, T\}$, our goal is to release the perturbed locations of the user to untrusted third parties at each timestamp under a pre-defined location privacy policy. We define ϵ-Indistinguishability as a building block for reasoning about our privacy goal.

Definition 1 (ϵ-Indistinguishability). *Two locations $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are ϵ-indistinguishable under a randomized mechanism $\mathcal{A}$ iff for any output $\boldsymbol{z} \subseteq Range(\mathcal{A})$, we have $\frac{\Pr(\mathcal{A}(\boldsymbol{s}_i)=\boldsymbol{z})}{\Pr(\mathcal{A}(\boldsymbol{s}_j)=\boldsymbol{z})} \leq e^{\epsilon}$, where $\epsilon \geq 0$.*

As we exemplified in the introduction, different LBS applications may have different metrics of utility. We aim at providing better utility-privacy trade-off by customizable ϵ-Indistinguishability between locations.

Adversarial Model We assume that the attackers know the user's mobility pattern modeled by Markov chain, which is widely used for modeling user mobility profiles [10, 20]. We use matrix $\mathbf{M} \in [0, 1]^{N \times N}$ to denote the transition probabilities of Markov chain with m_{ij} being the probability of moving from location $\mathbf{s}_i$ to location $\mathbf{s}_j$. Another adversarial knowledge is the initial probability distribution of the user's location at $t = 1$. To generalize the notation, we denote probability distribution of the user's location at t by a vector $\mathbf{p}_t \in [0, 1]^{1 \times N}$, and denote the ith element in $\mathbf{p}_t$ by $\mathbf{p}_t[i] = \Pr(\mathbf{s}_t^* = \mathbf{s}_i)$, where $\mathbf{s}_t^*$ is the user's true location at t and $\mathbf{s}_i \in \mathcal{S}$. Given the above knowledge, the attackers could infer the user's possible locations at time t, which is probably smaller than the location domain $\mathcal{S}$, and we call it a *constrained domain*.

Definition 2 (Constrained domain). *We denote $\mathcal{C}_t = \{\boldsymbol{s}_i | \Pr(\boldsymbol{s}_t^* = \boldsymbol{s}_i) > 0, \boldsymbol{s}_i \in \mathcal{S}\}$ as constrained domain, which indicates a set of possible locations at t.*

We note that the constrained domain can be explained as the requirement of LBS applications. For example, an App could only be used within a certain area, such as a university free shuttle tracker.

3 Policy Graph based Location Privacy

In this section, we first formalize the privacy requirement using *location policy graph* in Sec. 3.1. We then design the privacy metric of PGLP in Sec. 3.2. Finally, we propose two mechanisms for PLGP in Sec. 3.3.

3.1 Location Policy Graph

Inspired by Blowfish privacy [25], we use an undirected graph to define what should be protected, i.e., privacy policies. The nodes are secrets, and the edges are the required indistinguishability, which indicates an attacker should not be able to distinguish the input secrets by observing the perturbed output. In our setting, we treat possible locations as nodes and the indistinguishability between the locations as edges.

Definition 3 (Location Policy Graph). *A location policy graph is an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ where $\mathcal{S}$ denotes all the locations (nodes) and $\mathcal{E}$ represents indistinguishability (edges) between these locations.*

Definition 4 (Distance in Policy Graph). *We define the distance between two nodes $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ in a policy graph as the length of the shortest path (i.e., hops) between them, denoted by $d_{\mathcal{G}}(\boldsymbol{s}_i, \boldsymbol{s}_j)$. If $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ are disconnected, $d_{\mathcal{G}}(\boldsymbol{s}_i, \boldsymbol{s}_j) = \infty$.*

In DP, the two possible database instances with or without a user's data are called *neighboring databases*. In our location privacy setting, we define neighbors as two nodes with an edge in a policy graph.

Definition 5 (Neighbors). *The neighbors of location $\boldsymbol{s}$, denoted by $\mathcal{N}(\boldsymbol{s})$, is the set of nodes having an edge with $\boldsymbol{s}$, i.e., $\mathcal{N}(\boldsymbol{s}) = \{\boldsymbol{s}' | d_{\mathcal{G}}(\boldsymbol{s}, \boldsymbol{s}') = 1, \boldsymbol{s}' \in \mathcal{S}\}$.*

We denote the nodes having a path with $\mathbf{s}$ by $\mathcal{N}^P(\mathbf{s})$, i.e., the nodes in the same connected component with $\mathbf{s}$. In our framework, we assume the policy graph is given and public. In practice, the location privacy policy can be defined application-wise and identical for all users using the same application.

3.2 Definition of PGLP

We now formalize Policy Graph based Location Privacy (PGLP), which guarantees ϵ-indistinguishability in Definition 1 for every pair of neighbors (i.e., for each edge) in a given location policy graph.

Definition 6 ($\{\epsilon, \mathcal{G}\}$-Location Privacy). *A randomized algorithm $\mathcal{A}$ satisfies $\{\epsilon, \mathcal{G}\}$-location privacy iff for all $\boldsymbol{z} \subseteq Range(\mathcal{A})$ and for all pairs of neighbors $\boldsymbol{s}$ and $\boldsymbol{s}'$ in $\mathcal{G}$, we have $\frac{\Pr(\mathcal{A}(s)=z)}{\Pr(\mathcal{A}(s')=z)} \leq e^{\epsilon}$.*

In PGLP, privacy is rigorously guaranteed through ensuring indistinguishability between any two neighboring locations specified by a customizable location policy graph. The above definition implies the indistinguishability between two nodes that have a path in the policy graph.

Lemma 1. *An algorithm $\mathcal{A}$ satisfies $\{\epsilon, \mathcal{G}\}$-location privacy, iff any two nodes $\boldsymbol{s}_i, \boldsymbol{s}_j \in \mathcal{G}$ are $\epsilon \cdot d_{\mathcal{G}}(\boldsymbol{s}_i, \boldsymbol{s}_j)$-indistinguishable.*

Lemma 1 indicates that, if there is a path between two nodes $\mathbf{s}_i, \mathbf{s}_j$ in the policy graph, the corresponding indistinguishability is required at a certain degree; if two nodes are disconnected, the indistinguishability is not required (i.e., can be ∞) by the policy. As an extreme case, if a node is disconnected with any other nodes, it is allowed to be released without any perturbation.

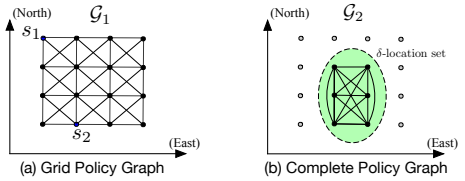

Fig. 2: Two examples of location policy graphs.

Comparison with Other Location Privacy Models. We analyze the relation between PGLP and two well-known DP-based location privacy models, i.e., Geo-Indistinguishability (Geo-Ind) [1] and δ-Location Set Privacy [39]. We show that PGLP can represent them under proper configurations of policy graphs.

Geo-Ind [1] guarantees a level of indistinguishability between two locations $\mathbf{s}_i$ and $\mathbf{s}_j$ that is scaled with their Euclidean distance, i.e., $\epsilon \cdot d_E(\mathbf{s}_i, \mathbf{s}_j)$-indistinguishability, where $d_E(\cdot, \cdot)$ denotes Euclidean distance. Note that the unit length used in Geo-Ind scales the level of indistinguishability. We assume that, for any neighbors $\mathbf{s}$ and $\mathbf{s}'$, the unit length used in Geo-Ind makes $d_E(\mathbf{s}, \mathbf{s}') \geq 1$.

Let $\mathcal{G}_1$ be a location policy graph that every location has edges with its closest eight locations on the map, as shown in Fig.2 (a). We can derive Theorem 1 by Lemma 1 with the fact of $d_{\mathcal{G}}(\mathbf{s}_i, \mathbf{s}_j) \leq d_E(\mathbf{s}_i, \mathbf{s}_j)$ for any $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{G}_1$ (e.g., in Fig.2(a), $d_{\mathcal{G}}(\mathbf{s}_1, \mathbf{s}_2) = 3$ and $d_E(\mathbf{s}_1, \mathbf{s}_2) = \sqrt{10}$).

Theorem 1. *An algorithm satisfying $\{\epsilon, \mathcal{G}_1\}$-location privacy also achieves ϵ-Geo-Indistinguishability.*

δ-Location Set Privacy [39] extends differential privacy on a subset of possible locations, which is assumed as adversarial knowledge. We note that the constrained domain in Definition 2 can be considered a generalization of δ-location set, whereas we do not specify the calculation of this set in PGLP. δ-Location Set Privacy ensures indistinguishability among any two locations in the δ-location set. Let $\mathcal{G}_2$ be a location policy graph that is complete, i.e., fully connected among all locations in the δ-location set as shown in Fig.2(b).

Theorem 2. *An algorithm satisfying $\{\epsilon, \mathcal{G}_2\}$-location privacy also achieves δ-Location Set privacy.*

We defer the proofs of the theorems to a full version because of space limitation.

3.3 Mechanisms for PGLP

In the following, we show how to transform existing DP mechanisms into one satisfying PGLP using *graph-calibrated sensitivity*. We temporarily assume the constrained domain $\mathcal{C} = \mathcal{S}$ and study the effect of $\mathcal{C}$ on policy $\mathcal{G}$ in Section 4.

As shown in Section 2.1, the problem of private location release can be seen as answering a location query $f : \mathcal{S} \to \mathbb{R}^2$ privately. Then we can adapt the existing DP mechanism for releasing private locations by adding random noises to longitude and latitude independently. We use this approach below to adapt the Laplace mechanism and Planar Isotropic Mechanism (PIM) (i.e., an optimal mechanism for Location Set Privacy [39]) to achieve PGLP.

Policy-based Laplace Mechanism (P-LM). Laplace mechanism is built on the ℓ_1-norm sensitivity [16], defined as the maximum change of the query results due to the difference of neighboring databases. In our setting, we calibrate this sensitivity w.r.t. the neighbors specified in a location policy graph.

Definition 7 (Graph-calibrated ℓ_1-norm Sensitivity). *For a location $\boldsymbol{s}$ and a query $f(\boldsymbol{s})$: $\boldsymbol{s} \to \mathbb{R}^2$, its ℓ_1-norm sensitivity $S_f^{\mathcal{G}}$ is the maximum ℓ_1 norm of $\Delta f^{\mathcal{G}}$ where $\Delta f^{\mathcal{G}}$ is a set of points (i.e., two-dimension vectors) of $\big(f(\boldsymbol{s}_i) - f(\boldsymbol{s}_j)\big)$ for $\boldsymbol{s}_i, \boldsymbol{s}_j \in \mathcal{N}^P(\boldsymbol{s})$ (i.e., the nodes with the same connected component of $\boldsymbol{s}$).*

We note that, for a true location $\mathbf{s}$, releasing $\mathcal{N}^P(\mathbf{s})$ does not violate the privacy defined by the policy graph. It is because, for any connected $\mathbf{s}$ and $\mathbf{s}'$, $\mathcal{N}^P(\mathbf{s})$ and $\mathcal{N}^P(\mathbf{s}')$ are the same; while, for any disconnected $\mathbf{s}$ and $\mathbf{s}'$, the indistinguishability between $\mathcal{N}^P(\mathbf{s})$ and $\mathcal{N}^P(\mathbf{s}')$ is not required by Definition 6.

Algorithm 1 Policy-based Laplace Mechanism (P-LM)

Require: ϵ, $\mathcal{G}$, the user's true location $\mathbf{s}$.
1: Calculate $S_f^{\mathcal{G}} = sup||\big(f(\mathbf{s}_i) - f(\mathbf{s}_j)\big)||_1$ for all neighbors $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{N}^P(\mathbf{s})$;
2: Perturb location $\mathbf{z}' = f(\mathbf{s}) + [Lap(S_f^{\mathcal{G}}/\epsilon), Lap(S_f^{\mathcal{G}}/\epsilon)]^T$;
3: **return** a location $\mathbf{z} \in \mathcal{S}$ that is closest to $\mathbf{z}'$ on the map.

Theorem 3. *P-LM satisfies $\{\epsilon, \mathcal{G}\}$-location privacy.*

Policy-based Planar Isotropic Mechanism (P-PIM). PIM [39] achieves the low bound of differential privacy on two-dimension space for Location Set Privacy. It adds noises to longitude and latitude using K-norm mechanism [24] with *sensitivity hull* [39], which extends the convex hull of the sensitivity space in K-norm mechanism. We propose a *graph-calibrated sensitivity hull* for PGLP.

Definition 8 (Graph-calibrated Sensitivity Hull). *For a location $\boldsymbol{s}$ and a query $f(\boldsymbol{s})$: $\boldsymbol{s} \to \mathbb{R}^2$, the graph-calibrated sensitivity hull $K(\mathcal{G})$ is the convex hull of $\Delta f^{\mathcal{G}}$ where $\Delta f^{\mathcal{G}}$ is a set of points (i.e., two-dimension vectors) of $\big(f(\boldsymbol{s}_i) - f(\boldsymbol{s}_j)\big)$ for any $\boldsymbol{s}_i, \boldsymbol{s}_j \in \mathcal{N}^P(\boldsymbol{s})$ and $\boldsymbol{s}_i, \boldsymbol{s}_j$ are neighbors, i.e., $K(\mathcal{G}) = Conv(\Delta f^{\mathcal{G}})$.*

We note that, in Definitions 7 and 8, the sensitivities are independent of the true location $\mathbf{s}$ and all the nodes in $\mathcal{N}(\mathbf{s})$ have the same sensitivity.

Definition 9 (K-norm Mechanism [24]). *Given any function $f(\boldsymbol{s})$: $\boldsymbol{s} \to \mathbb{R}^d$ and its sensitivity hull K, K-norm mechanism outputs $\boldsymbol{z}$ withh probability below.*

$$\Pr(\boldsymbol{z}) = \frac{1}{\Gamma(d+1)\mathrm{VOL}(K/\epsilon)} exp\left(-\epsilon||\boldsymbol{z} - f(\boldsymbol{s})||_K\right) \tag{1}$$

where $\Gamma(\cdot)$ is Gamma function and $\mathrm{VOL}(\cdot)$ denotes volume.

Algorithm 2 Policy-based Planar Isotropic Mechanism (P-PIM)

Require: ϵ, $\mathcal{G}$, the user's true location $\mathbf{s}$.
1: Calculate $K(\mathcal{G}) = Conv\big(f(\mathbf{s}_i) - f(\mathbf{s}_j)\big)$ for all neighbors $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{N}^P(\mathbf{s})$;
2: $\mathbf{z}' = f(\mathbf{s}) + Y$ where Y is two-dimension noise drawn by Eq.(1) with sensitivity hull $K(\mathcal{G})$;
3: **return** a location $\mathbf{z} \in \mathcal{S}$ that is closest to $\mathbf{z}'$ on the map.

Theorem 4. *P-PIM satisfies $\{\epsilon, \mathcal{G}\}$-location privacy.*

We can prove Theorems 3 and 4 using Lemma 1. The sensitivity is scaled with the graph-based distance. We note that directly using Laplace machanism or PIM can satisfy a fully connected policy graph over locations in the constrained domain as shown in Fig.2(b).

Theorem 5. *Algorithm 2 has the time complexity* $O(|\mathcal{C}|\log(h)+h^2\log(h))$ *where* h *is number of vertices on the polygon of* $Conv(\Delta f^{\mathcal{G}})$.

4 Policy Graph under Constrained Domain

In this section, we investigate and prevent the location exposure of a policy graph under constrained domain in Sec. 4.1 and 4.2, respectively; then we repair the policy graph in Sec. 4.3.

4.1 Location Exposure

As shown in Fig.1 (right) and introduced in Section 1, a given policy graph may not be viable under adversarial knowledge of constrained domain (Definition 2). We illustrate the potential risks due to the constrained domain shown in Fig.3.

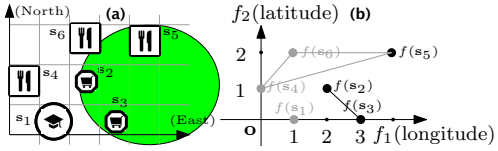

Fig. 3: **(a)** The constrained domain $\mathcal{C} = \{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_5\}$; **(b)** The constrained policy graph.

We first examine the immediate consequences of the constrained domain to the policy graph by defining the excluded and disconnected nodes. We then show the disconnected node may lead to *location exposure*.

Definition 10 (Excluded node). *Given a location policy graph* $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ *and a constrained domain* $\mathcal{C} \subset \mathcal{S}$*, if* $s \in \mathcal{S}$ *and* $s \notin \mathcal{C}$*,* $\boldsymbol{s}$ *is an excluded node.*

Definition 11 (Disconnected node). *Given a location policy graph* $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ *and a constrained domain* $\mathcal{C} \subset \mathcal{S}$*, if a node* $\boldsymbol{s} \in \mathcal{C}$*,* $\mathcal{N}(\boldsymbol{s}) \neq \emptyset$ *and* $\mathcal{N}(\boldsymbol{s}) \cap \mathcal{C} = \emptyset$*, we call* $\boldsymbol{s}$ *a disconnected node.*

Intuitively, the *excluded* node is outside of the constrained domain $\mathcal{C}$, such as the gray nodes $\{\mathbf{s}_1, \mathbf{s}_4, \mathbf{s}_6\}$ in Fig.3; whereas the *disconnected* node (e.g., $\mathbf{s}_5$ in Fig.3) is inside of $\mathcal{C}$ and has neigbors, yet all its neighbors are outside of $\mathcal{C}$.

Next, we analyze the feasibility of a location policy graph under a constrained domain. The first problem is that, by the definition of excluded nodes, it is not possible to achieve indistinguishability between the excluded nodes and any other nodes. For example in Fig.3, the indistinguishability indicated by the gray edges is not feasible because of $\Pr(\mathcal{A}(\mathbf{s}_4) = \mathbf{z}) = \Pr(\mathcal{A}(\mathbf{s}_6) = \mathbf{z}) = 0$ for any $\mathbf{z}$ given the adversarial knowledge of $\Pr(\mathbf{s}_4) = \Pr(\mathbf{s}_6) = 0$. Hence, one can only achieve a *constrained policy graph*, such as the one with nodes $\{\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_5\}$ in Fig.3(b).

Definition 12 (Constrained Location Policy Graph). *A constrained location policy graph* $\mathcal{G}^{\mathcal{C}}$ *is a subgraph of the original location policy graph* $\mathcal{G}$ *under a constrained domain* $\mathcal{C}$ *that only includes the edges inside of* $\mathcal{C}$*. Formally,* $\mathcal{G}^{\mathcal{C}} = (\mathcal{C}, \mathcal{E}^{\mathcal{C}})$ *where* $\mathcal{C} \subseteq \mathcal{S}$ *and* $\mathcal{E}^{\mathcal{C}} \subseteq \mathcal{E}$.

Definition 13 (Location Exposure under constrained domain). *Given a policy graph $\mathcal{G}$, constrained domain $\mathcal{C}$ and an algorithm $\mathcal{A}$ satisfying $(\epsilon, \mathcal{G}^{\mathcal{C}})$-location privacy, for a disconnected node $\boldsymbol{s}$, if $\mathcal{A}$ does not guarantee ϵ-indistinguishability between $\boldsymbol{s}$ and any other nodes in $\mathcal{C}$, we call $\boldsymbol{s}$ an isolated node. The user suffers location exposure when she is at the location of the isolated node.*

4.2 Detecting Isolated Node

An interesting finding is that a disconnected node may not always lead to location exposure, which also depends on the algorithm for PGLP. Intuitively, the indistinguishability between a disconnected node and a node in the constrained domain could be guaranteed implicitly. We design Algorithm 3 to detect the isolated node in a constrained policy graph w.r.t. P-PIM. It could be extended to any other PGLP mechanism. For each disconnected node, we check whether it is indistinguishable with other nodes. The problem is equivalent to checking if there is any node inside the convex body $f(\mathbf{s}_i) + K(\mathcal{G}^{\mathcal{C}})$, which can be solved by the convexity property (if a point $\mathbf{s}_j$ is inside a convex hull K, then $\mathbf{s}_j$ can be expressed by the vertices of K with coefficients in $[0, 1]$) with complexity $O(m^3)$. We design a faster method with complexity $O(m^2 log(m))$ by exploiting the definition of convex hull: if $\mathbf{s}_j$ is inside $f(\mathbf{s}_i) + K(\mathcal{G}^{\mathcal{C}})$, then the new convex hull of the new graph by adding edge $\overline{\mathbf{s}_i\mathbf{s}_j}$ will be the same as $K(\mathcal{G}^{\mathcal{C}})$. We give an example of *disconnected but not isolated* node in appendix A.

Algorithm 3 Finding Isolated Node

Require: $\mathcal{G}$, $\mathcal{C}$, disconnected node $\mathbf{s}_i \in \mathcal{C}$.
1: $\Delta f^{\mathcal{G}} = \bigvee_{\overline{\mathbf{s}_j\mathbf{s}_k} \in \mathcal{E}^{\mathcal{C}}} (f(\mathbf{s}_j) - f(\mathbf{s}_k))$; ▷ We use $\vee$ to denote Union operator.
2: $K(\mathcal{G}^{\mathcal{C}}) \leftarrow Conv(\Delta f^{\mathcal{G}})$;
3: **for all** $\mathbf{s}_j \in \mathcal{C}, \mathbf{s}_j \neq \mathbf{s}_i$ **do**
4: **if** $Conv(\Delta f^{\mathcal{G}}, f(\mathbf{s}_j) - f(\mathbf{s}_i)) == K(\mathcal{G}^{\mathcal{C}})$ **then**
5: **return** false ▷ not isolated
6: **end if**
7: **end for**
8: **return** true ▷ isolated

4.3 Repairing Location Policy Graph

To prevent location exposure under the constrained domain, we need to make sure that there is no isolated node in a constrained policy graph. A simple way is to modify the policy graph to ensure the indistinguishability of the isolated node by adding an edge between it and another node in the constrained domain. The selection of such a node could depend on the requirement of the application. Without the loss of generality, a baseline method for repairing the policy graph could be choosing an *arbitrary* node from the constrained domain and adding an edge between it and the isolated node.

A natural question is how can we repair the policy graph with better utility. Since different ways of adding edges in the policy graph may lead to distinct graph-based sensitivity, which is propositional to the area of the sensitivity hull (i.e., a polygon on the map), the question is equivalent to adding an edge with the *minimum* area of sensitivity hull (thus the least noise). We design Algorithm 4 to find the minimum area of the new sensitivity hull, as shown in an example in Fig.4. The analysis is shown in Appendix B.

We note that both Algorithms 3 and 4 are oblivious to the true location, so they do not consume the privacy budget. Additionally, the adversary may be able to "reverse" the process of graph repair and extract the information about the original location policy graph; however, this does not compromise our privacy notion since the location policy graph is public in our setting.

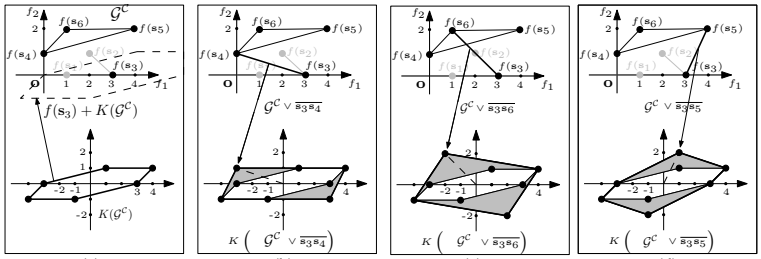

Fig. 4: An example of graph repair with high utility. (a): if $\mathcal{C} = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$, then $\mathbf{s}_3$ is isolated because $f(\mathbf{s}_3) + K(\mathcal{G}^{\mathcal{C}})$ only contains $\mathbf{s}_3$; to protect $\mathbf{s}_3$, we can re-connect $\mathbf{s}_3$ to one of the valid nodes $\{\mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$. (b) shows the new sensitivity hull after adding $f(\mathbf{s}_4) - f(\mathbf{s}_3)$ to $K(\mathcal{G}^{\mathcal{C}})$; (c) shows the new sensitivity hull after adding $f(\mathbf{s}_6) - f(\mathbf{s}_3)$ to $K(\mathcal{G}^{\mathcal{C}})$; (d) shows the new sensitivity hull after adding $f(\mathbf{s}_5) - f(\mathbf{s}_3)$ to $K(\mathcal{G}^{\mathcal{C}})$. Because (b) has the smallest area of the sensitivity hull, $\mathbf{s}_3$ should be connected to $\mathbf{s}_4$.

Algorithm 4 Graph Repair with High Utility

Require: $\mathcal{G}$, $\mathcal{C}$, isolated node $\mathbf{s}_i$
1: $\mathcal{G}^{\mathcal{C}} \leftarrow \mathcal{G} \wedge \mathcal{C}$;
2: $K \leftarrow K(\mathcal{G}^{\mathcal{C}})$;
3: $\mathbf{s}_k \leftarrow \emptyset$;
4: $minArea \leftarrow \infty$;
5: **for all** $\mathbf{s}_j \in \mathcal{C}, \mathbf{s}_j \neq \mathbf{s}_i$ **do**
6: $K \leftarrow K(\mathcal{G}^{\mathcal{C}} \vee \overline{\mathbf{s}_i\mathbf{s}_j})$; ▷ new sensitivity hull in $O(mlog(m))$
7: $Area = \sum_{i=1,j=i+1}^{i=h} det(\mathbf{v}_i, \mathbf{v}_j)$ where $\mathbf{v}_{h+1} = \mathbf{v}_1$; ▷ $\Theta(h)$ time
8: **if** $Area < minArea$ **then**
9: $\mathbf{s}_k \leftarrow \mathbf{s}_j$;
10: $minArea = Area$; ▷ find minimum area
11: **end if**
12: **end for**
13: $\mathcal{G}^{\mathcal{C}} \leftarrow \mathcal{G}^{\mathcal{C}} \vee \overline{\mathbf{s}_i\mathbf{s}_k}$ ▷ add edge $\overline{\mathbf{s}_i\mathbf{s}_k}$ to the graph
14: return repaired policy graph $\mathcal{G}^{\mathcal{C}}$;

5 Location Trace Release with PGLP

5.1 Location Release via Hidden Markov Model

A remaining question for continuously releasing private location with PGLP is how to calculate the adversarial knowledge of constrained domain $\mathcal{C}_t$ at each time t. According to our adversary model described in Sec. 2.2, the attacker knows the user's mobility pattern modeled by the Markov chain and the initial probability distribution of the user's location. The attacker also knows the released mechanisms for PGLP. Hence, the problem of calculating the possible location domain (i.e., locations that $\Pr(\mathbf{s}_t^*) > 0$) can be modeled as an inference problem in Hidden Markov Model (HMM) in Figure 5: the attacker attempts to infer the probability distribution of the true location $\mathbf{s}_t^*$, given the PGLP mechanism, the

Markov model of $\mathbf{s}_t^*$, and the observation of $\mathbf{z}_1, \cdots, \mathbf{z}_t$ at the current time t. The constrained domain at each time is derived as the locations in the probability distribution of $\mathbf{s}_t^*$ with non-zero probability.

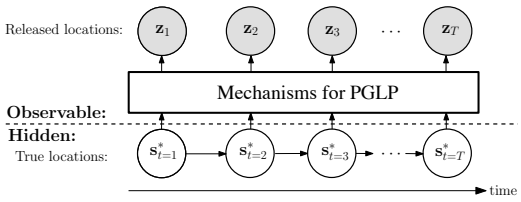

Fig. 5: Private location trace release via HMM.

We elaborate the calculation of the probability distribution of $\mathbf{s}_t^*$ as follows. The probability $\Pr(\mathbf{z}_t|\mathbf{s}_t^*)$ denotes the distribution of the released location $\mathbf{z}_t$ where $\mathbf{s}_t^*$ is the true location at any timestamp t. At timestamp t, we use $\mathbf{p}_t^-$ and $\mathbf{p}_t^+$ to denote the prior and posterior probabilities of an adversary about current state before and after observing $\mathbf{z}_t$ respectively. The prior probability can be derived by the (posterior) probability at previous timestamp $t-1$ and the Markov transition matrix as $\mathbf{p}_t^- = \mathbf{p}_{t-1}^+ \mathbf{M}$. The posterior probability can be computed using Bayesian inference as follows. For each state $\mathbf{s}_i$:

$$\mathbf{p}_t^+[i] = \Pr(\mathbf{s}_t^* = \mathbf{s}_i|\mathbf{z}_t) = \frac{\Pr(\mathbf{z}_t|\mathbf{s}_t^* = \mathbf{s}_i)\mathbf{p}_t^-[i]}{\sum_j Pr(\mathbf{z}_t|\mathbf{s}_t^* = \mathbf{s}_j)\mathbf{p}_t^-[j]} \tag{2}$$

Algorithm 5 shows the location trace release algorithm. At each timestamp t, we compute the constrained domain (Line 2). For all disconnected nodes under the constrained domain, we check if they are isolated by Algorithm 3. If so, we derive a minimum protectable graph $\mathcal{G}_t$ by Algorithm 4. Next, we use the proposed PGLP mechanisms (i.e., P-LM or P-PIM) to release a perturbed location $\mathbf{z}_t$. Then the released $\mathbf{z}_t$ will also be used to update the posterior probability $\mathbf{p}_t^+$ (in the equation below) by Equation (2), which subsequently will be used to compute the prior probability for the next time $t+1$. We note that, only Line 9 (invoking PGLP mechanisms) uses the true location $\mathbf{s}_t^*$. Algorithms 3 and 4 are independent of the true location, so they do not consume the privacy budget.

Algorithm 5 Location Trace Release Mechanism for PGLP

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

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: $\mathcal{G}_t^{\mathcal{C}} \leftarrow \mathcal{G} \wedge \mathcal{C}_t$; ▷ Definition 12
4: **for all** disconnected node $\mathbf{s}_i$ in $\mathcal{G}_t^{\mathcal{C}}$ **do**
5: **if** $\mathbf{s}_i$ is isolated **then** ▷ isolated node detection by Algorithm 3
6: $\mathcal{G}_t^{\mathcal{C}} \leftarrow$ Algorithm 4($\mathcal{G}_t^{\mathcal{C}}$, $\mathcal{C}_t$, $\mathbf{s}_i$); ▷ repair graph $\mathcal{G}_t$ by Algorithm 4
7: **end if**
8: **end for**
9: mechanisms for PGLP with parameters ϵ, $\mathbf{s}_t^*$, $\mathcal{G}_t$; ▷ Algorithms 1 or 2
10: Derive $\mathbf{p}_t^+$ by Equation (2); ▷ inference go to next timestamp
11: **return** Algorithm 5(ϵ, $\mathcal{G}_t^{\mathcal{C}}$, $\mathbf{M}$, $\mathbf{p}_t^+$, $\mathbf{s}_{t+1}^*$);

Theorem 6 (Complexity). *Algorithm 5 has complexity $O(dm^2 log(m))$ where d is the number of disconnected nodes and m is the number of nodes in $\mathcal{G}_t^{\mathcal{C}}$.*

5.2 Privacy Composition

We analyze the composition of privacy for multiple location releases under PGLP. In Definition 6, we define $\{\epsilon, \mathcal{G}\}$-location privacy for single location release, where ϵ can be considered the privacy leakage w.r.t. the privacy policy $\mathcal{G}$. A natural question is what would be the privacy leakage of multiple releases at a single timestamp (i.e., for the same true location) or at multiple timestamps (i.e., for a trajectory). In either case, the privacy guarantee (or the upper bound of privacy leakage) in multiple releases depends on the achievable location policy graphs. Hence, the key is to study the composition of the policy graphs in multiple releases. Let $\mathcal{A}_1, \cdots, \mathcal{A}_T$ be T independent random algorithms that takes true locations $\mathbf{s}_1^*, \cdots, \mathbf{s}_T^*$ as inputs (note that it is possible $\mathbf{s}_1^* = \cdots = \mathbf{s}_T^*$) and outputs $\mathbf{z}_1, \cdots, \mathbf{z}_T$, respectively. When the viable policy graphs are the same at each release, we have Lemma 2 as below.

Lemma 2. *If all $\mathcal{A}_1, \cdots, \mathcal{A}_T$ satisfy $(\epsilon, \mathcal{G})$-location privacy, the combination of $\{\mathcal{A}_1, \cdots, \mathcal{A}_T\}$ satisfies $(T\epsilon, \mathcal{G})$-location privacy.*

As shown in Sec. 4, the feasibility of achieving a policy graph is affected by the constrained domain, which may change along with the released locations. We denote $\mathcal{G}_1, \cdots, \mathcal{G}_T$ as viable policy graphs at each release (for single location or for a trajectory), which could be obtained by algorithms in Sec. 4.2 and Sec. 4.3. We give a more general composition theorem for PGLP below.

Theorem 7. *If $\mathcal{A}_1, \cdots, \mathcal{A}_T$ satisfy $(\epsilon_1, \mathcal{G}_1), \cdots, (\epsilon_T, \mathcal{G}_T)$, -location privacy, respectively, the combination of $\{\mathcal{A}_1, \cdots, \mathcal{A}_T\}$ satisfies $\left(\sum_{i=1}^{T} \epsilon_i, \mathcal{G}_1 \wedge \cdots \wedge \mathcal{G}_T\right)$-location privacy, where $\wedge$ denotes the intersection between the edges of policy graphs.*

The above theorem provides a method to reason about the overall privacy in continuous releases using PGLP. We note that the privacy composition does not depend on the adversarial knowledge of Markov model, but replies on the soundness of the policy graph and PGLP mechanisms at each t. However, the resulting $\mathcal{G}_1 \wedge \cdots \wedge \mathcal{G}_T$ may not be the original policy graph. It is an interesting future work to study how to ensure a given policy graph across the timeline.

6 Experiments

6.1 Experimental Setting

We implement the algorithms use Python 3.7. The code is available in github[5]. We run the algorithms on a machine with Intel core i7 6770k CPU and 64 GB of memory running Ubuntu 15.10 OS.

Datasets. We evaluate the algorithms on three real-world datasets with similar configurations in [39] for comparison purpose. The Markov models were learned from the raw data. For each dataset, we randomly choose 20 users' location trace with 100 timestamps for testing.

[5] https://github.com/emory-aims/pglp.

- Geolife dataset [41] contains tuples with attributes of user ID, latitude, longitude and timestamp. We extracted all the trajectories within the Fourth Ring of Beijing to learn the Markov model, with the map partitioned into cells of $0.34 \times 0.34\ km^2$.
- Gowalla dataset [13] contains $6,442,890$ check-in locations of $196,586$ users over 20 months. We extracted all the check-ins in Los Angeles to train the Markov model, with the map partitioned into cells of $0.37 \times 0.37\ km^2$.
- Peopleflow dataset[6] includes $102,468$ locations of $11,406$ users with semantic labels of POI in Tokyo. We partitioned the map into cells of $0.27 \times 0.27\ km^2$.

Policy Graphs. We evaluate two types of location privacy policy graphs for different applications as introduced in Section 1. One is for the policy of "*allowing the app to access a user's location in which area but ensuring indistinguishability among locations in each area*", represented by G_{k9}, G_{k16}, G_{k25} below. The other is for the policy of "*allowing the app to access the semantic label (e.g., a restaurant or a shop) of a user's location but ensuring indistinguishability among locations with the same category*", represented by G_{poi} below.

- G_{k9} is a policy graph that all locations in each 3×3 region (i.e., 9 grid cells using *grid coordinates*) are fully connected with each other. Similarly, we have G_{k16} and G_{k25} for region size 4×4 and 5×5, respectively.
- G_{poi}: all locations with both the same category and the same 6×6 region are fully connected. We test the category of restaurant in Peopleflow dataset.

Utility Metrics. We evaluate three types of utility (error) for different applications. We run the mechanisms 200 times and average the results. Note that the lower value of the following metrics, the better utility.

- The general utility was measured by Euclidean distance (km), i.e., E_{eu}, between the released location and the true location as defined in Sec. 2.2.
- The utility for weather apps or road traffic monitoring, i.e., "whether the released location is in the same region with the true location". We measure it by $E_r = ||R(\mathbf{s}^*), R(\mathbf{z})||_0$ where $R(\cdot)$ is a region query that returns the index of the region. Here we define the region size as 5×5 grid cells.
- The utility for POI mining or crowd monitoring during the pandemic, i.e., "whether the released location is the same category with the true location". We measure it by $E_{poi} = ||C(\mathbf{s}^*), C(\mathbf{z})||_0$ where $C(\cdot)$ returns the category of the corresponding location. We evaluated the location category of "restaurant".

6.2 Results and Analysis

P-LM vs. P-PIM. Fig.6 compares the utility of two proposed mechanisms P-LM and P-PIM for PGLP under the policy graphs G_{k9}, G_{k16} , G_{k25} and G_{poi} on Peopleflow dataset. The utility of P-PIM outperforms P-LM for different policy graphs and different ϵ since the sensitivity hull could achieve lower error bound.

[6] http://pflow.csis.u-tokyo.ac.jp/

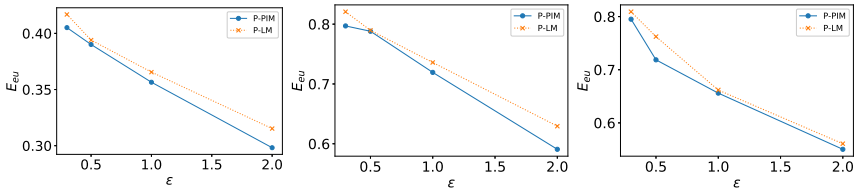

Fig. 6: Utility of P-LM vs. P-PIM with respect to G_{k9}, G_{k16} and G_{poi}.

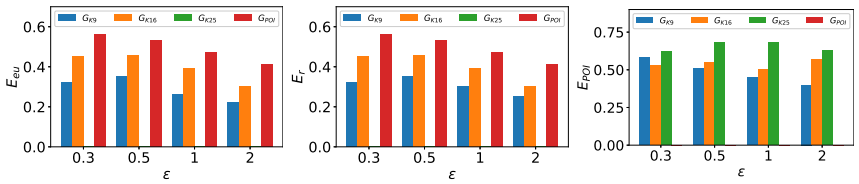

Fig. 7: Utility of different policy graphs.

Utility Gain by Tuning Policy Graphs. Fig.7 demonstrates that the utility of different applications can be boosted with appropriate policy graphs. We evaluate the three types of utility metrics using different policy graphs on Peopleflow dataset. Fig.7 shows that, for utility metrics E_{eu}, E_r and E_{poi}, the policy graphs with the best utility are G_{k9}, G_{k25} and G_{poi}, respectively. G_{k9} has smallest E_{eu} because of the least sensitivity. When the query is 5×5 region query, G_{k25} has the full usability (E_r=0). When the query is POI query like the one mentioned above, G_{poi} leads to full utility (E_r=0) since G_{poi} allows to disclose the semantic category of the true location while maintaining the indistinguishability among the set of locations with the same category. Note that E_{poi} is decreasing with larger ϵ for policy graph G_9 because the perturbed location has a higher probability to be the true location; while this effect is diminished in larger policy graphs such as G_{16} or G_{25} due to their larger sensitivities. We conclude that location policy graphs can be tailored flexibly for better utility-privacy trade-off.

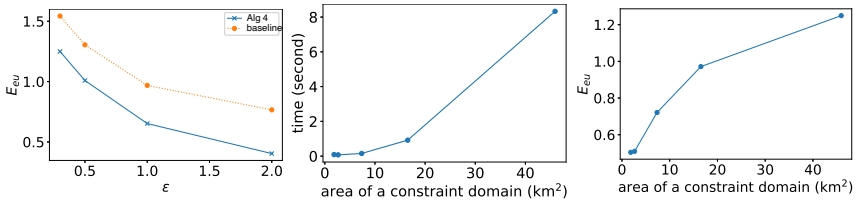

Fig. 8: Evaluation of Graph Repair.

Evaluation of Graph Repair. Fig. 8 shows the results of graph repair algorithms. We compare the proposed Algorithm 4 with a baseline method that repairs the problematic policy graph by adding an edge between the isolated node with its nearest node in the constrained domain. It shows that the utility measured by E_{eu} of Algorithm 4 is always better than the baseline but at the cost of higher runtime. Notably, the utility is decreasing (i.e., higher E_{eu}) with larger constrained domains because of larger policy graph (thus higher sensitivity); a larger constrained domain also incurs higher runtime because more isolated nodes need to be processed.

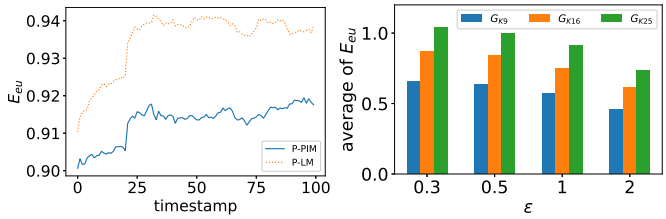

Fig. 9: Utility of Private Trajectory Release with P-LM and P-PIM.

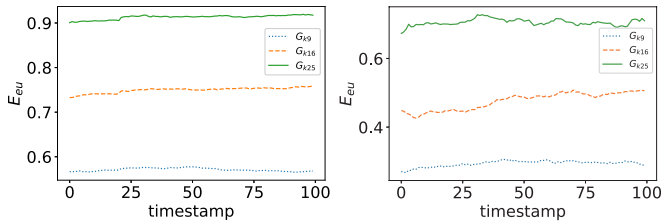

Fig. 10: Utility of Private Trajectory Release with different Policy Graphs.

Evaluation of Location Trace Release. We demonstrate the utility of private trajectory release with PGLP in Fig. 9 and Fig.10. In Fig. 9, we show the results of P-LM and P-PIM on the Geolife Dataset. We test 20 users' trajectories with 100 timestamps and report average E_{eu} at each timestamp. We can see P-PIM has higher utility than P-PIM, which in line with the results for single location release. The error E_{eu} is increasing along with timestamps due to the enlarged constrained domain, which is in line with Fig.8. The average of E_{eu} across 100 timestamps on different policy graphs, i.e., G_{k9}, G_{k16} and G_{k25} is also in accordance with the single location release in Fig.7. G_{k9} has the least average error of E_{eu} due to the smallest sensitivity.

In Fig.10, we show the utility of P-PIM with different policy graphs on two different datasets Geolife and Gowalla. The utility of G_{k9} is always the best over different timestamps for both datasets. In general, the Gowalla dataset has better utility than the Geolife dataset because the constraint domain of the Gowalla dataset is smaller. The reason is that the Gowalla dataset collects check-in locations that have an apparent mobility pattern, as shown in [13]. While Geolife dataset collects GPS trajectory with diverse transportation modes such as walk, bus, or train; thus, the trained Markov model is less accurate.

7 Conclusion

In this paper, we proposed a flexible and rigorous location privacy framework named PGLP, to release private location continuously under the real-world constraints with customized location policy graphs. We design an end-to-end private location trace release algorithm satisfying a pre-defined location privacy policy.

For future work, there are several promising directions. One is to study how to use the rich interface of PGLP for the utility-privacy tradeoff in the real-world location-based applications, such as carefully designing location privacy policies for COVID-19 contact tracing [4]. Another exciting direction is to design advanced mechanisms to achieve location privacy policies with less noise.

References

1. M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*, pages 901–914, 2013.
2. J. Bao, Y. Zheng, D. Wilkie, and M. Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.
3. C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In W. Jonker and M. Petkovi, editors, *Lecture Notes in Computer Science*, pages 185–199, 2005.
4. Y. Cao, S. Takagi, Y. Xiao, L. Xiong, and M. Yoshikawa. PANDA: policy-aware location privacy for epidemic surveillance. In *VLDB Demonstration Track, to appear*, 2020.
5. Y. Cao, Y. Xiao, L. Xiong, and L. Bai. PriSTE: from location privacy to spatiotemporal event privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1606–1609, 2019.
6. Y. Cao, Y. Xiao, L. Xiong, L. Bai, and M. Yoshikawa. PriSTE: protecting spatiotemporal event privacy in continuous location-based services. *Proc. VLDB Endow.*, 12(12):1866–1869, 2019.
7. Y. Cao, Y. Xiao, L. Xiong, L. Bai, and M. Yoshikawa. Protecting spatiotemporal event privacy in continuous location-based services. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.
8. Y. Cao, L. Xiong, M. Yoshikawa, Y. Xiao, and S. Zhang. ConTPL: controlling temporal privacy leakage in differentially private continuous data release. *VLDB Demonstration Track*, 11(12):2090–2093, 2018.
9. Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 821–832, 2017.
10. Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy in continuous data release under temporal correlations. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1281–1295, 2019.
11. K. Chatzikokolakis, C. Palamidessi, and M. Stronati. A predictive differentially-private mechanism for mobility traces. In E. D. Cristofaro and S. J. Murdoch, editors, *Lecture Notes in Computer Science*, number 8555, pages 21–41. 2014.
12. K. Chatzikokolakis, C. Palamidessi, and M. Stronati. Constructing elastic distinguishability metrics for location privacy. *Proceedings on Privacy Enhancing Technologies*, 2015(2):156–170, 2015.
13. E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.
14. C.-Y. Chow, M. F. Mokbel, and X. Liu. Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments. *GeoInformatica*, 15(2):351–380, 2011.
15. C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.
16. C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
17. L. Fan, L. Bonomi, L. Xiong, and V. Sunderam. Monitoring web browsing behavior with differential privacy. In *WWW*, pages 177–188, 2014.
18. K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. In *CCS*, pages 239–250, 2014.

19. M. Furuhata, M. Dessouky, F. Ordez, M.-E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.
20. S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, pages 1–6, 2012.
21. B. Gedik and L. Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2008.
22. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, pages 31–42, 2003.
23. Y. Han, S. Li, Y. Cao, Q. Ma, and M. Yoshikawa. Voice-indistinguishability: Protecting voiceprint in privacy-preserving speech data release. In *IEEE ICME*, 2020.
24. M. Hardt and K. Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.
25. X. He, A. Machanavajjhala, and B. Ding. Blowfish privacy: tuning privacy-utility trade-offs using policies. pages 1447–1458, 2014.
26. M. Ingle, O. Nash, V. Nguyen, J. Petrie, J. Schwaber, Z. Szabo, M. Voloshin, T. White, and H. Xue. Slowing the spread of infectious diseases using crowdsourced data. *IEEE Data Engineering Bulletin*, page 12, 2020.
27. D. Kifer and A. Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.
28. N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE ICDE*, pages 106–115, 2007.
29. N. Li, M. Lyu, D. Su, and W. Yang. *Differential Privacy: From Theory to Practice*. 2016.
30. Y. Luo, N. Tang, G. Li, W. Li, T. Zhao, and X. Yu. DEEPEYE: a data science system for monitoring and exploring COVID-19 data. *IEEE Data Engineering Bulletin*, page 12, 2020.
31. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. L-diversity: privacy beyond k-anonymity. In *IEEE ICDE*, pages 24–24, 2006.
32. C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32, 2013.
33. B. Pej and D. Desfontaines. SoK: differential privacies. In *Proceedings on Privacy Enhancing Technologies Symposium*, 2020.
34. V. Primault, A. Boutet, S. B. Mokhtar, and L. Brunie. The long road to computational location privacy: A survey. *IEEE Communications Surveys Tutorials*, pages 1–1, 2018.
35. R. Recabarren and B. Carbunar. What does the crowd say about you? evaluating aggregation-based location privacy. In *WPES*, volume 2017, pages 156–176, 2017.
36. S. Song, Y. Wang, and K. Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *SIGMOD*, pages 1291–1306, 2017.
37. L. Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
38. S. Takagi, Y. Cao, Y. Asano, and M. Yoshikawa. Geo-graph-indistinguishability: Protecting location privacy for LBS over road networks. In *DBSec*, pages 143–163, 2019.

39. Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *CCS*, pages 1298–1309, 2015.
40. Y. Xiao, L. Xiong, S. Zhang, and Y. Cao. LocLok: location cloaking with differential privacy via hidden markov model. *Proc. VLDB Endow.*, 10(12):1901–1904, 2017.
41. Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. GeoLife2.0: a location-based social networking service. In *IEEE MDM*, pages 357–358, 2009.

Appendix A An example of Isolated Node

Intuition. We examine the privacy guarantee of P-PIM w.r.t. $\mathcal{G}^{\mathcal{C}}$ in Fig.3(a). According to K-norm Mechanism [24] in Definition 9, P-PIM guarantees that, for any two neighbors $\mathbf{s}_i$ and $\mathbf{s}_j$, their difference is bounded in the convex body K, i.e. $f(\mathbf{s}_i) - f(\mathbf{s}_j) \in K$. Geometrically, for a location $\mathbf{s}$, all other locations in the convex body of $K + f(\mathbf{s})$ are ϵ-indistinguishable with $\mathbf{s}$.

Example 1 (Disconnected but Not Isolated Node). In Figure 11, $\mathbf{s}_2$ is disconnected under constraint $\mathcal{C} = \{\mathbf{s}_2, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$. However, $\mathbf{s}_2$ is not isolated because $f(\mathbf{s}_2) + K$ contains $f(\mathbf{s}_4)$ and $f(\mathbf{s}_5)$. Hence, $\mathbf{s}_2$ and other nodes in $\mathcal{C}$ are indistinguishable.

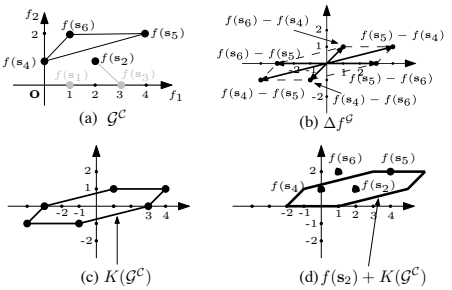

Fig. 11: (a) A policy graph under $\mathcal{C} = \{\mathbf{s}_2, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$; (b) the $\Delta f^{\mathcal{G}}$ of vectors $f(\mathbf{s}_i) - f(\mathbf{s}_j)$; (c) the sensitivity hull $K(\mathcal{G}^{\mathcal{C}})$ covering the $\Delta f^{\mathcal{G}}$; (d) the shape $f(\mathbf{s}_2) + K(\mathcal{G}^{\mathcal{C}})$ containing $f(\mathbf{s}_4)$ and $f(\mathbf{s}_5)$. That is to say, $\mathbf{s}_2$ is indistinguishable with $\mathbf{s}_4$ and $\mathbf{s}_5$.

Appendix B Policy Graph Repair Algorithm

Figure 4(a) shows the graph under constraint $\mathcal{C} = \{\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5, \mathbf{s}_6\}$. Then $\mathbf{s}_3$ is exposed because $f(\mathbf{s}_3) + K^{\mathcal{G}}$ contains no other node. To satisfy the PGLP without isolated nodes, we need to connect $\mathbf{s}_3$ to another node in $\mathcal{C}$, i.e. $\mathbf{s}_4$, $\mathbf{s}_5$ or $\mathbf{s}_6$.

If $\mathbf{s}_3$ is connected to $\mathbf{s}_4$, then Figure 4(b) shows the new graph and its sensitivity hull. By adding two new edges $\{f(\mathbf{s}_3) - f(\mathbf{s}_4), f(\mathbf{s}_4) - f(\mathbf{s}_3)\}$ to $\Delta f^{\mathcal{G}}$, the shaded areas are attached to the sensitivity hull. Similarly, Figures 4(c) and (d) show the new sensitivity hulls when $\mathbf{s}_3$ is connected to $\mathbf{s}_6$ and $\mathbf{s}_5$ respectively. Because the smallest area of $K^{\mathcal{G}}$ is in Figure 4(b), the repaired graph is $\mathcal{G}^{\mathcal{C}} \vee \overline{\mathbf{s}_3\mathbf{s}_4}$, i.e., add adge $\overline{\mathbf{s}_3\mathbf{s}_4}$ to the graph $\mathcal{G}^{\mathcal{C}}$.

Theorem 8. *Algorithm 4 takes $O(m^2 log(m))$ time where m is the number of valid nodes (with edge) in the policy graph.*

Algorithm. Algorithm 4 derives the minimum protectable graph for location data when an isolated node is detected. We can connect the isolated node $\mathbf{s}_i$ to

the rest (at most m) nodes, generating at most m convex hulls where $m = |\mathcal{V}|$ is the number of valid nodes. In two-dimensional space, it only takes $O(mlog(m))$ time to find a convex hull. To derive *Area* of a shape, we exploit the computation of determinant whose intrinsic meaning is the VOLUME of the column vectors. Therefore, 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 $O(m^2log(m))$ time where m is the number of valid nodes. Note that Algorithms 3 and 4 can also be combined together to protect any disconnected nodes.

Appendix C Related Works

C.1 Differential Privacy

While differential privacy [15] has been accepted as a standard notion for privacy protection, the concept of standard differential privacy is not generally applicable for complicated data types. Many variants of differential privacy have been proposed, such as Pufferfish privacy [27], Geo-Indistinguishability [1] and Voice-Indistinguishability [23] (see Survey [33]). Blowfish privacy [25] 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. Our definition of PGLP is inspired by Blowfish framework. Notably, we find that the policy graph may not be viable under temporal correlations represented by Markov model, which was not considered in the previous work. This is also related to another line of works studying how to achieve differential privacy under temporal correlations [8–10, 36].

C.2 Location Privacy

A close line of works focus on extending differential privacy to location setting. The first DP-based location privacy model is Geo-Indistinguishability [1], which scales the sensitivity of two locations to their Euclidean distance. Hence, it is suitable for proximity-based applications. Following by Geo-Indistinguishability, several location privacy notions [38,40] have been proposed based on differential privacy. A recent DP-based location privacy, spatiotemporal event privacy [5–7], proposed a new representation for secrets in spatiotemporal data called spatiotemoral events using Boolean expression. It is essentially different from this work since here we are considering the traditional representation of secrets, i.e., each single location or a sequence of locations.

Several works considered Markov models for improving utility of released location traces or web browsing activities [11, 17], but did not consider the inference risks when an adversary has the knowledge of the Markov model. Xiao et al. [39] studied how to protect the true location if a user's movement follows Markov model. The technique can be viewed as a special instantiation of PGLP. In addition, PGLP uses a policy graph to tune the privacy and utility to meet diverse the requirement of location-based applications.